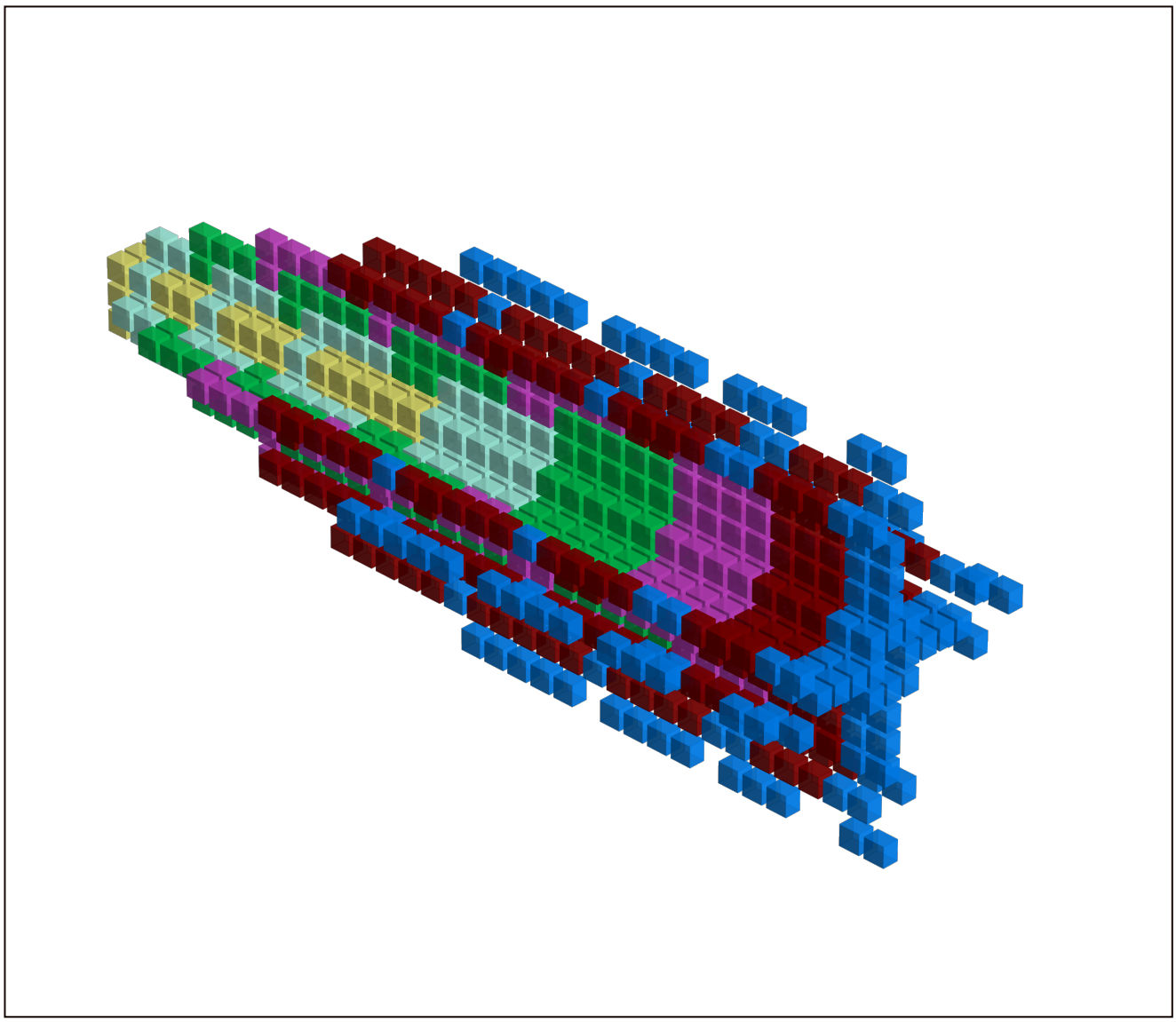
Psychedelia Syndrome

Findings and Recommendations



Llamasoft Corporation

Contents

Patterns

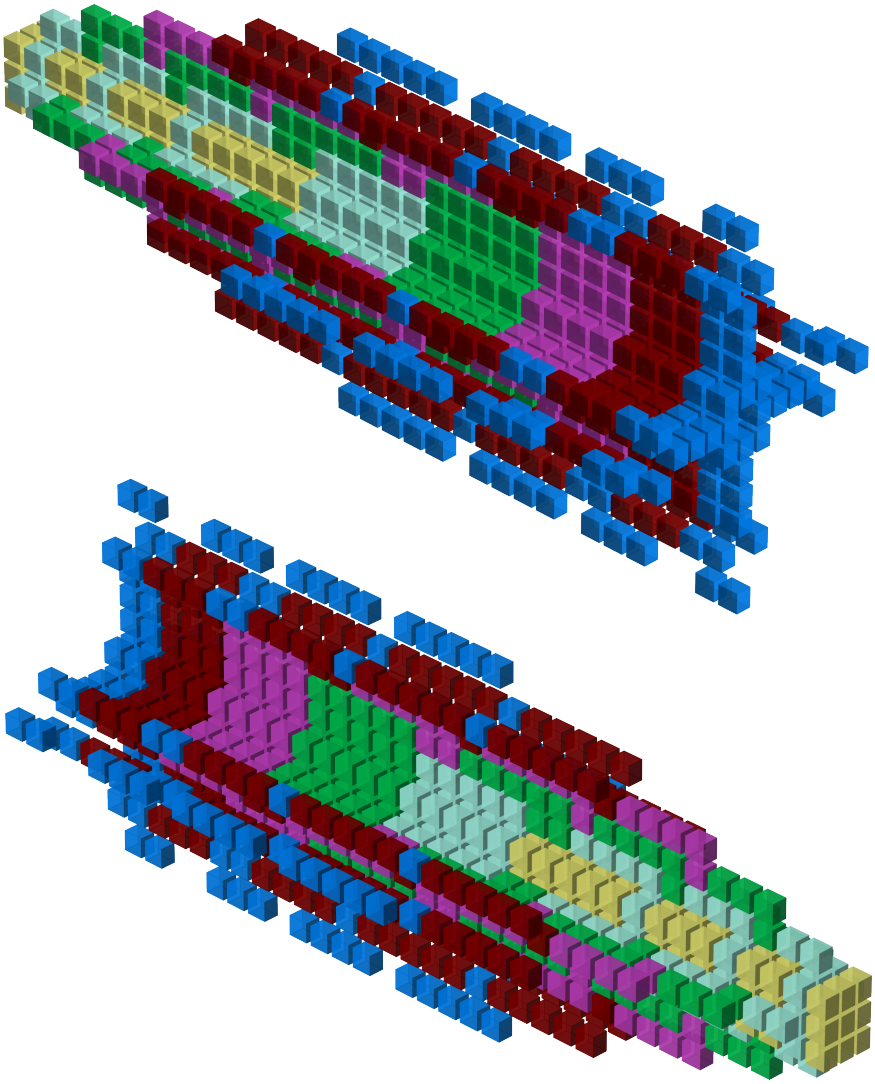


Figure 1.1: Evolution of the ‘Star One’ pattern.

```
starOneXPosArray
  .BYTE $00,$01,$01,$01,$00,$FF,$FF,$FF,$55        ;         5
  .BYTE $00,$02,$00,$FE,$55                        ;
  .BYTE $00,$03,$00,$FD,$55                        ;        4 4
  .BYTE $00,$04,$00,$FC,$55                        ;         3
  .BYTE $FF,$01,$05,$05,$01,$FF,$FB,$FB,$55        ;         2
  .BYTE $00,$07,$00,$F9,$55                        ;         1
  .BYTE $55                                        ;    4    000    4
starOneYPosArray                                   ; 5  3210 0123   5
  .BYTE $FF,$FF,$00,$01,$01,$01,$00,$FF,$55        ;    4    000    4
  .BYTE $FE,$00,$02,$00,$55                        ;         1
  .BYTE $FD,$00,$03,$00,$55                        ;         2
  .BYTE $FC,$00,$04,$00,$55                        ;         3
  .BYTE $FB,$FB,$FF,$01,$05,$05,$01,$FF,$55        ;        4 4
  .BYTE $F9,$00,$07,$00,$55                        ;
  .BYTE $55                                        ;         5
```

Listing 1.1: Source code for the Star.

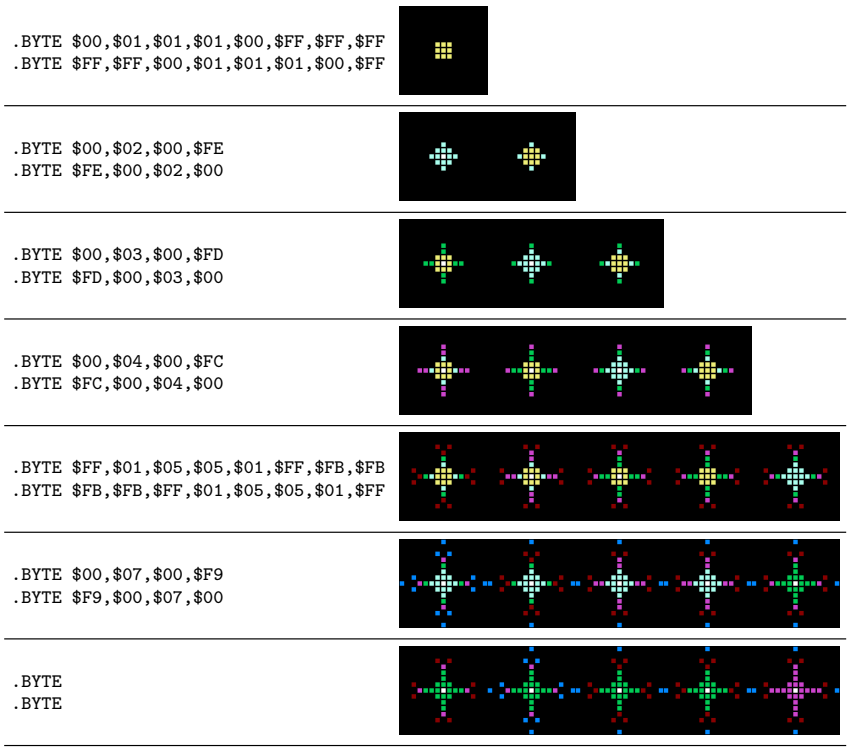


Figure 1.2: Pattern Progression for 'Star One'

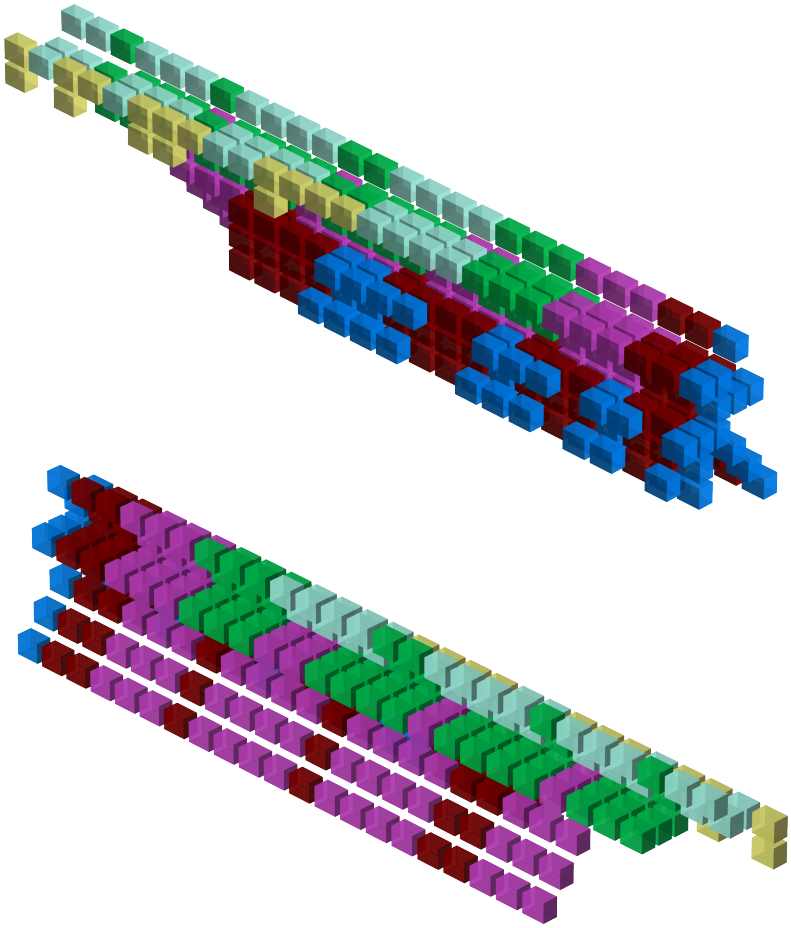


Figure 1.3: The 'Twist'.

```
theTwistXPosArray  .BYTE $00,$55                          ;      1
                   .BYTE $01,$02,$55                      ;    01
                   .BYTE $01,$02,$03,$55                  ;  6 222
                   .BYTE $01,$02,$03,$04,$55              ;  543
                   .BYTE $00,$00,$00,$55                  ; 5 4 3
                   .BYTE $FF,$FE,$55                      ;   4  3
                   .BYTE $FF,$55                          ;       3
                   .BYTE $55
theTwistYPosArray  .BYTE $FF,$55
                   .BYTE $FF,$FE,$55
                   .BYTE $00,$00,$00,$55
                   .BYTE $01,$02,$03,$04,$55
                   .BYTE $01,$02,$03,$55
                   .BYTE $01,$02,$55
                   .BYTE $00,$55
                   .BYTE $55
```

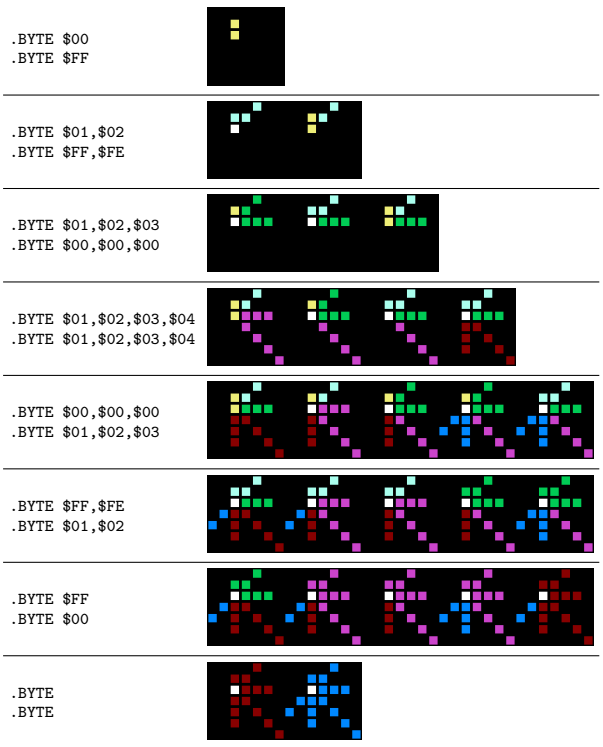


Figure 1.4: Pattern Progression for 'The Twister'

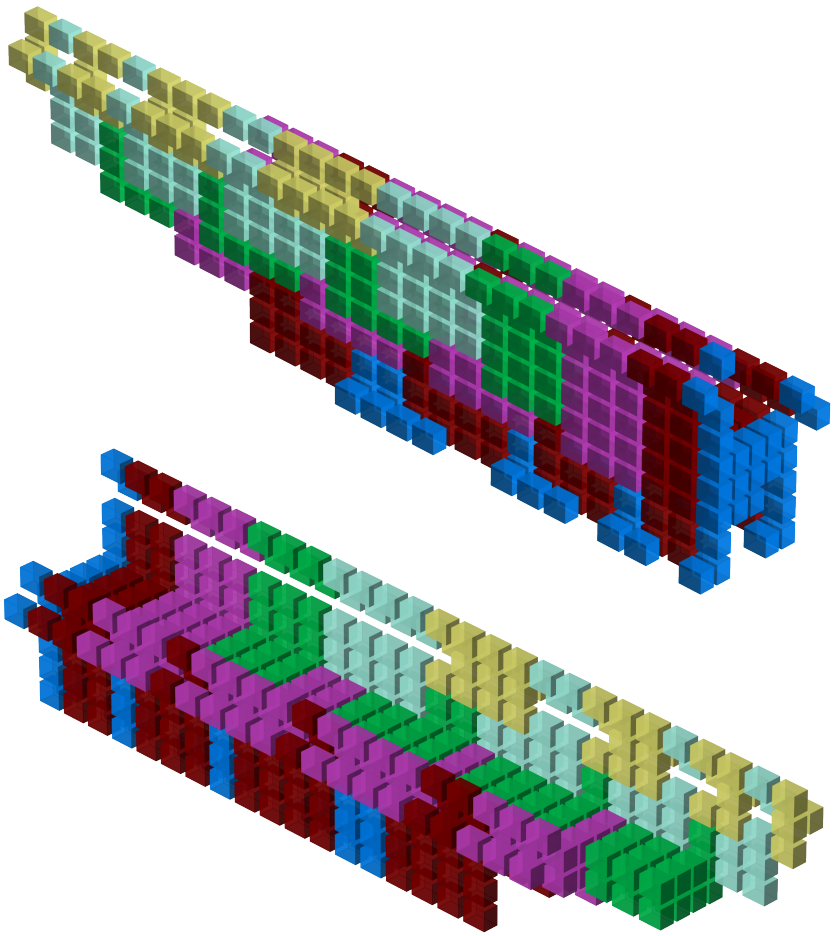


Figure 1.5: 'La Llamita'.

```
laLlamitaXPosArray   .BYTE $00,$FF,$00,$55                    ;  0
                     .BYTE $00,$00,$55                        ; 06
                     .BYTE $01,$02,$03,$00,$01,$02,$03,$55    ;  0
                     .BYTE $04,$05,$06,$04,$00,$01,$02,$55    ;  1     3
                     .BYTE $04,$00,$04,$00,$04,$55            ;  12223 3
                     .BYTE $FF,$03,$55                        ;  22223
                     .BYTE $00,$55                            ;  333 4
laLlamitaYPosArray   .BYTE $FF,$00,$01,$55                    ;  4   4
                     .BYTE $02,$03,$55                        ; 54  54
                     .BYTE $03,$03,$03,$04,$04,$04,$04,$55
                     .BYTE $03,$02,$03,$04,$05,$05,$05,$55
                     .BYTE $05,$06,$06,$07,$07,$55
                     .BYTE $07,$07,$55
                     .BYTE $00,$55
```

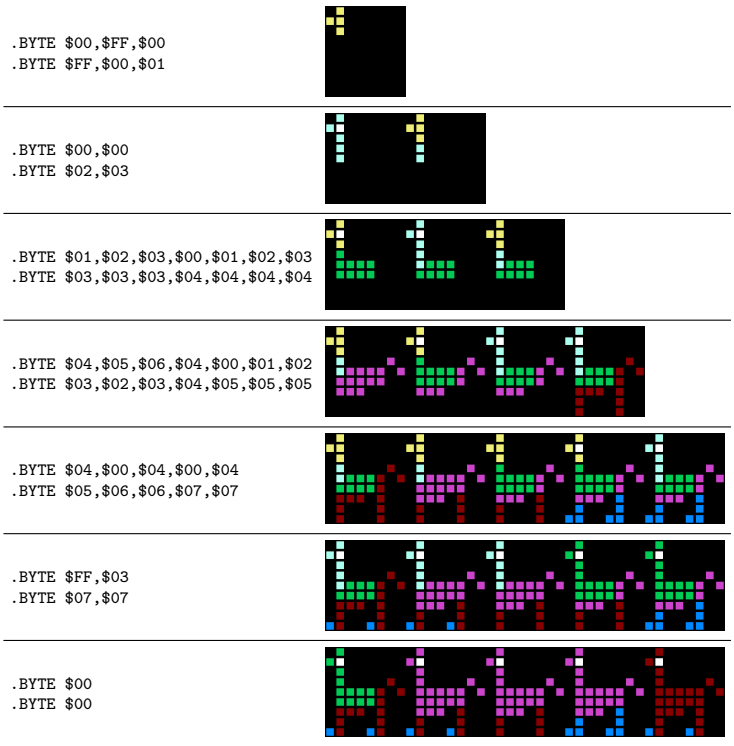


Figure 1.6: Pattern Progression for 'La Llamita'

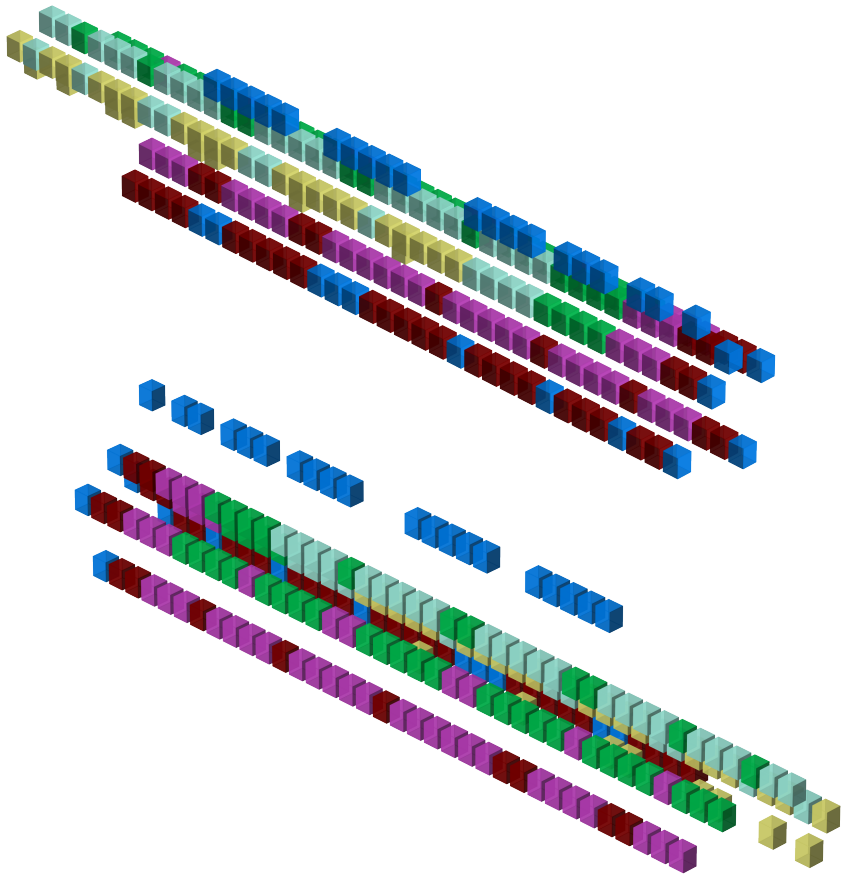


Figure 1.7: 'Star Two'.

```
starTwoXPosArray   .BYTE $FF,$55                    ;     1
                   .BYTE $00,$55                    ;    0  2
                   .BYTE $02,$55                    ;     6
                   .BYTE $01,$55                    ; 4
                   .BYTE $FD,$55                    ;      3
                   .BYTE $FE,$55                    ;  5
                   .BYTE $00,$55
starTwoYPosArray   .BYTE $FF,$55
                   .BYTE $FE,$55
                   .BYTE $FF,$55
                   .BYTE $02,$55
                   .BYTE $01,$55
                   .BYTE $FC,$55
                   .BYTE $00,$55
```

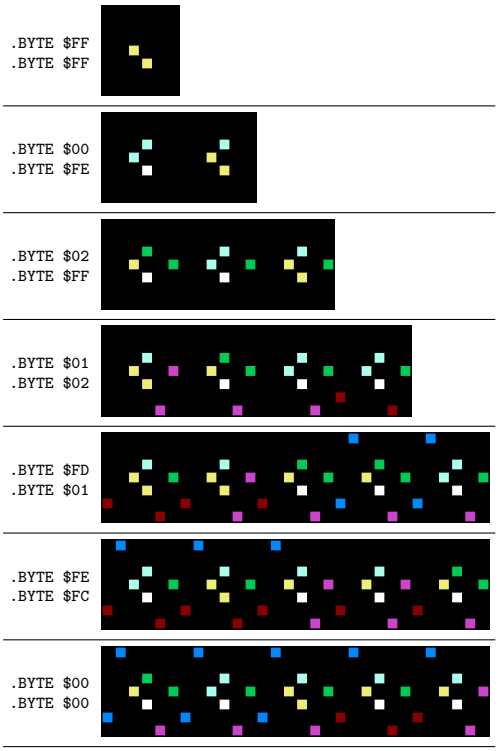


Figure 1.8: Pattern Progression for 'Star Two'

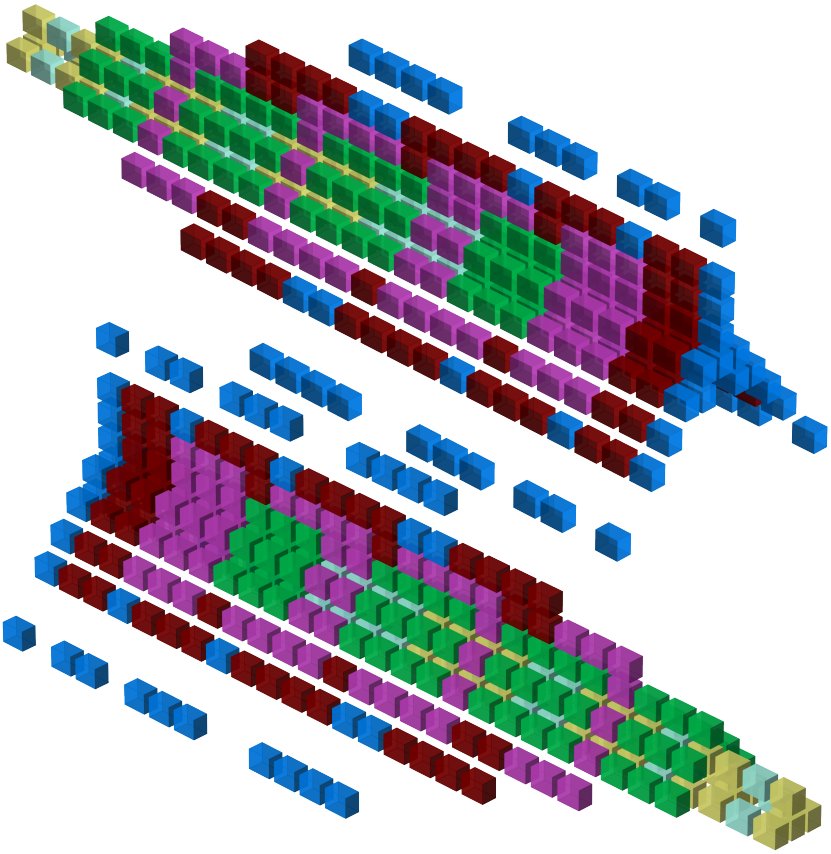


Figure 1.9: 'Deltoid'.

```
deltoidXPosArray   .BYTE $00,$01,$FF,$55            ;        5
                   .BYTE $00,$55                    ;
                   .BYTE $00,$01,$02,$FE,$FF,$55    ;        4
                   .BYTE $00,$03,$FD,$55            ;        3
                   .BYTE $00,$04,$FC,$55            ;        2
                   .BYTE $00,$06,$FA,$55            ;      202
                   .BYTE $00,$55                    ;    20602
deltoidYPosArray   .BYTE $FF,$00,$00,$55            ;    3     3
                   .BYTE $00,$55                    ;   4       4
                   .BYTE $FE,$FF,$00,$00,$FF,$55    ;
                   .BYTE $FD,$01,$01,$55            ; 5           5
                   .BYTE $FC,$02,$02,$55
                   .BYTE $FA,$04,$04,$55
                   .BYTE $00,$55
```

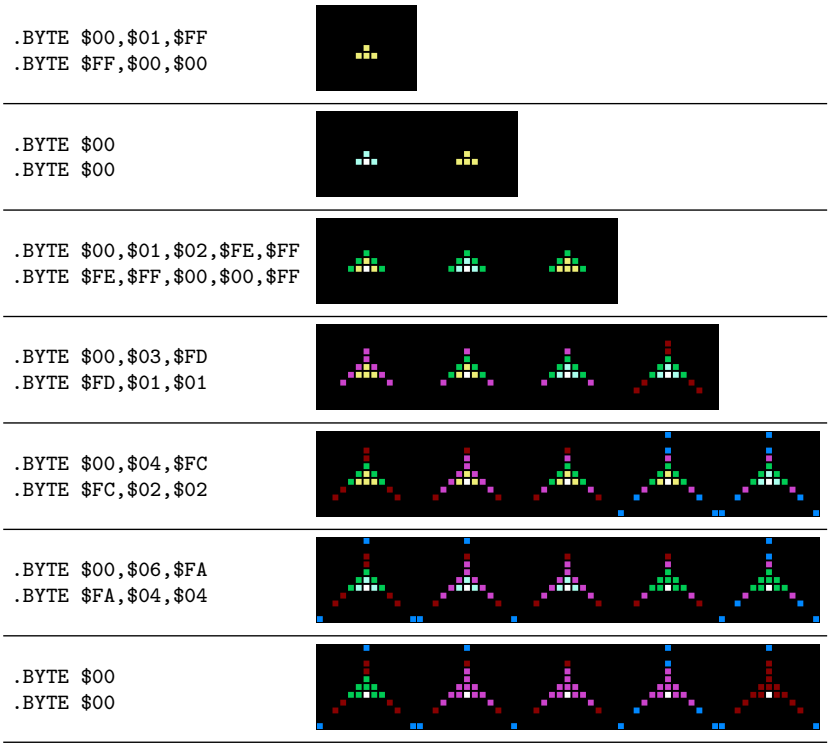


Figure 1.10: Pattern Progression for 'Deltoid'

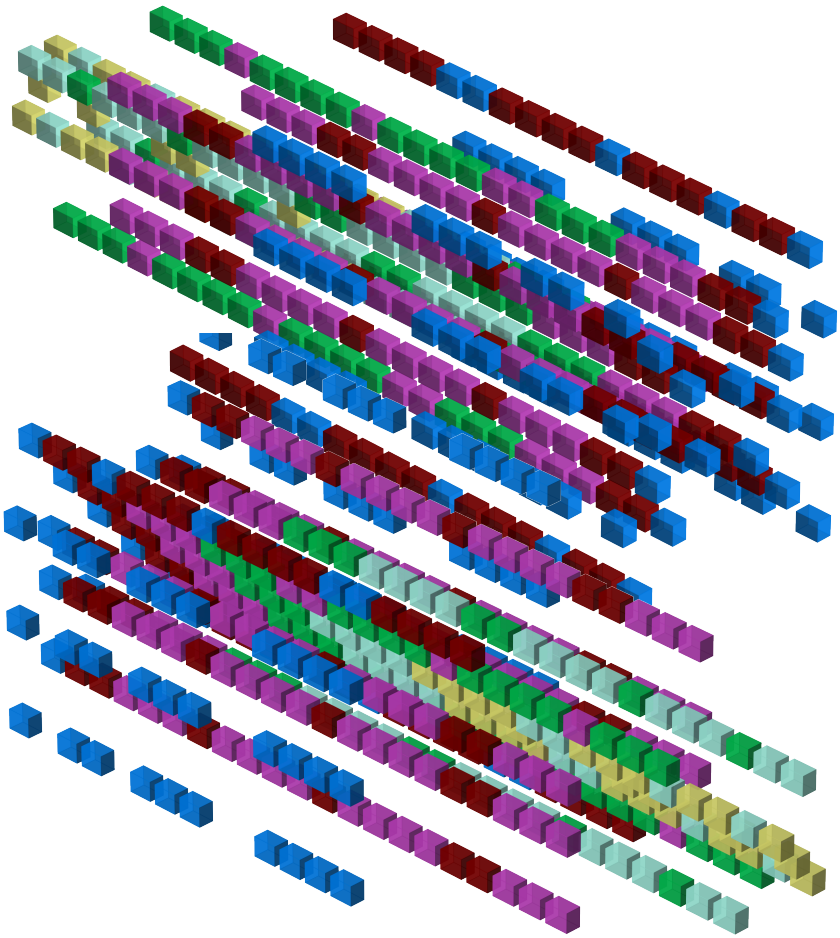


Figure 1.11: ‘Diffused’.

```
diffusedXPosArray  .BYTE $FF,$01,$55                    ; 5
                   .BYTE $FE,$02,$55                    ;              4
                   .BYTE $FD,$03,$55                    ;    3
                   .BYTE $FC,$04,$FC,$FC,$04,$04,$55    ;           2
                   .BYTE $FB,$05,$55                    ; 5    1         5
                   .BYTE $FA,$06,$FA,$FA,$06,$06,$55    ;    3     0   3
                   .BYTE $00,$55                        ;         6
diffusedYPosArray  .BYTE $01,$FF,$55                    ;    3   0     3
                   .BYTE $FE,$02,$55                    ; 5         1    5
                   .BYTE $03,$FD,$55                    ;     2
                   .BYTE $FC,$04,$FF,$01,$FF,$01,$55    ;              3
                   .BYTE $05,$FB,$55                    ;   4
                   .BYTE $FA,$06,$FE,$02,$FE,$02,$55    ;                5
                   .BYTE $00,$55
```

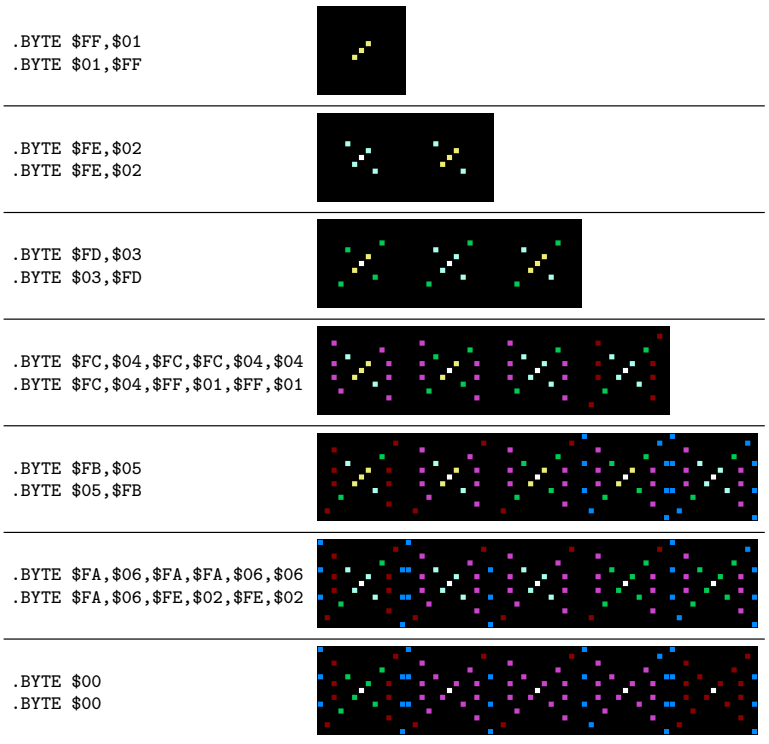


Figure 1.12: Pattern Progression for 'Diffused'

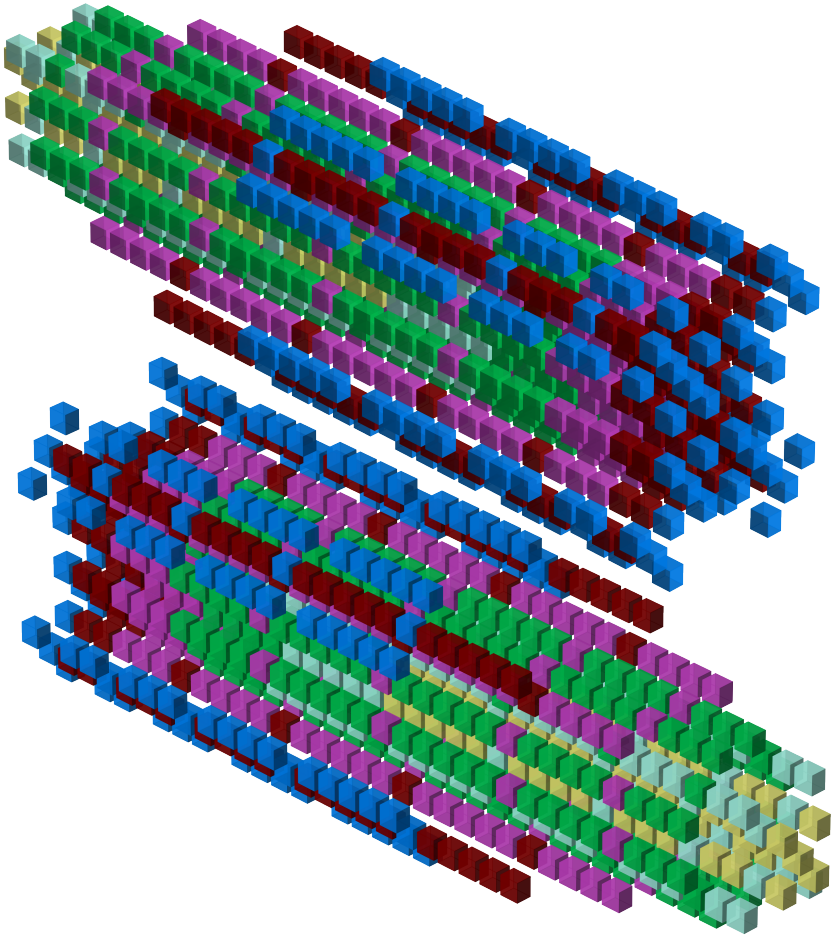


Figure 1.13: ‘Multi-Cross’.

```
multicrossXPosArray
        .BYTE $01,$01,$FF,$FF,$55                       ;
        .BYTE $02,$02,$FE,$FE,$55                       ;   5     5
        .BYTE $01,$03,$03,$01,$FF,$FD,$FD,$FF,$55       ;  4       4
        .BYTE $03,$03,$FD,$FD,$55                       ; 5 3 2 2 3 5
        .BYTE $04,$04,$FC,$FC,$55                       ;    1   1
        .BYTE $03,$05,$05,$03,$FD,$FB,$FB,$FD,$55       ;   2 0 0 2
        .BYTE $00,$55                                   ;     6
multicrossYPosArray                                     ;   2 0 0 2
        .BYTE $FF,$01,$01,$FF,$55                       ;    1   1
        .BYTE $FE,$02,$02,$FE,$55                       ; 5 3 2 2 3 5
        .BYTE $FD,$FF,$01,$03,$03,$01,$FF,$FD,$55       ;  4       4
        .BYTE $FD,$03,$03,$FD,$55                       ;   5     5
        .BYTE $FC,$04,$04,$FC,$55
        .BYTE $FB,$FD,$03,$05,$05,$03,$FD,$FB,$55
        .BYTE $00,$55
```

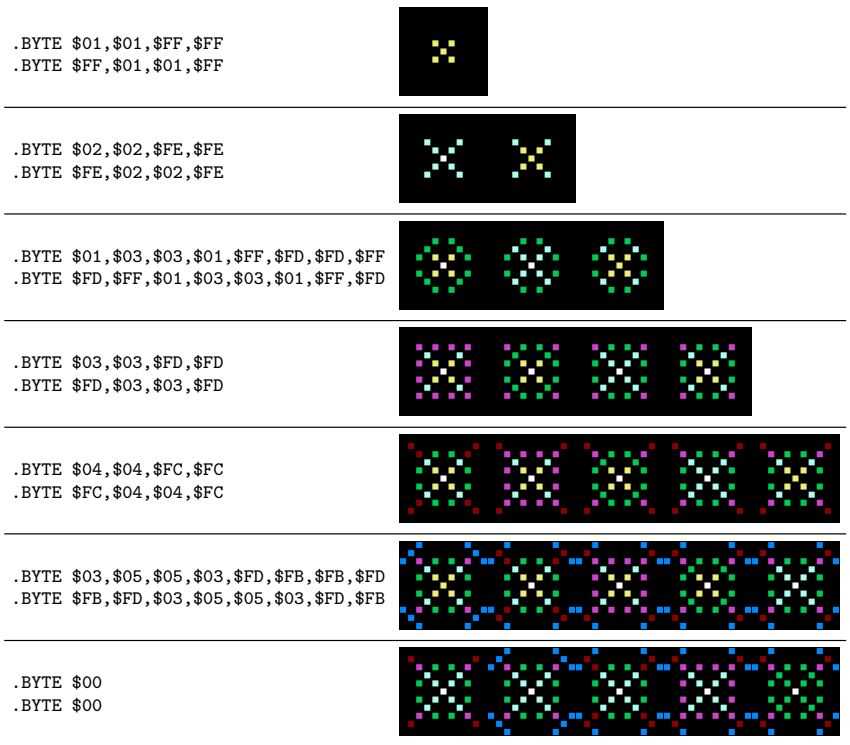


Figure 1.14: Pattern Progression for 'Multi-Cross'

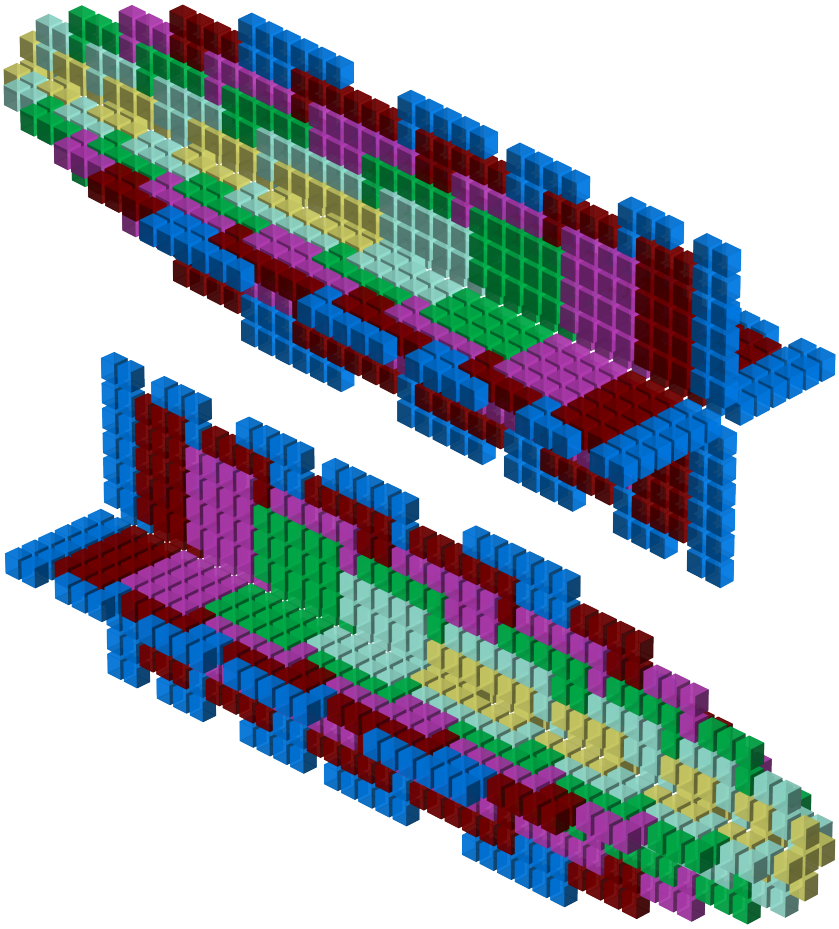


Figure 1.15: 'Pulsar'.

```
pulsarXPosArray  .BYTE  $00,$01,$00,$FF,$55        ;
                 .BYTE  $00,$02,$00,$FE,$55        ;        5
                 .BYTE  $00,$03,$00,$FD,$55        ;        4
                 .BYTE  $00,$04,$00,$FC,$55        ;        3
                 .BYTE  $00,$05,$00,$FB,$55        ;        2
                 .BYTE  $00,$06,$00,$FA,$55        ;        1
                 .BYTE  $00,$55                    ;        0
pulsarYPosArray  .BYTE  $FF,$00,$01,$00,$55        ; 5432106012345
                 .BYTE  $FE,$00,$02,$00,$55        ;        0
                 .BYTE  $FD,$00,$03,$00,$55        ;        1
                 .BYTE  $FC,$00,$04,$00,$55        ;        2
                 .BYTE  $FB,$00,$05,$00,$55        ;        3
                 .BYTE  $FA,$00,$06,$00,$55        ;        4
                 .BYTE  $00,$55                    ;        5
```

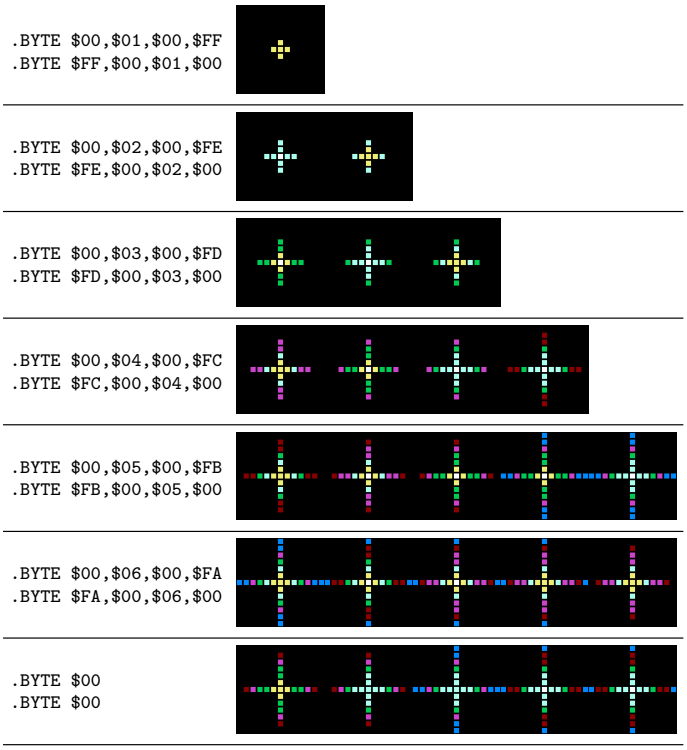


Figure 1.16: Pattern Progression for 'Pulsar'

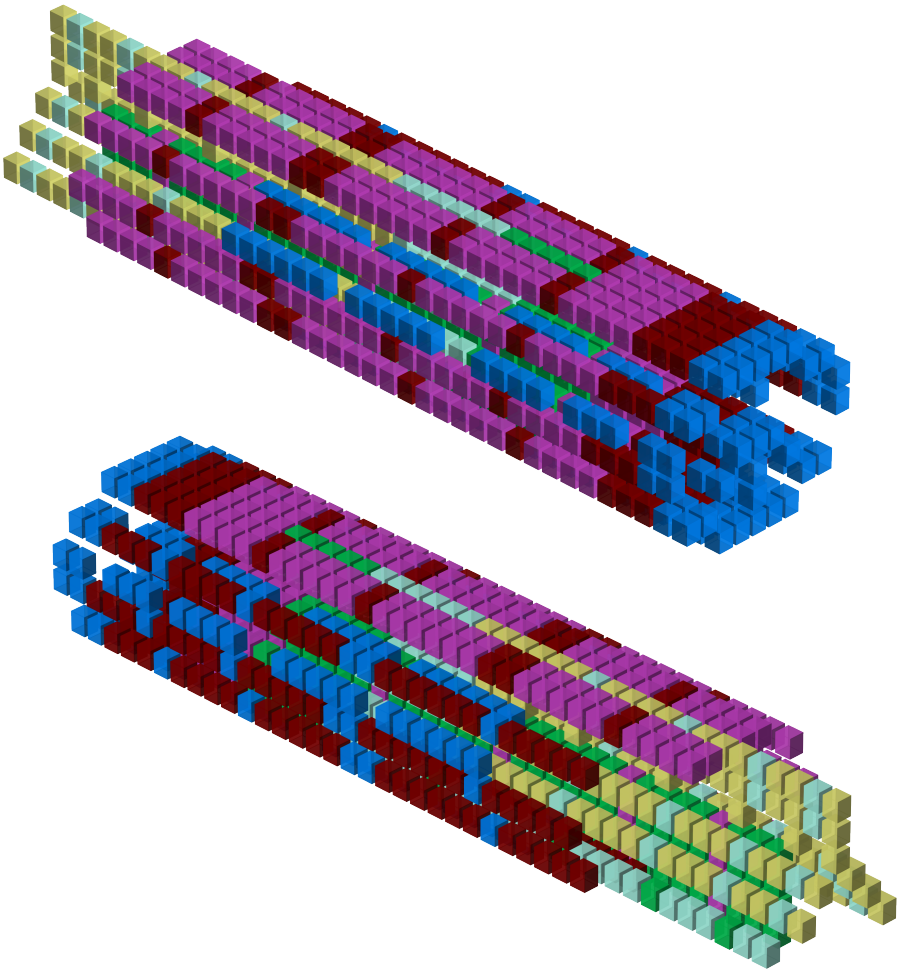


Figure 1.17: 'Custom Pattern 1'.

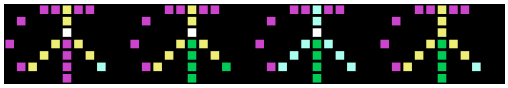
```
; customPattern0XPosArray                              ;
        .BYTE $00,$00,$00,$FF,$FE,$FD,$01,$02,$55 ;     33033
        .BYTE $00,$03,$55                         ;  35   0   54
        .BYTE $00,$00,$00,$00,$00,$55             ; 5     6     5
        .BYTE $00,$FF,$FE,$FC,$FB,$FC,$01,$02,$55 ; 3    020    4
        .BYTE $00,$04,$05,$04,$FF,$01,$55         ;     0 2 0
        .BYTE $00,$FD,$FB,$03,$05,$02,$FE,$55     ;  30   2   14
        .BYTE $00,$55                             ;     54245
; customPattern0YPosArray
        .BYTE $00,$FF,$FE,$01,$02,$03,$01,$02,$55
        .BYTE $00,$03,$55
        .BYTE $00,$01,$02,$03,$04,$55
        .BYTE $00,$FE,$FE,$FF,$01,$03,$FE,$FE,$55
        .BYTE $00,$FF,$01,$03,$04,$04,$55
        .BYTE $00,$FF,$00,$FF,$00,$04,$04,$55
        .BYTE $00,$55
```

```
.BYTE $00,$00,$00,$FF,$FE,$FD,$01,$02
.BYTE $00,$FF,$FE,$01,$02,$03,$01,$02
```

```
.BYTE $00,$03
.BYTE $00,$03
```

```
.BYTE $00,$00,$00,$00,$00
.BYTE $00,$01,$02,$03,$04
```

```
.BYTE $00,$FF,$FE,$FC,$FB,$FC,$01,$02
.BYTE $00,$FE,$FE,$FF,$01,$03,$FE,$FE
```



```
.BYTE $00,$04,$05,$04,$FF,$01
.BYTE $00,$FF,$01,$03,$04,$04
```

```
.BYTE $00,$FD,$FB,$03,$05,$02,$FE
.BYTE $00,$FF,$00,$FF,$00,$04,$04
```

```
.BYTE $00
.BYTE $00
```

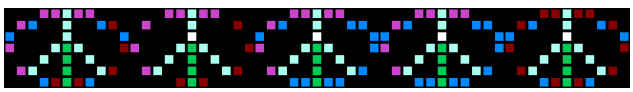


Figure 1.18: Pattern Progression for 'Custom Pattern 1'

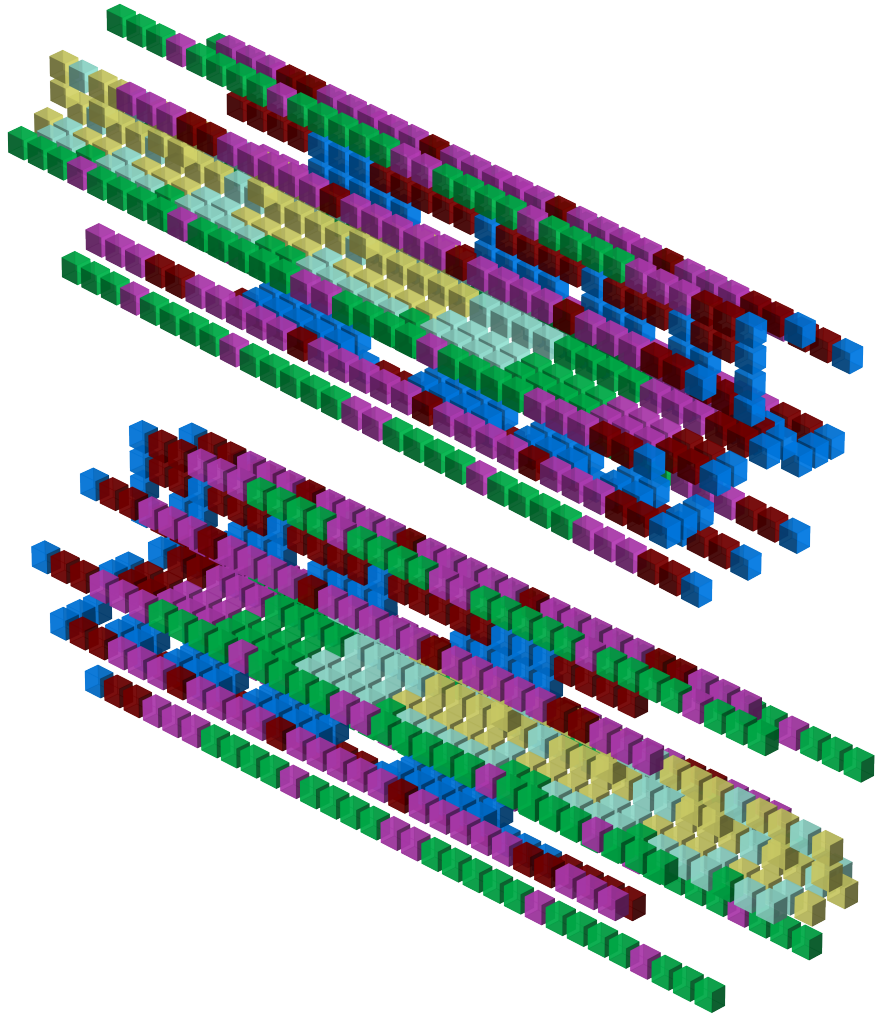


Figure 1.19: 'Custom Pattern 2'.

```
; customPattern1XPosArray                    ;        3
        .BYTE $00,$00,$FF,$01,$55            ;     4  5  4
        .BYTE $00,$FE,$02,$55                ;        6
        .BYTE $00,$00,$FA,$06,$03,$FD,$55    ; 3      1      3
        .BYTE $00,$FD,$03,$FB,$05,$55        ;        7
        .BYTE $00,$00,$00,$55                ;      21 12
        .BYTE $00,$00,$FC,$04,$03,$FD,$55    ;  466      664
        .BYTE $00,$55                        ;
; customPattern1YPosArray                    ;     3  5  3
        .BYTE $00,$FF,$01,$01,$55
        .BYTE $00,$01,$01,$55
        .BYTE $00,$FC,$FF,$FF,$05,$05,$55
        .BYTE $00,$FD,$FD,$02,$02,$55
        .BYTE $00,$05,$FD,$55
        .BYTE $00,$FE,$02,$02,$02,$02,$55
        .BYTE $00,$55
```

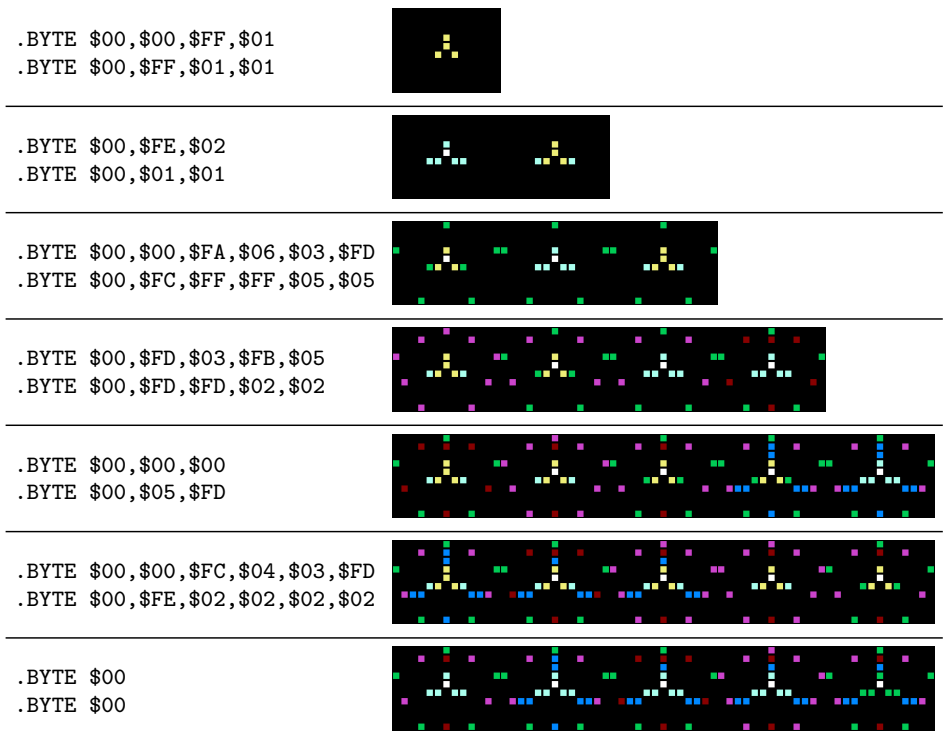


Figure 1.20: Pattern Progression for 'Custom Pattern 2'

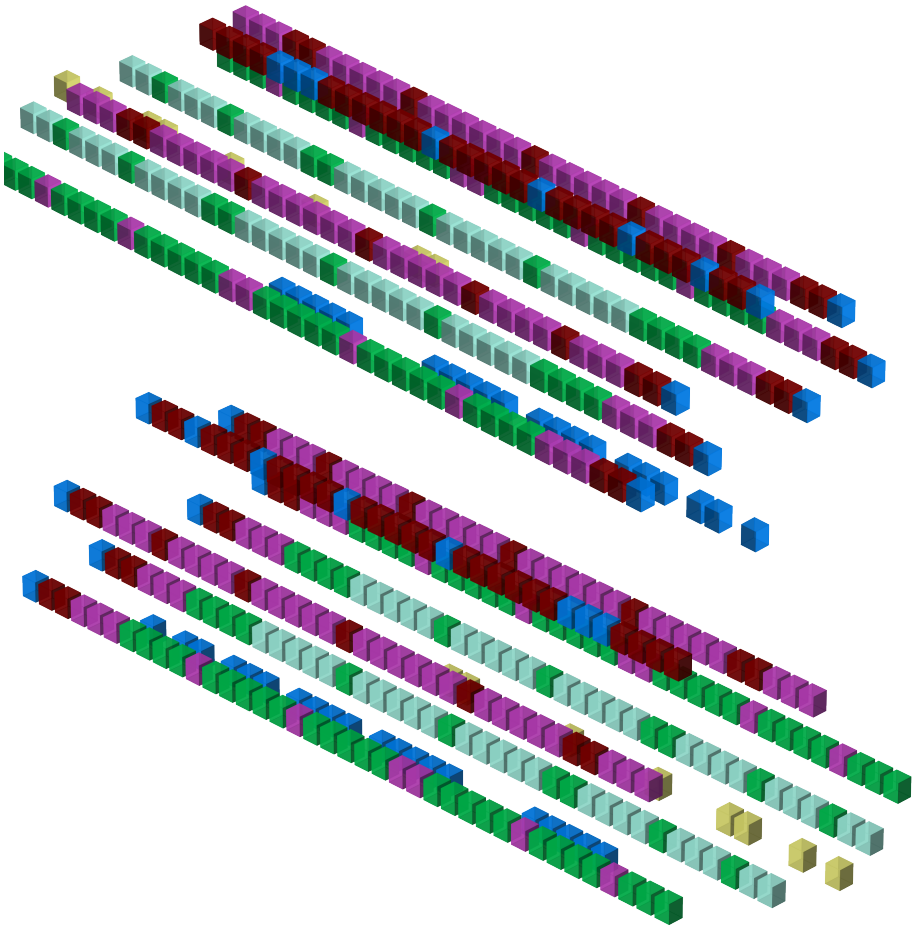


Figure 1.21: 'Custom Pattern 3'.

```
; customPattern2XPosArray      ;          5
        .BYTE $00,$55           ;       8    8
        .BYTE $00,$FD,$03,$55   ;    4           4
        .BYTE $00,$F9,$07,$55   ;
        .BYTE $00,$FB,$05,$55   ;
        .BYTE $00,$00,$55       ; 3    2   9   2    3
        .BYTE $00,$00,$55       ;
        .BYTE $00,$55           ;
        .BYTE $FE,$02,$55       ;
        .BYTE $00,$55           ;          6
; customPattern2YPosArray
        .BYTE $00,$55
        .BYTE $00,$00,$00,$55
        .BYTE $00,$00,$00,$55
        .BYTE $00,$FD,$FD,$55
        .BYTE $00,$FB,$55
        .BYTE $00,$04,$55
        .BYTE $00,$55
        .BYTE $FC,$FC,$55
        .BYTE $00,$55
```

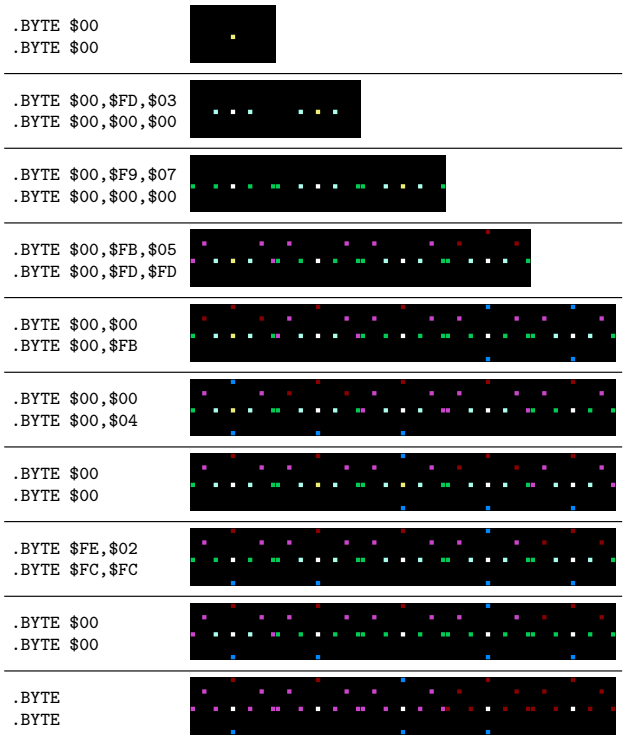


Figure 1.22: Pattern Progression for 'Custom Pattern 3'

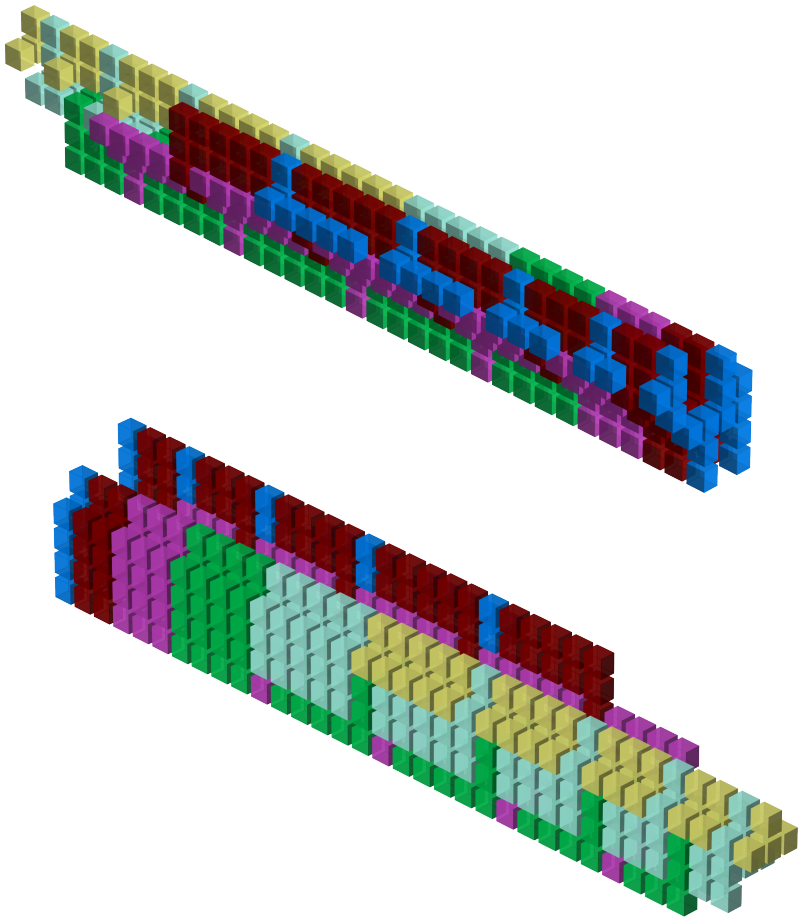


Figure 1.23: 'Custom Pattern 4'.

```
; customPattern3XPosArray                ;  5
        .BYTE $00,$01,$01,$02,$55        ; 66  1
        .BYTE $00,$00,$01,$02,$02,$55    ;  4 711
        .BYTE $00,$00,$00,$02,$55        ;    4222
        .BYTE $00,$FF,$FE,$55            ;      3 2
        .BYTE $00,$FE,$FE,$55            ;      3 3
        .BYTE $00,$FD,$FE,$55
        .BYTE $00,$55
; customPattern3YPosArray
        .BYTE $00,$FF,$00,$00,$55
        .BYTE $00,$01,$01,$01,$02,$55
        .BYTE $00,$02,$03,$03,$55
        .BYTE $00,$01,$00,$55
        .BYTE $00,$FF,$FE,$55
        .BYTE $00,$FF,$FF,$55
        .BYTE $00,$55
```

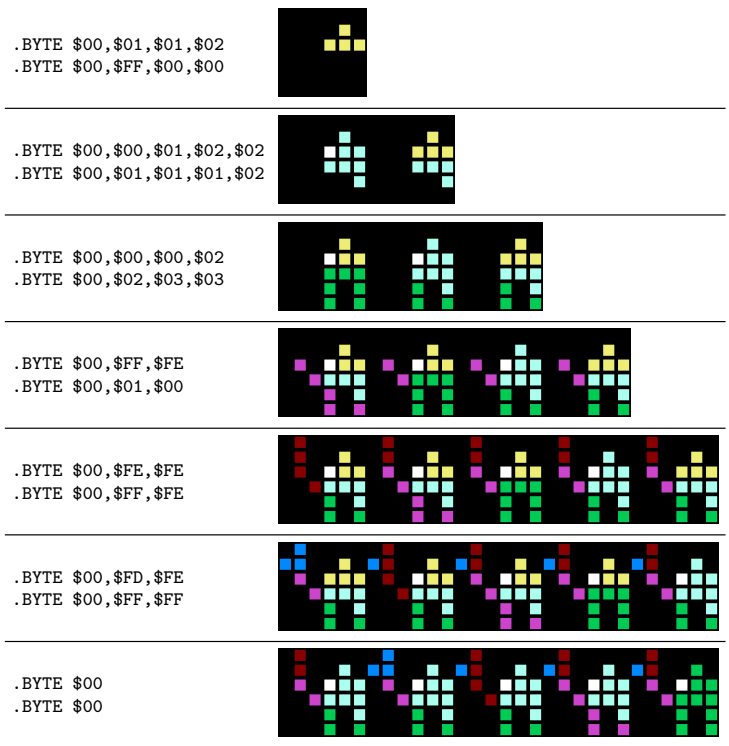


Figure 1.24: Pattern Progression for 'Custom Pattern 4'

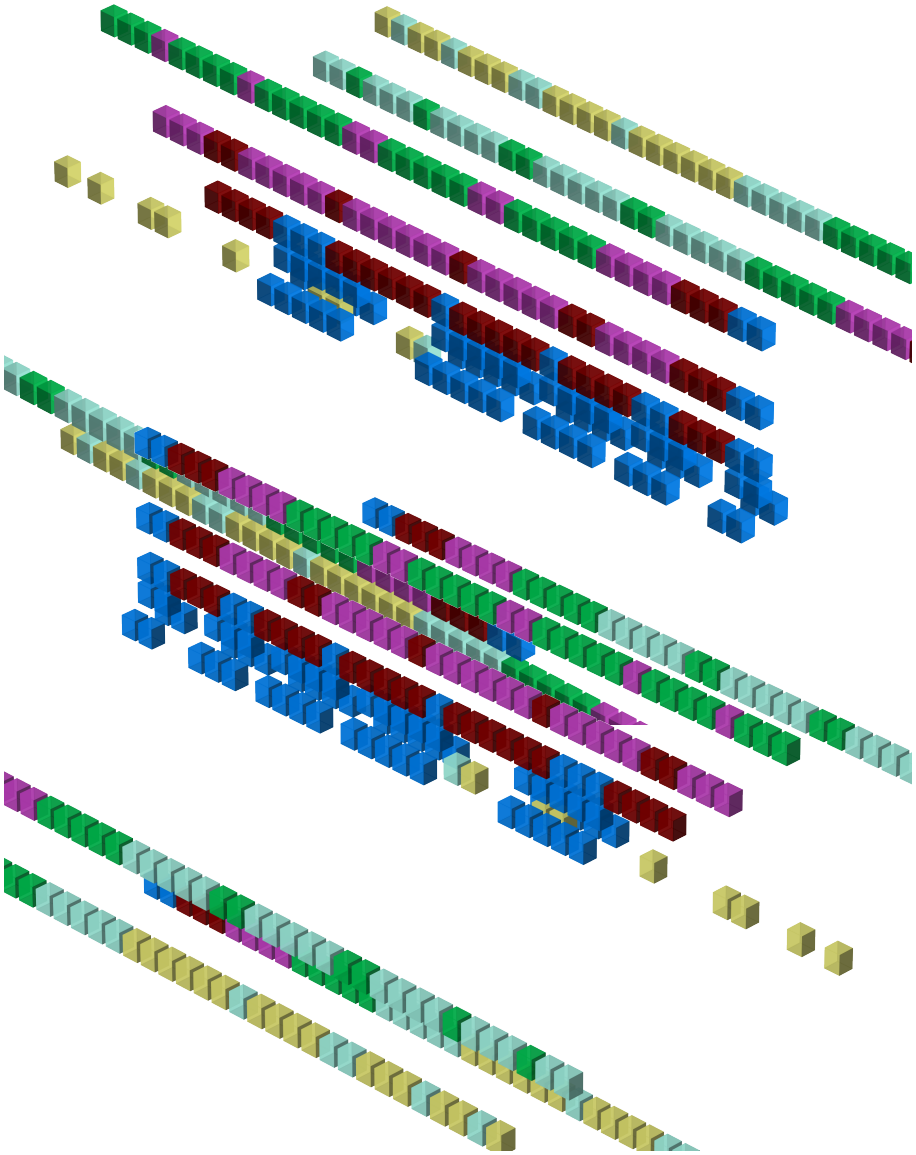


Figure 1.25: 'Custom Pattern 5'.

```
  ; customPattern4XPosArray      ;                  1
    .BYTE $00,$00,$00,$ED,$14,$55 ;
    .BYTE $00,$F2,$0F,$55         ;
    .BYTE $00,$00,$55             ;
    .BYTE $00,$00,$55             ;
    .BYTE $00,$00,$55             ;                  3
    .BYTE $00,$00,$FF,$01,$55     ;
    .BYTE $00,$55                 ;
    .BYTE $02,$55                 ;                  4
    .BYTE $00,$FC,$FD,$03,$04,$55 ;
    .BYTE $00,$55                 ;                  5
                                  ;                  6
; customPattern4YPosArray        ; 1    2       99 6106899       2    1
    .BYTE $00,$0B,$F4,$00,$00,$55 ;
    .BYTE $00,$00,$00,$55         ;
    .BYTE $00,$F9,$55             ;
    .BYTE $00,$FC,$55             ;
    .BYTE $00,$FE,$55             ;
    .BYTE $00,$FF,$00,$00,$55     ;
    .BYTE $00,$55                 ;
    .BYTE $00,$55                 ;
    .BYTE $00,$00,$00,$00,$00,$55 ;
    .BYTE $00,$55                 ;
                                  ;                  1
```

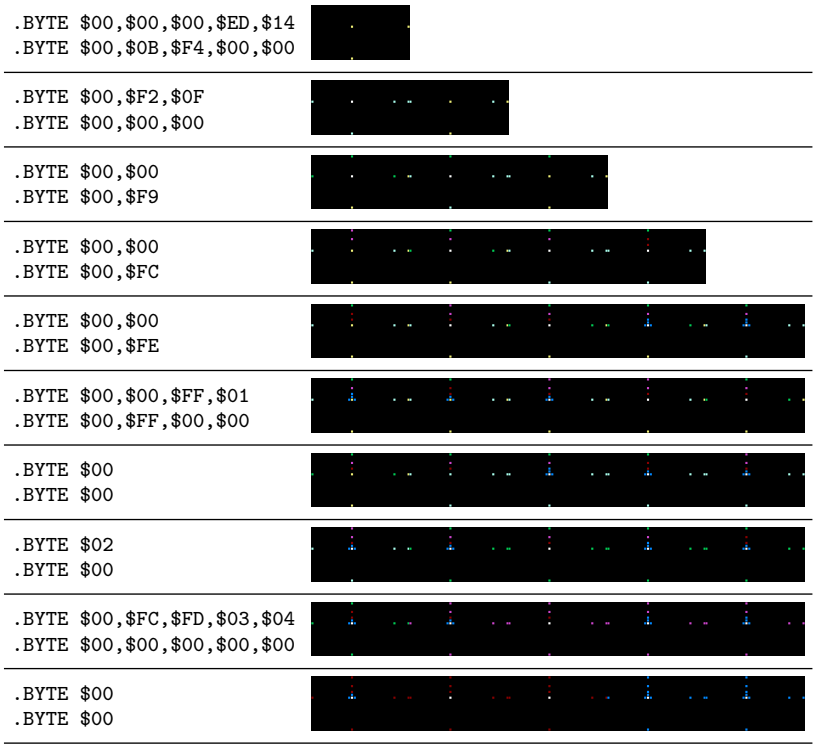


Figure 1.26: Pattern Progression for 'Custom Pattern 5'

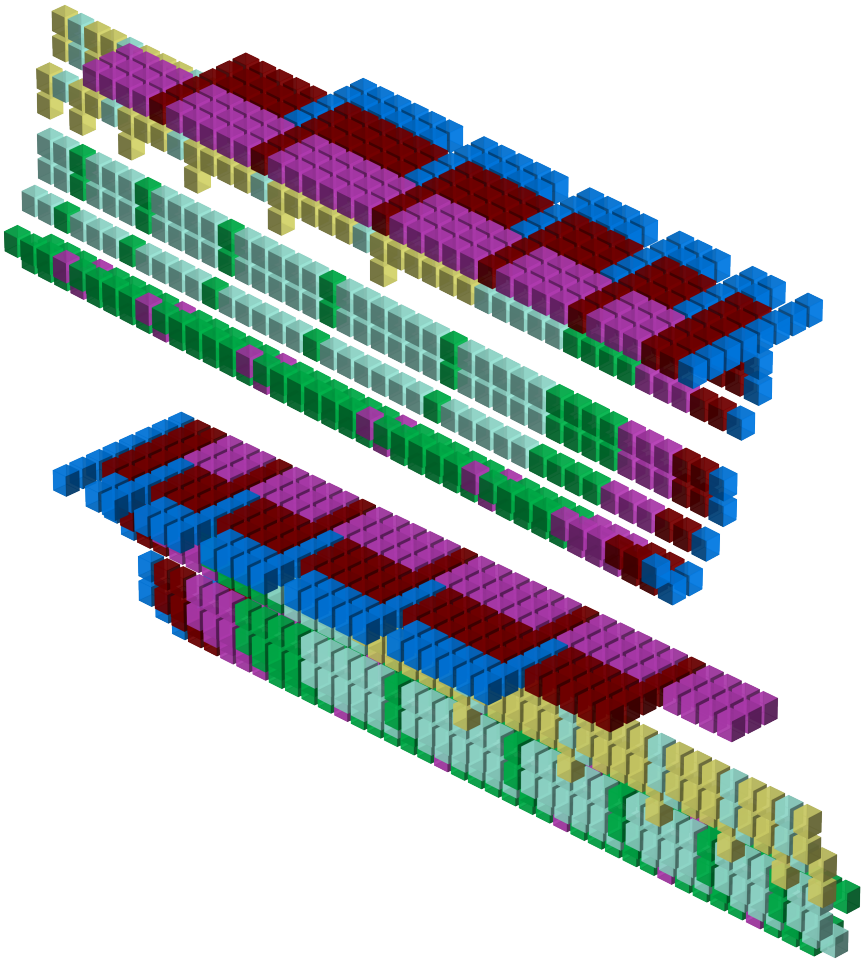


Figure 1.27: 'Custom Pattern 6'.

```
; customPattern5XPosArray              ;    44455566
        .BYTE $00,$00,$01,$01,$55      ;        1
        .BYTE $00,$FF,$FF,$FE,$55      ;        1
        .BYTE $00,$FD,$FC,$FB,$55      ;       1
        .BYTE $00,$FD,$FE,$FF,$55      ;       7
        .BYTE $00,$00,$01,$02,$55      ;      2
        .BYTE $00,$03,$04,$55          ;      2
        .BYTE $00,$55                  ; 3   2
                                       ;  33
; customPattern5YPosArray
        .BYTE $00,$FF,$FE,$FD,$55
        .BYTE $00,$01,$02,$03,$55
        .BYTE $00,$04,$04,$03,$55
        .BYTE $00,$FC,$FC,$FC,$55
        .BYTE $00,$FC,$FC,$FC,$55
        .BYTE $00,$FC,$FC,$55
        .BYTE $00,$55
```

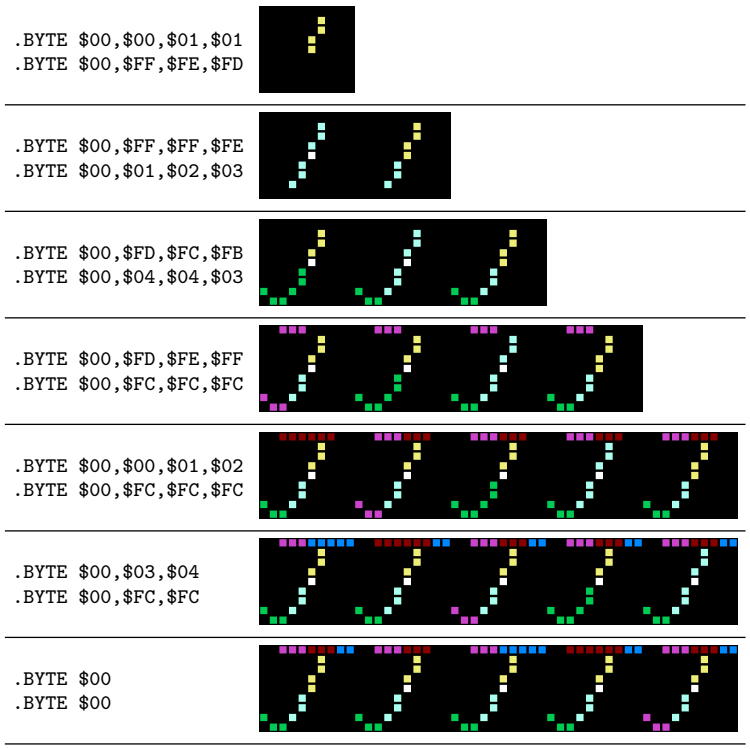


Figure 1.28: Pattern Progression for 'Custom Pattern 6'

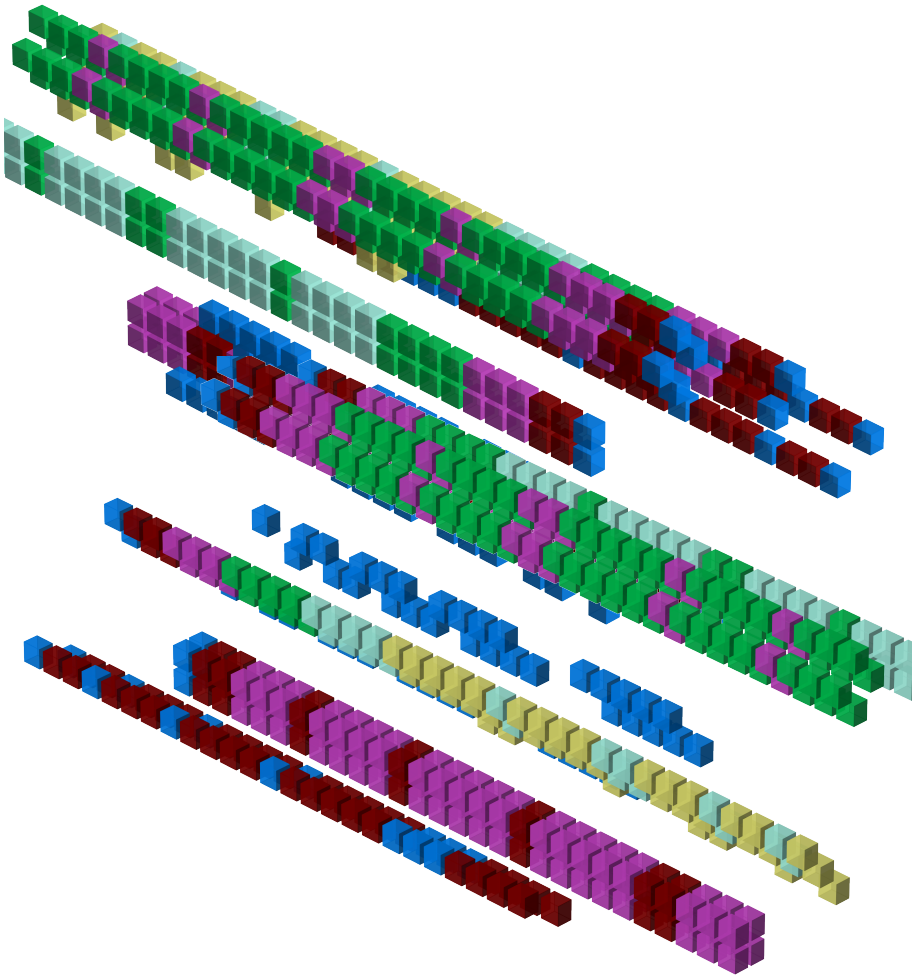


Figure 1.29: 'Custom Pattern 7'.

```
; customPattern6XPosArray              ;       3
        .BYTE $00,$01,$02,$55          ;      3 3
        .BYTE $00,$F6,$F6,$55          ; 2     3
        .BYTE $00,$FB,$FA,$FB,$FC,$55  ; 2
        .BYTE $00,$FD,$FD,$FE,$FE,$55  ;               1
        .BYTE $00,$05,$07,$55          ;              1
        .BYTE $00,$F9,$F7,$FB,$55      ;             8
        .BYTE $00,$55                  ;    6
        .BYTE $00,$55                  ;                     5
                                       ;  6    6          5
; customPattern6YPosArray              ;
        .BYTE $00,$FF,$FE,$55          ;         44
        .BYTE $00,$FC,$FD,$55          ;         44
        .BYTE $00,$FA,$FB,$FC,$FB,$55
        .BYTE $00,$05,$06,$06,$05,$55
        .BYTE $00,$03,$02,$55
        .BYTE $00,$01,$03,$03,$55
        .BYTE $00,$55
        .BYTE $00,$55
```

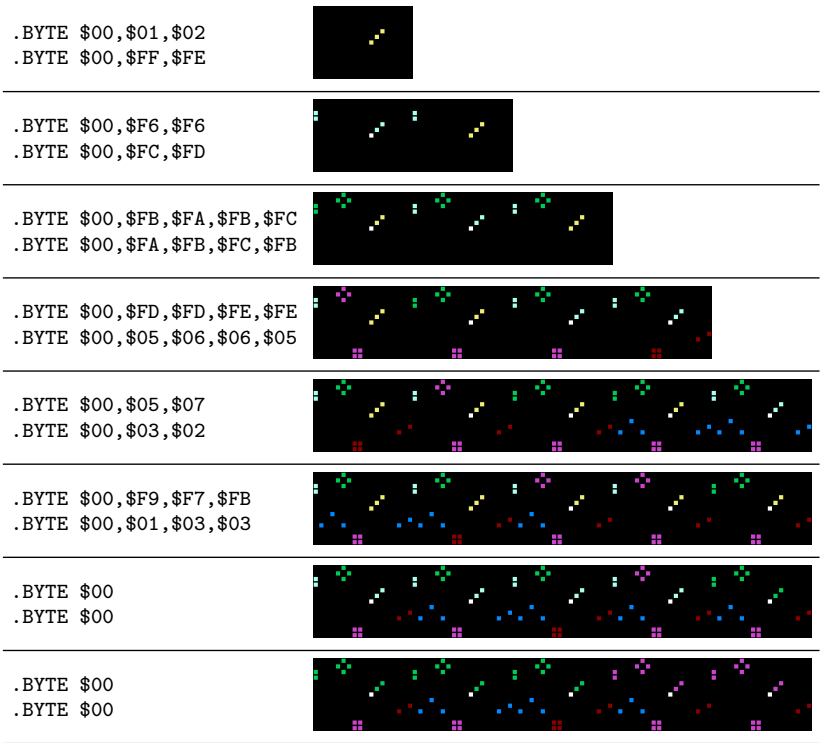
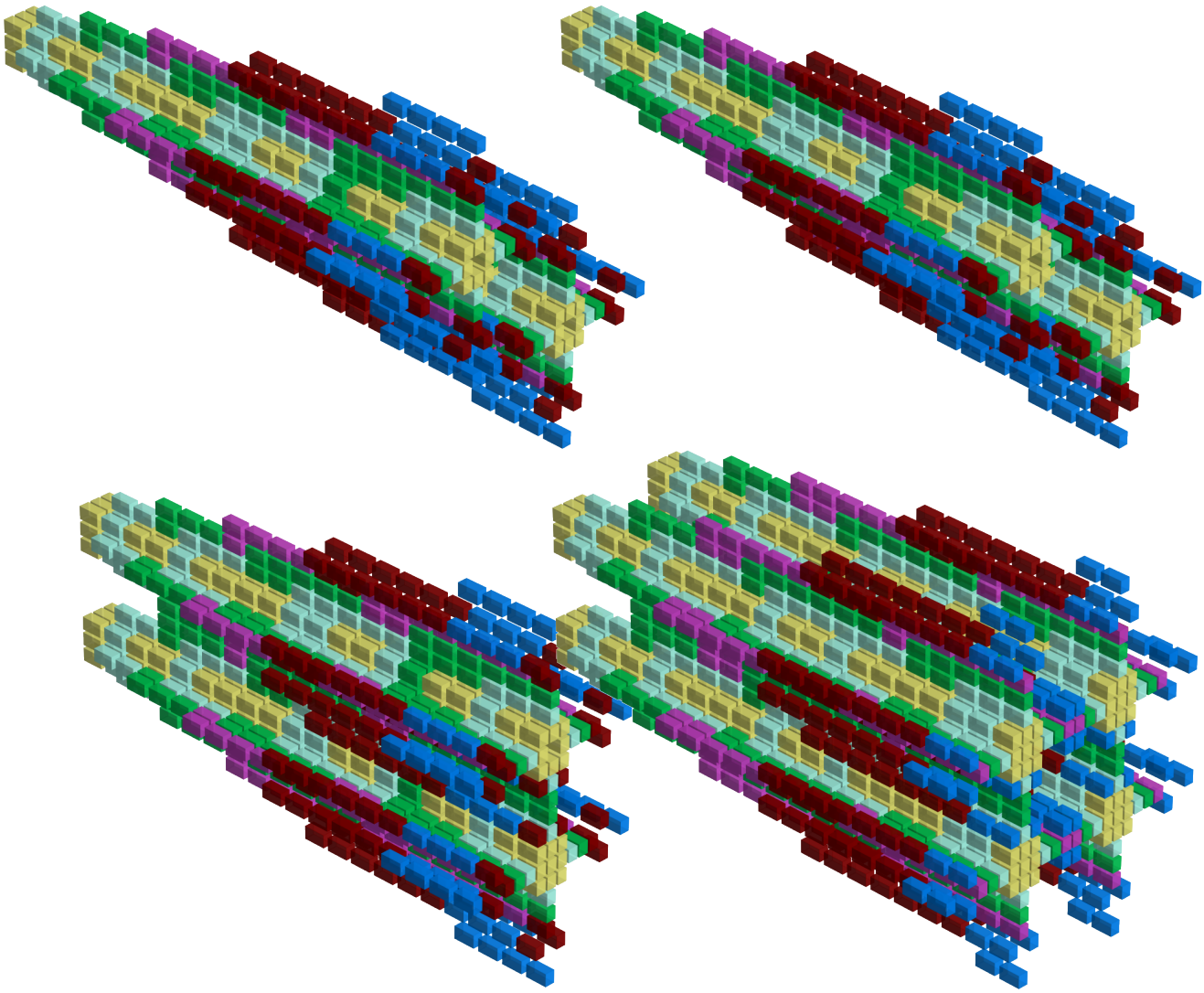
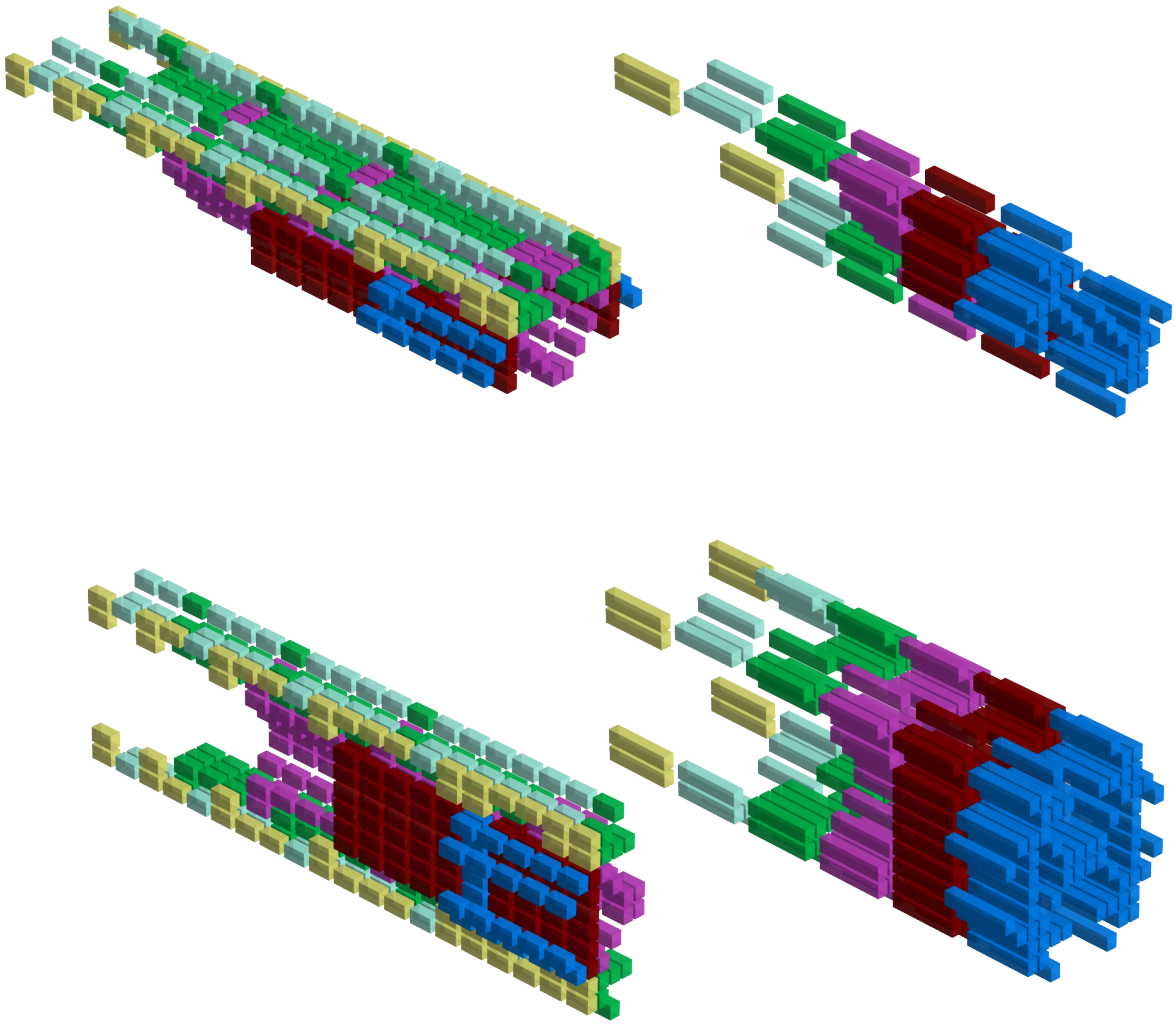


Figure 1.30: Pattern Progression for 'Custom Pattern 7'

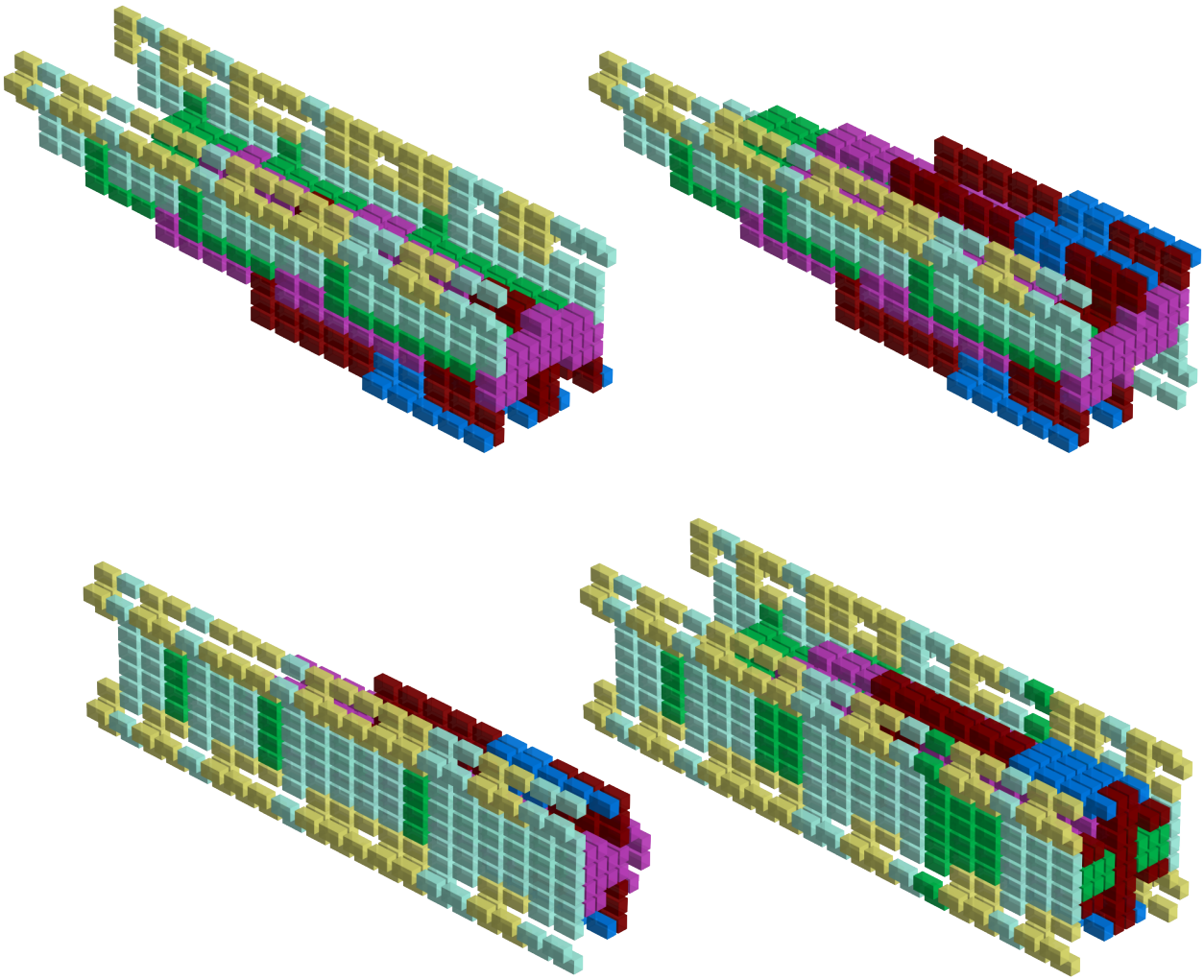
Fearful Symmetries



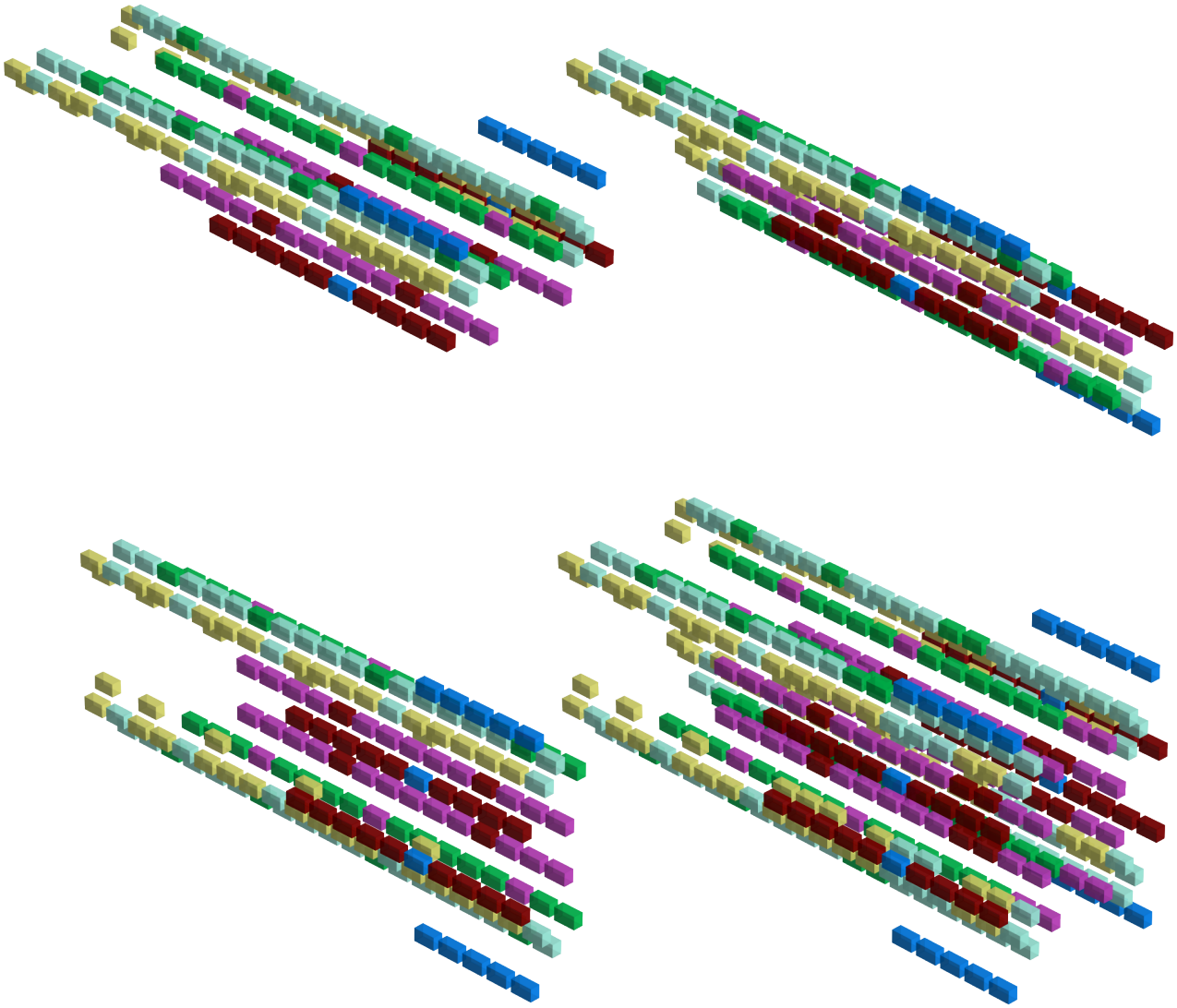
Star One: Y-Axis, X-Y Symmetry, X-Axis Symmetry, Quad Symmetry



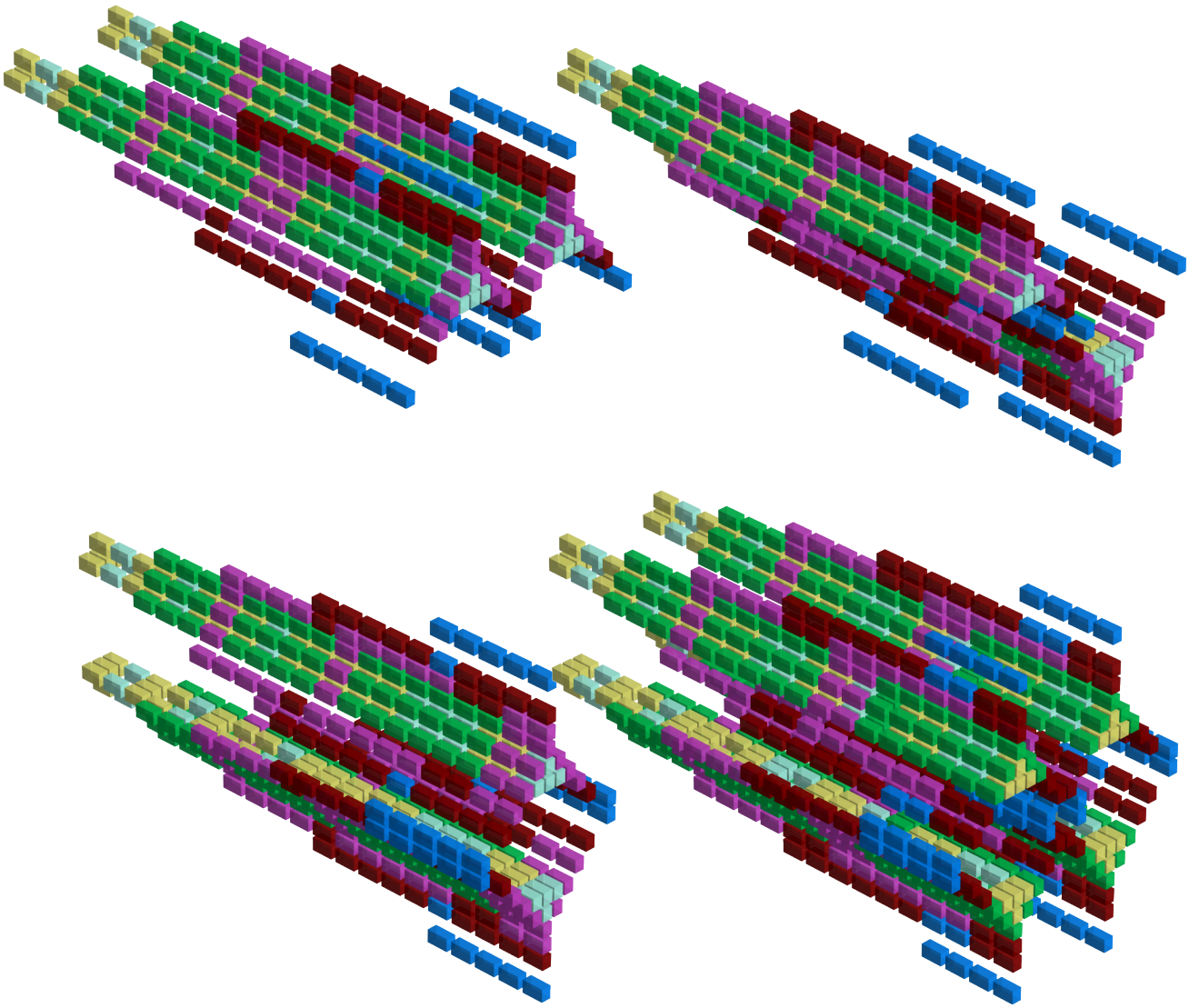
The Twist: Y-Axis, X-Y Symmetry, X-Axis Symmetry, Quad Symmetry



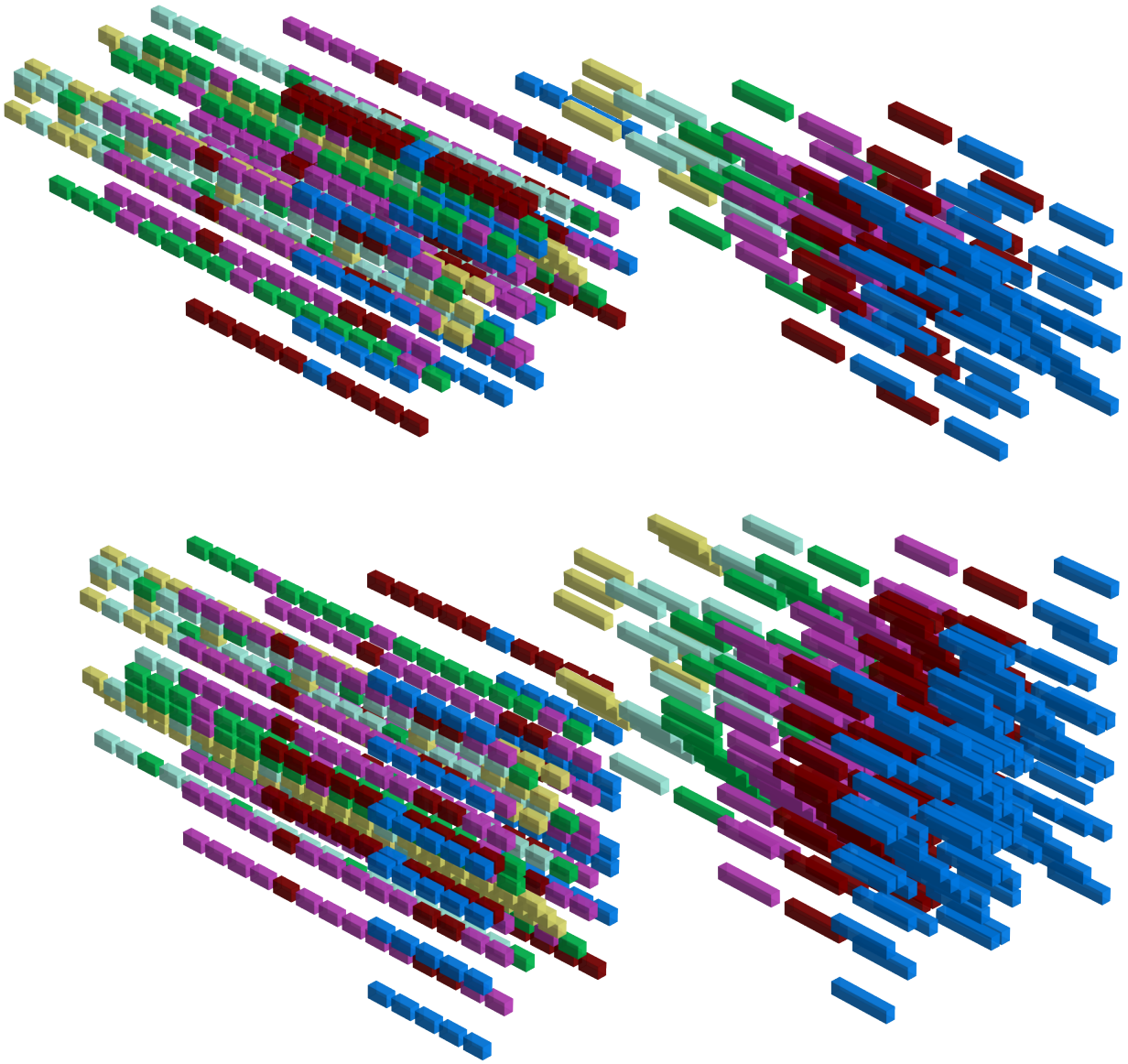
La Llamita: Y-Axis, X-Y Symmetry, X-Axis Symmetry, Quad Symmetry



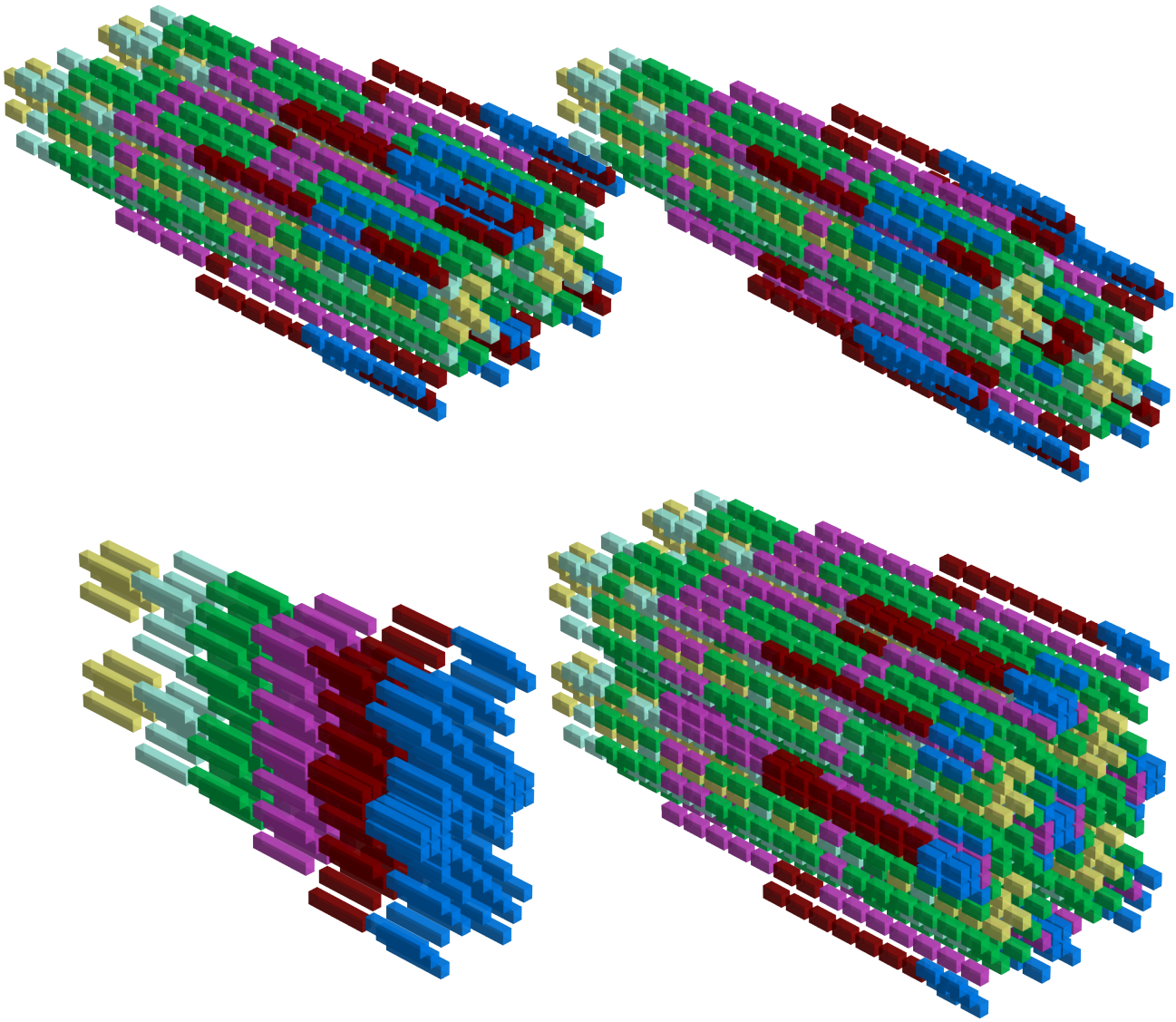
Star Two: Y-Axis, X-Y Symmetry, X-Axis Symmetry, Quad Symmetry



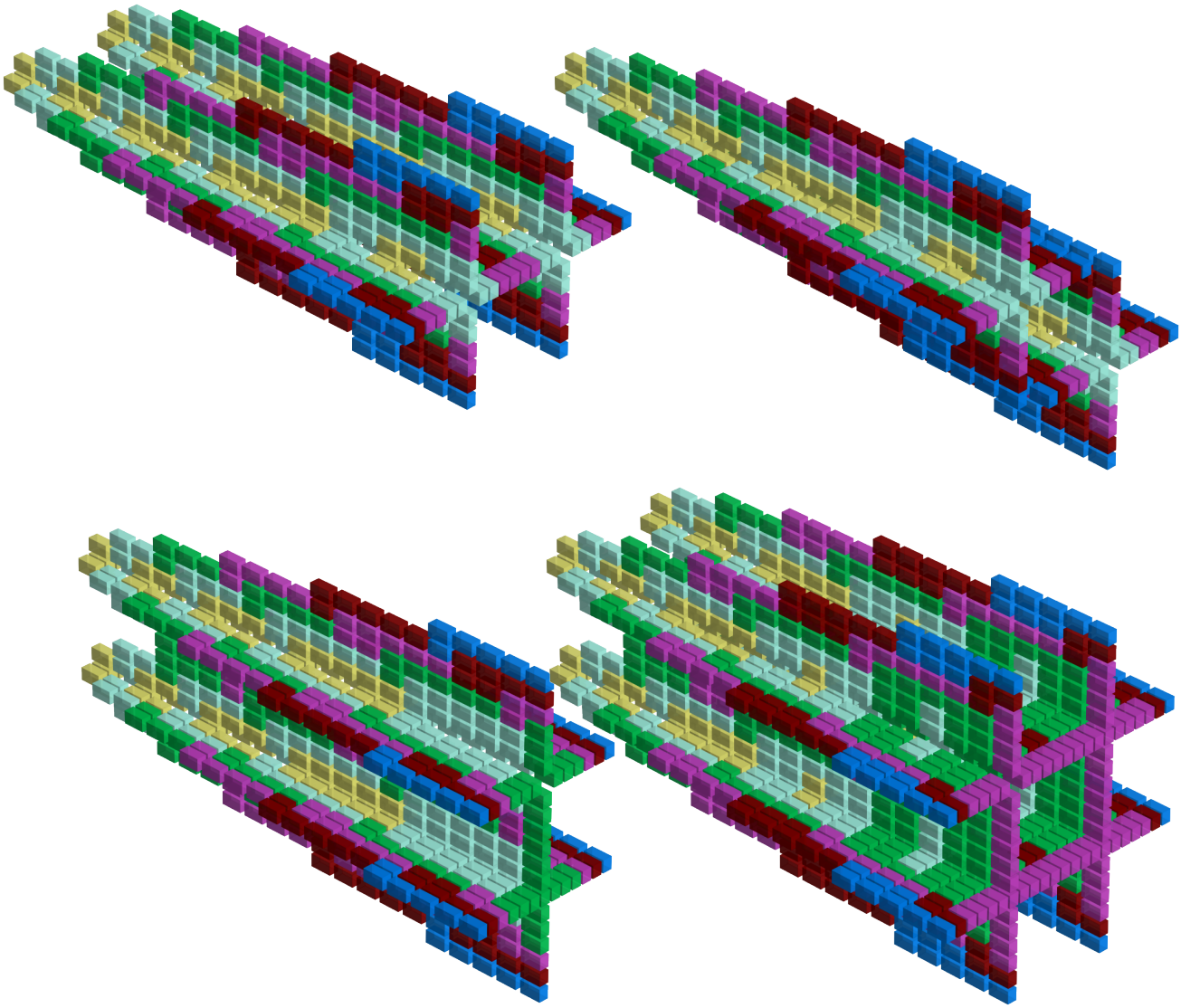
Deltoid: Y-Axis, X-Y Symmetry, X-Axis Symmetry, Quad Symmetry



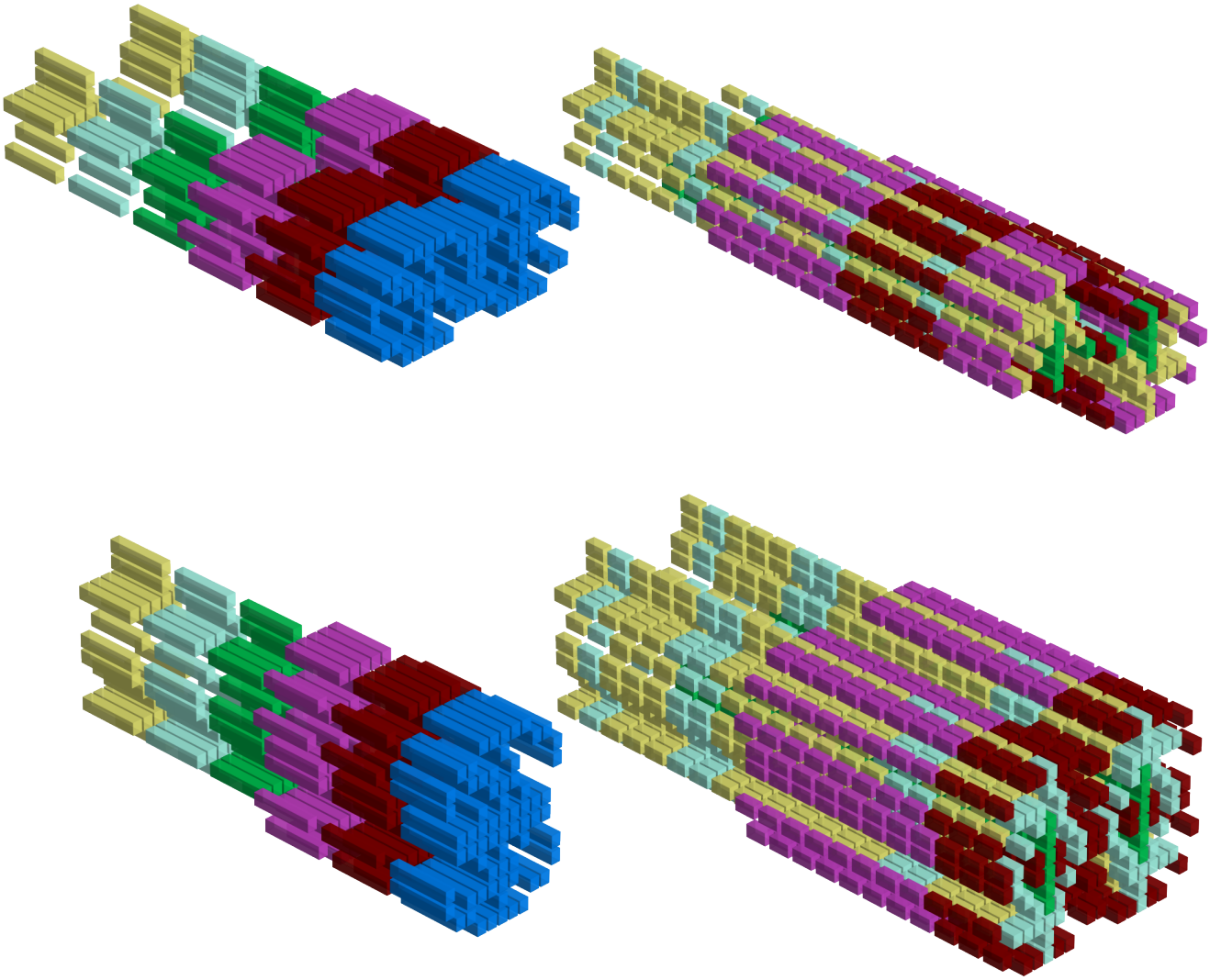
Diffused: Y-Axis, X-Y Symmetry, X-Axis Symmetry, Quad Symmetry



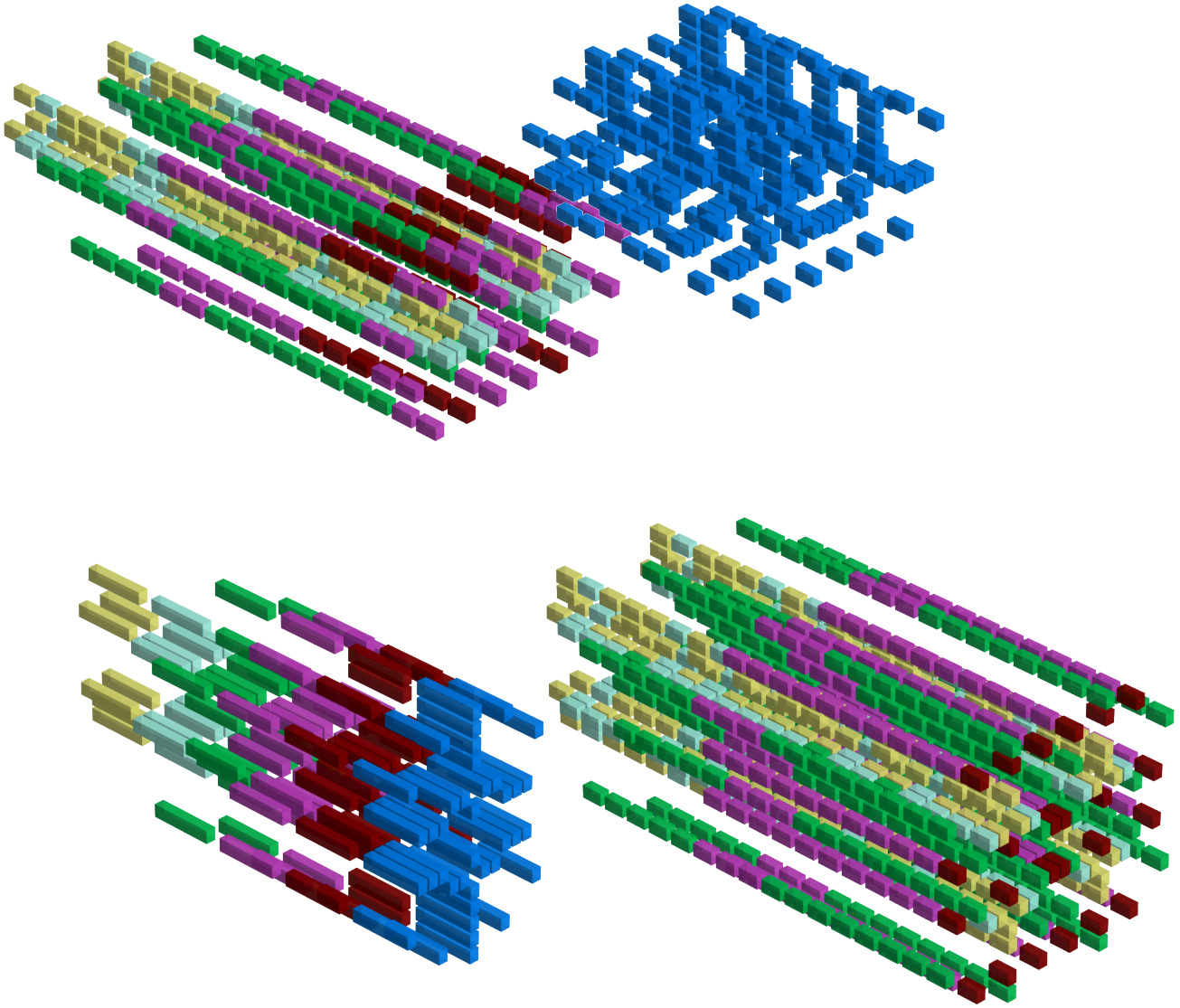
Multi-Cross: Y-Axis, X-Y Symmetry, X-Axis Symmetry, Quad Symmetry



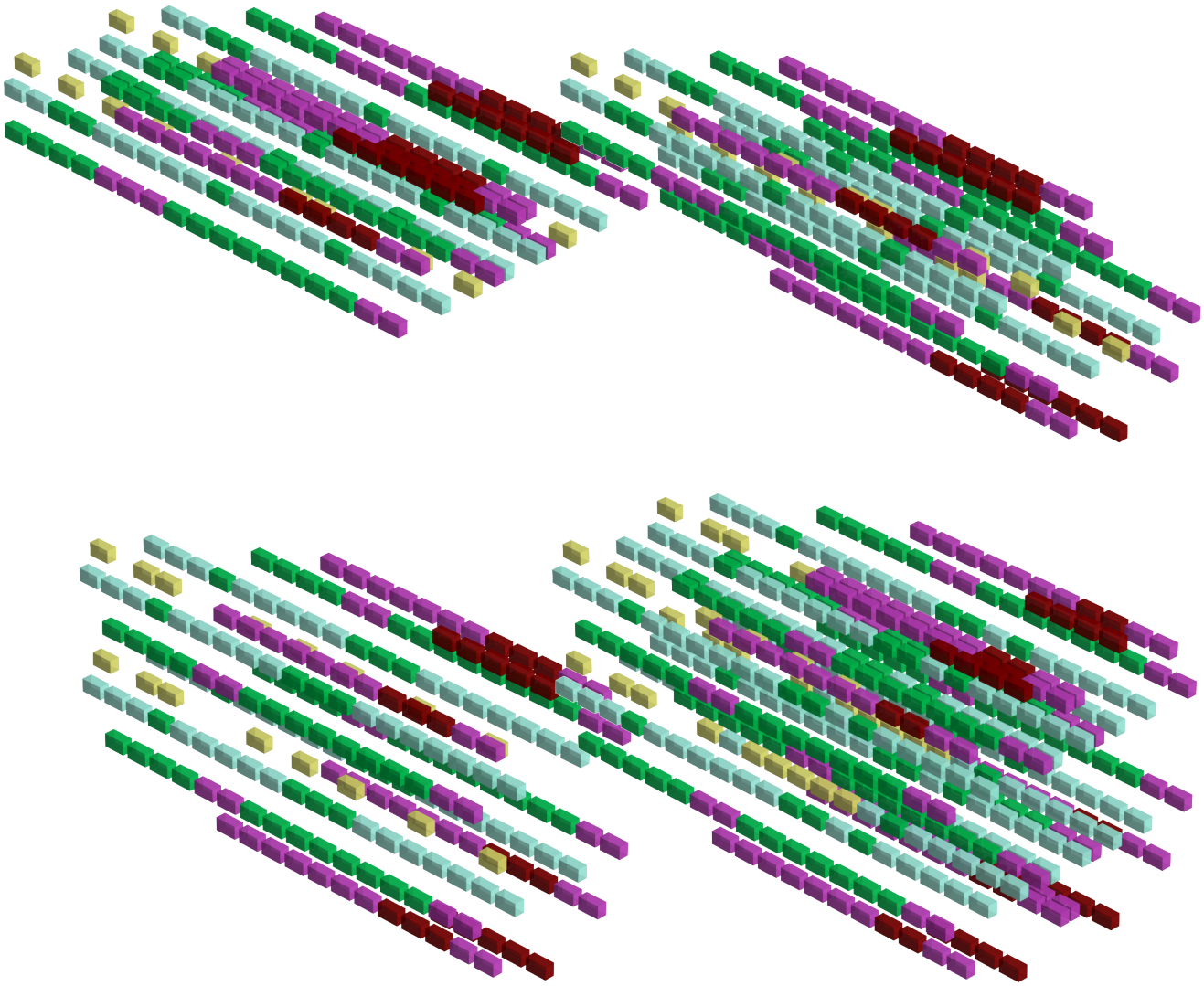
Pulsar: Y-Axis, X-Y Symmetry, X-Axis Symmetry, Quad Symmetry



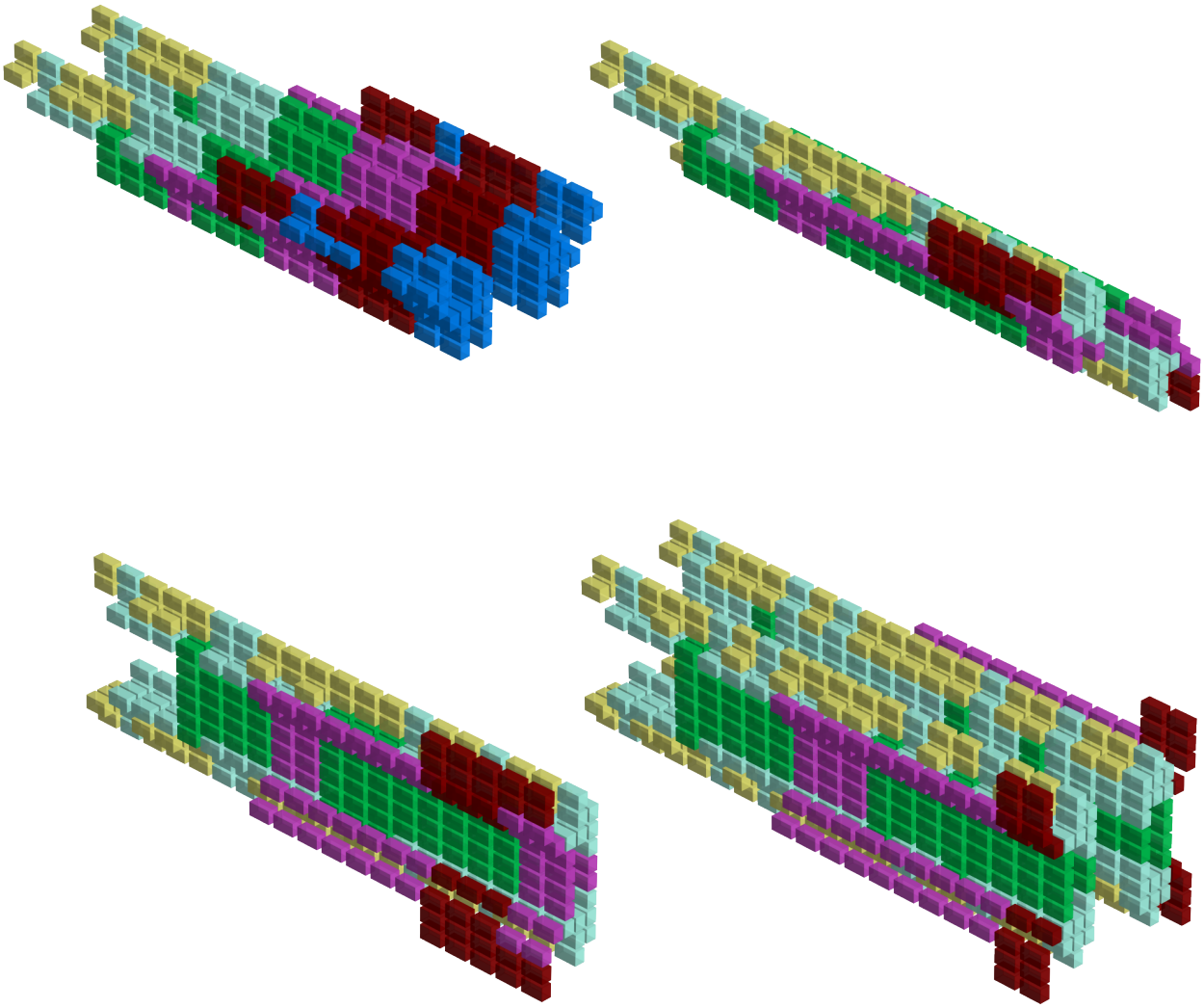
Custom Pattern 1: Y-Axis, X-Y Symmetry, X-Axis Symmetry, Quad Symmetry



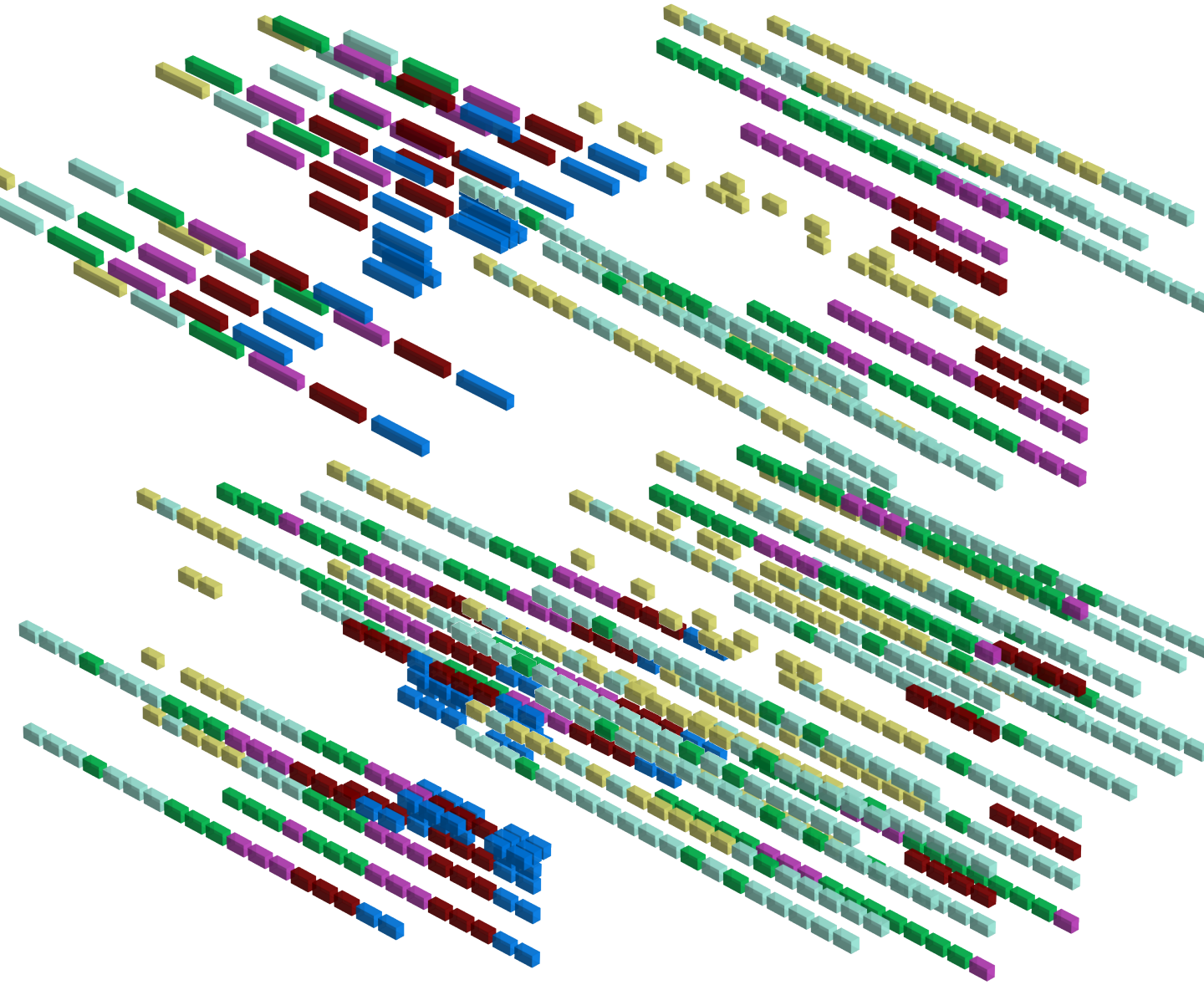
Custom Pattern 2: Y-Axis, X-Y Symmetry, X-Axis Symmetry, Quad Symmetry



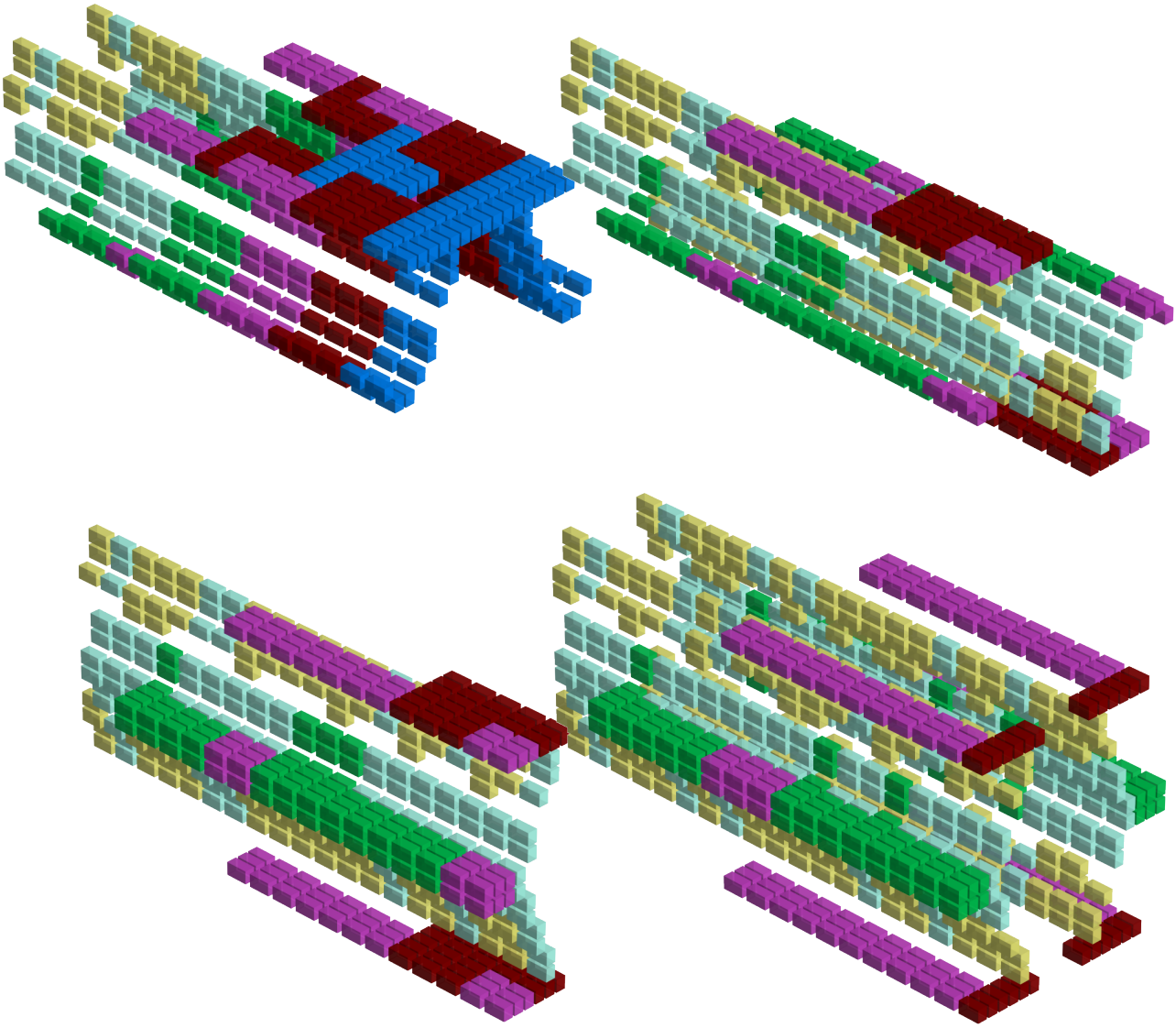
Custom Pattern 3: Y-Axis, X-Y Symmetry, X-Axis Symmetry, Quad Symmetry



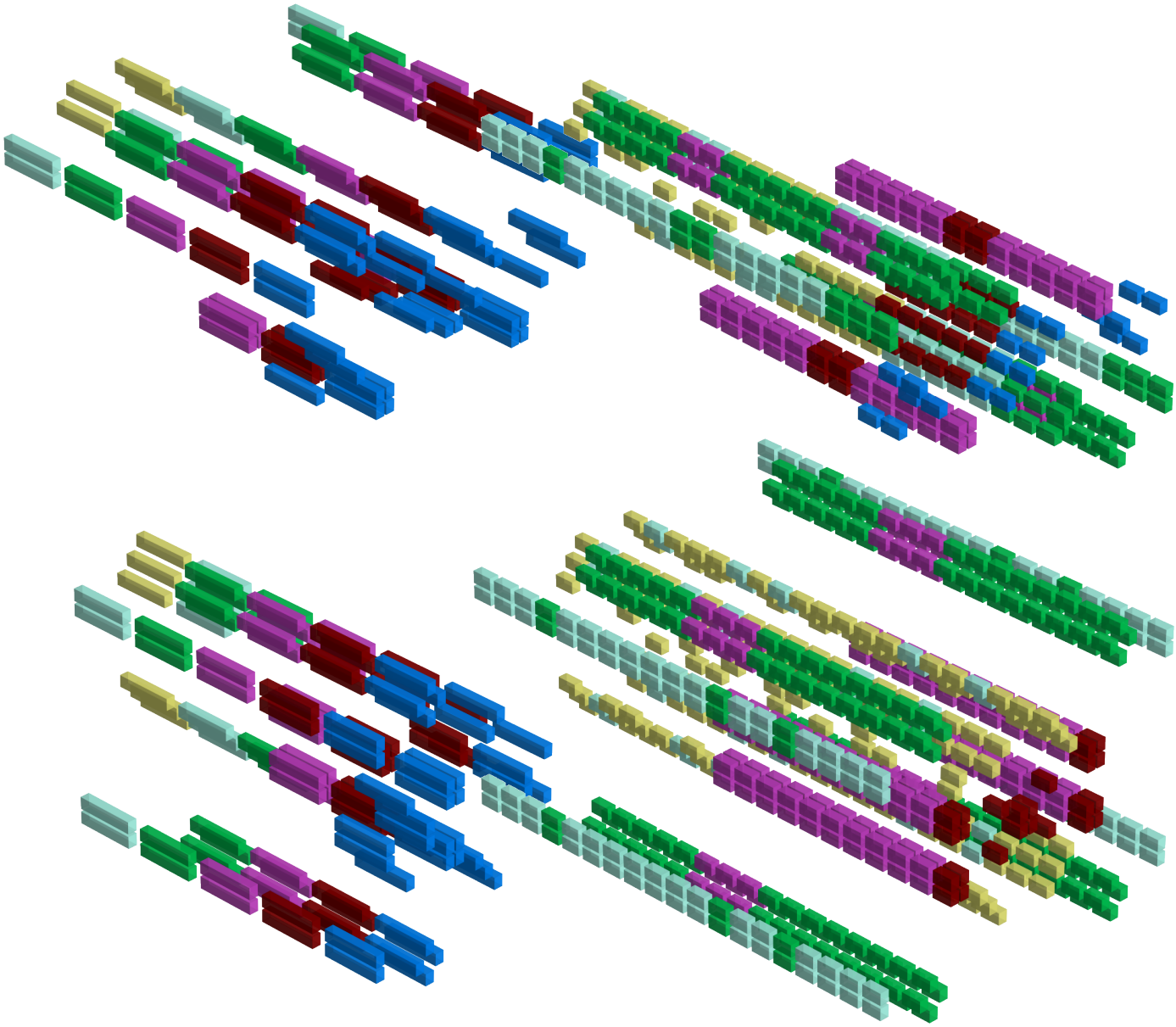
Custom Pattern 4: Y-Axis, X-Y Symmetry, X-Axis Symmetry, Quad Symmetry



Custom Pattern 5: Y-Axis, X-Y Symmetry, X-Axis Symmetry, Quad Symmetry



Custom Pattern 6: Y-Axis, X-Y Symmetry, X-Axis Symmetry, Quad Symmetry



Custom Pattern 7: Y-Axis, X-Y Symmetry, X-Axis Symmetry, Quad Symmetry

Bursts

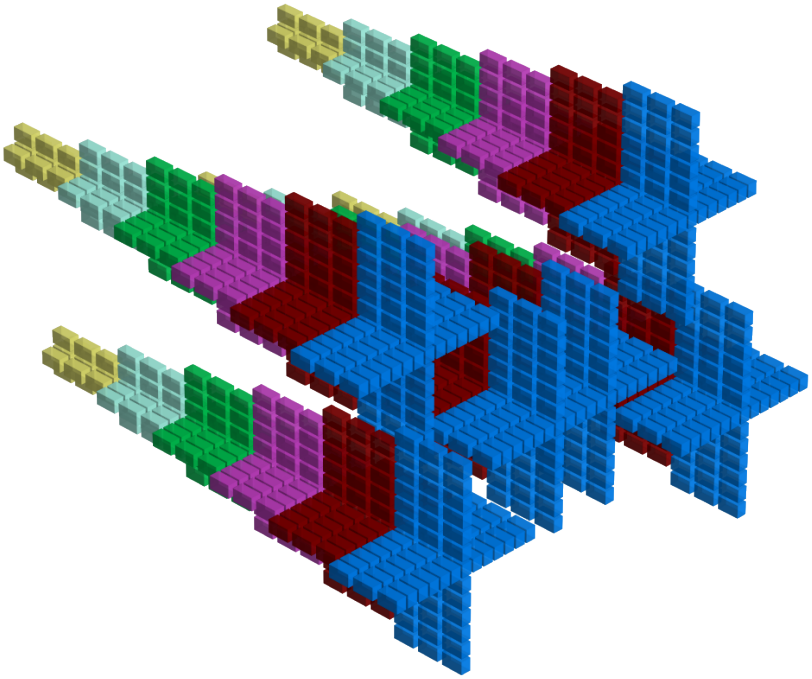


Figure 3.1: Evolution of the default burst at F1.

```
; burstGeneratorF1
        ; currentSymmetrySetting: 'Current symmetry setting.'
        ; Possible values are 0 - 4:
        ; 'NO SYMMETRY     '
        ; 'Y-AXIS SYMMETRY '
        ; 'X-Y SYMMETRY    '
        ; 'X-AXIS SYMMETRY '
        ; 'QUAD SYMMETRY   '
        .BYTE $01
        ; smoothingDelay: 'Because of the time taken to draw larger patterns speed
        ; increase-decrease is not linear. You can adjust the compensating delay
        ; which often smooths out jerky patterns. Can be used just for special FX)
        ; though. Suck it and see.'
        .BYTE $0C

        ; Burst Position 1
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $07,$06
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $07

        ; Burst Position 2
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $11,$0D
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $07

        ; Burst Position 3
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $06,$11
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $07

        ; Burst Position 4
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $FF,$0B
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $07

        ; Burst Position 5
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $FF,$00
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $FF

        ; Burst Position 6
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $21,$06
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $00

        ; Burst Position 7
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $06,$01
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $06

        ; Burst Position 8
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $41,$FF
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $00

        ; Burst Position 9
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $06,$01
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $06

        ; Burst Position 10
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $01,$06
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $00
```

Listing 3.1: Source code for the F1 Burst.

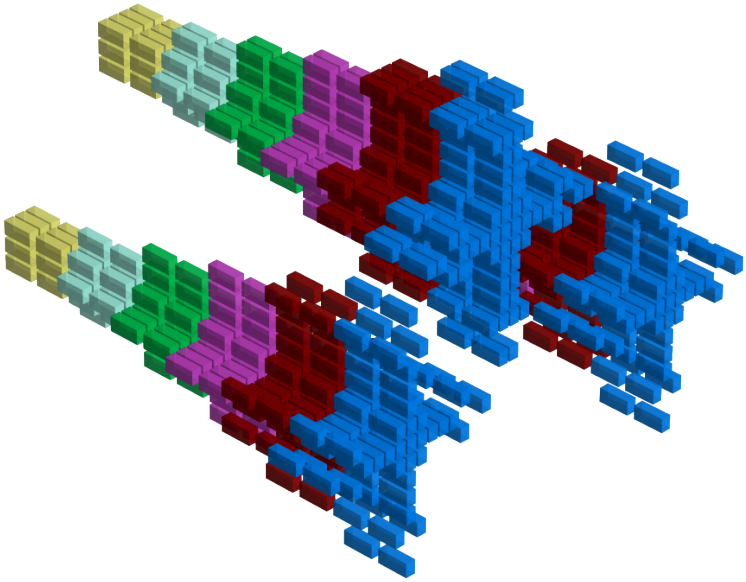


Figure 3.2: Evolution of the default burst at F2.

```
; burstGeneratorF2
        ; currentSymmetrySetting: 'Current symmetry setting.'
        ; Possible values are 0 - 4:
        ; 'NO SYMMETRY     '
        ; 'Y-AXIS SYMMETRY '
        ; 'X-Y SYMMETRY    '
        ; 'X-AXIS SYMMETRY '
        ; 'QUAD SYMMETRY   '
        .BYTE $01
        ; smoothingDelay: 'Because of the time taken to draw larger patterns speed
        ; increase/decrease is not linear. You can adjust the compensating delay
        ; which often smooths out jerky patterns. Can be used just for special FX)
        ; though. Suck it and see.'
        .BYTE $0C

        ; Burst Position 1
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $13,$08
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $00

        ; Burst Position 2
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $07,$0F
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $00

        ; Burst Position 3
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $FF,$00
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $06

        ; Burst Position 4
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $01,$2A
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $41

        ; Burst Position 5
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $02,$00
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $04

        ; Burst Position 6
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $62,$FF
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $41

        ; Burst Position 7
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $06,$40
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $00

        ; Burst Position 8
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $6B,$04
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $41

        ; Burst Position 9
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $FF,$00
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $FF

        ; Burst Position 10
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $00,$FF
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $00
```

Listing 3.2: Source code for the F2 Burst.

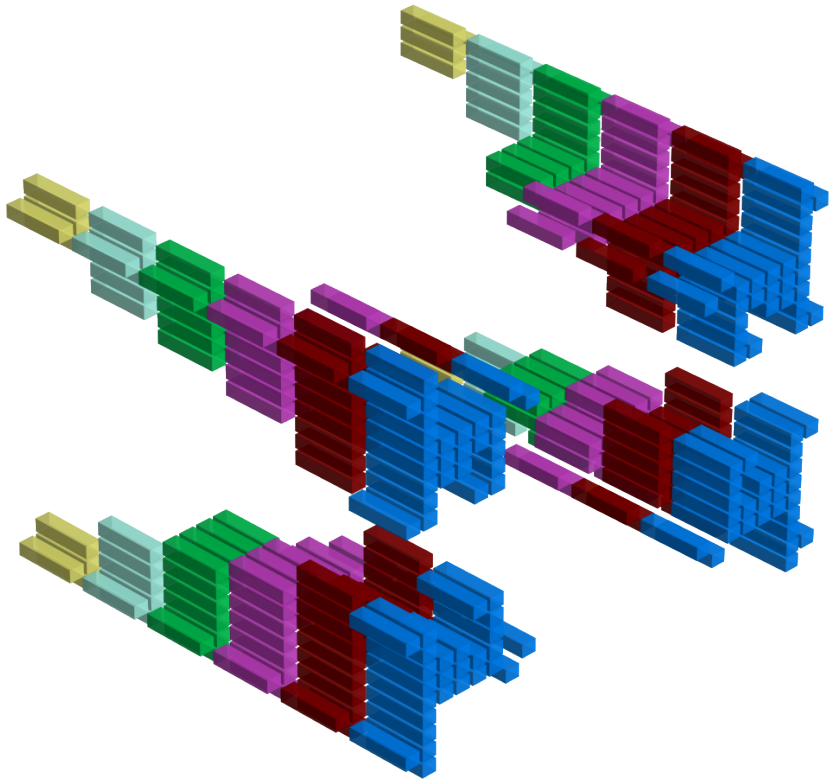


Figure 3.3: Evolution of the default burst at F3.

```
; burstGeneratorF3
        ; currentSymmetrySetting: 'Current symmetry setting.'
        ; Possible values are 0 - 4:
        ; 'NO SYMMETRY      '
        ; 'Y-AXIS SYMMETRY  '
        ; 'X-Y SYMMETRY     '
        ; 'X-AXIS SYMMETRY  '
        ; 'QUAD SYMMETRY    '
        .BYTE $04
        ; smoothingDelay: 'Because of the time taken to draw larger patterns speed
        ; increase/decrease is not linear. You can adjust the compensating delay
        ; which often smooths out jerky patterns. Can be used just for special FX)
        ; though. Suck it and see.'
        .BYTE $01

        ; Burst Position 1
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $08,$01
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $02

        ; Burst Position 2
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $FF,$01
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $02

        ; Burst Position 3
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $08,$01
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $02

        ; Burst Position 4
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $08,$01
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $02

        ; Burst Position 5
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $08,$01
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $02

        ; Burst Position 6
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $08,$01
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $02

        ; Burst Position 7
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $08,$01
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $02

        ; Burst Position 8
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $FF,$03
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $02

        ; Burst Position 9
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $08,$03
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $02

        ; Burst Position 10
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $08,$03
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $02
```

Listing 3.3: Source code for the F3 Burst.

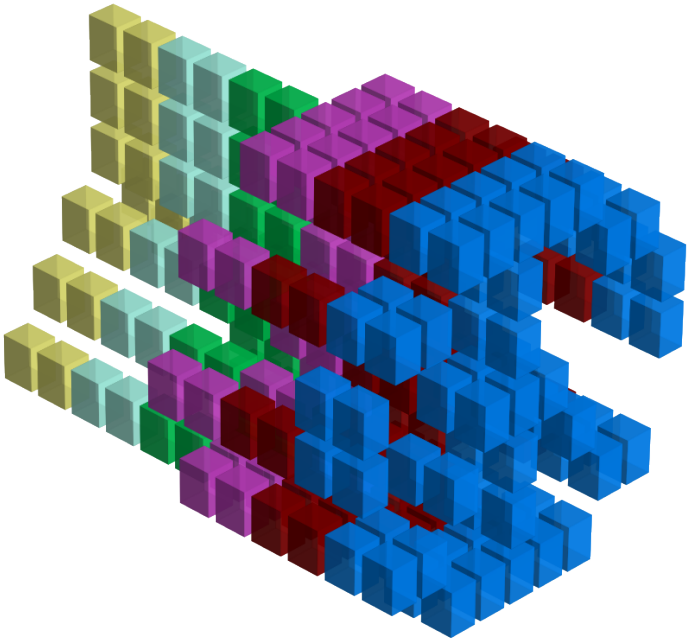


Figure 3.4: Evolution of the default burst at F3.

```
; burstGeneratorF4
        ; currentSymmetrySetting: 'Current symmetry setting.'
        ; Possible values are 0 - 4:
        ; 'NO SYMMETRY     '
        ; 'Y-AXIS SYMMETRY '
        ; 'X-Y SYMMETRY    '
        ; 'X-AXIS SYMMETRY '
        ; 'QUAD SYMMETRY   '
        .BYTE $00
        ; smoothingDelay: 'Because of the time taken to draw larger patterns speed
        ; increase/decrease is not linear. You can adjust the compensating delay
        ; which often smooths out jerky patterns. Can be used just for special FX)
        ; though. Suck it and see.'
        .BYTE $11

        ; Burst Position 1
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $12,$09
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $08

        ; Burst Position 2
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $12,$09
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $08

        ; Burst Position 3
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $FF,$08
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $03

        ; Burst Position 4
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $02,$08
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $03

        ; Burst Position 5
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $02,$08
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $03

        ; Burst Position 6
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $02,$FF
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $00

        ; Burst Position 7
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $00,$00
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $00

        ; Burst Position 8
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $01,$24
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $00

        ; Burst Position 9
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $05,$01
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $00

        ; Burst Position 10
        ; X/Y Co-ordinates: X/Y Position relative to cursor to place the burst.
        .BYTE $00,$00
        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
        .BYTE $00
```

Listing 3.4: Source code for the F3 Burst.

Particular Presets

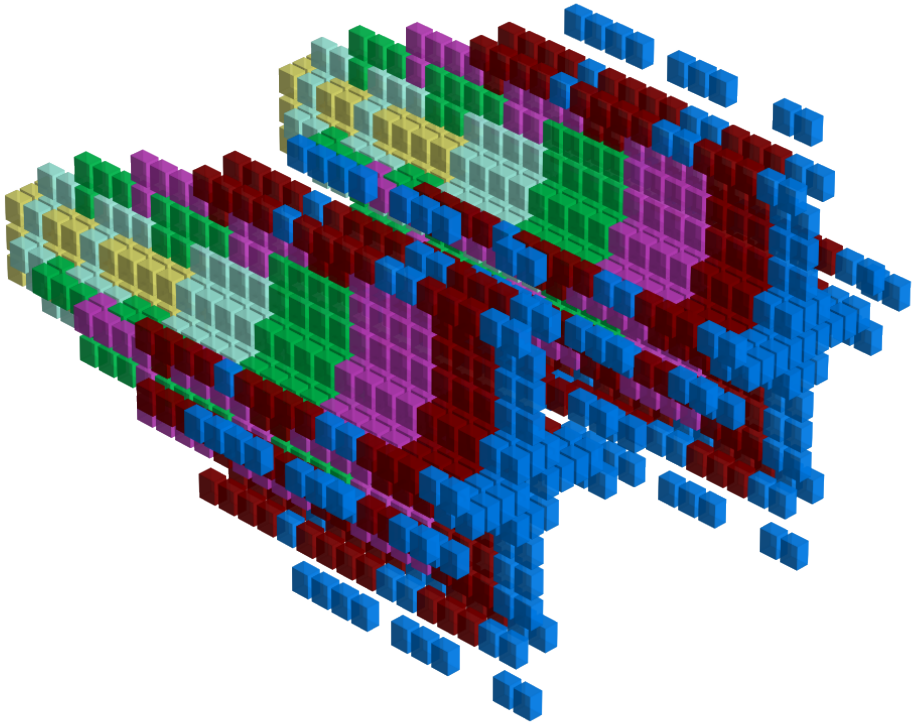


Figure 4.1: Evolution of Preset 0.

```
preset0
        ; unusedPresetByte: Unused Byte
        .BYTE $00
        ; smoothingDelay: 'Because of the time taken to draw larger patterns speed
        ; increase/decrease is not linear. You can adjust the 'compensating delay'
        ; which often smooths out jerky patterns. Can be used just for special FX),
        ; though. Suck it and see.'
        .BYTE $0C
        ; cursorSpeed: 'Gives you a slow or fast little cursor, according to setting.'
        .BYTE $02
        ; bufferLength: 'Larger patterns flow more smoothly with a shorter
        ; Buffer Length - not so many positions are retained so less plotting to do.
        ; Small patterns with a long Buffer Length are good for 'steamer' effects.
        ; N.B. Cannot be adjusted whilst patterns are actually onscreen.'
        .BYTE $1F
        ; pulseSpeed: 'Usually if you hold down the button you get a continuous
        ; stream. Setting the Pulse Speed allows you to generate a pulsed stream, as
        ; if you were rapidly pressing and releasing the FIRE button.'
        .BYTE $01
        ; indexForColorBarDisplay: 'The initial index for the color displayed
        ; in the color bar when adjusting the colors for each step.'
        .BYTE $01
        ; lineWidth: 'Sets the width of the lines produced in Line Mode.'
        .BYTE $07
        ; sequencerSpeed: 'Controls the rate at which sequencer feeds in its data. '
        .BYTE $04
        ; pulseWidth: 'Sets the length of the pulses in a pulsed stream output.
        ; Don't worry about what that means - just get in there and mess with it.'
        .BYTE $01
        ; baseLevel: 'Controls how many 'levels' of pattern are plotted.'
        .BYTE $07
        ; presetColorValuesArray: 'Allows you to set the colour for each of the
        ; seven pattern steps. Set up the colour you want, press RETURN, and the
        ; command offers the next colour along, up to no. 7, then ends. Cannot be
        ; adjusted while patterns being generated.'
        .BYTE BLACK,BLUE,RED,PURPLE,GREEN,CYAN,YELLOW,WHITE
        ; trackingActivated: 'Controls whether logic-seeking is used in the
        ; buffer or not. The upshot of this for you is a slightly different feel -
        ; continuous but fragmented when ON, or together-ish bursts when OFF. Try it.'
        .BYTE $00
        ; lineModeActivated: 'A bit like drawing with the Aurora Borealis'
        .BYTE $00
        ; presetIndex: 'This calls in one of the 16 presets, stored Lightsynth
        ; parameters which give different effects. Try them all out io see some uf
        ; the multitude of effects which you cai achieve using the system. Some are
        ; fast, some slow, some pulse, others swirl. Play with them all, try them to
        ; different music.'
        .BYTE $00
        ; currentPatternElement: 'Initial pattern used by this preset.'
        .BYTE $00
        ; currentSymmetrySetting: 'Current symmetry setting.'
        ; Possible values are 0 - 4:
        ; 'NO SYMMETRY     '
        ; 'Y-AXIS SYMMETRY '
        ; 'X-Y SYMMETRY    '
        ; 'X-AXIS SYMMETRY '
        ; 'QUAD SYMMETRY   '
        .BYTE $01
        ; Unused Data.
        .BYTE $FF,$00,$FF,$FF,$00,$FF,$00,$FF,$00
```

Listing 4.1: Source code for Preset 0.

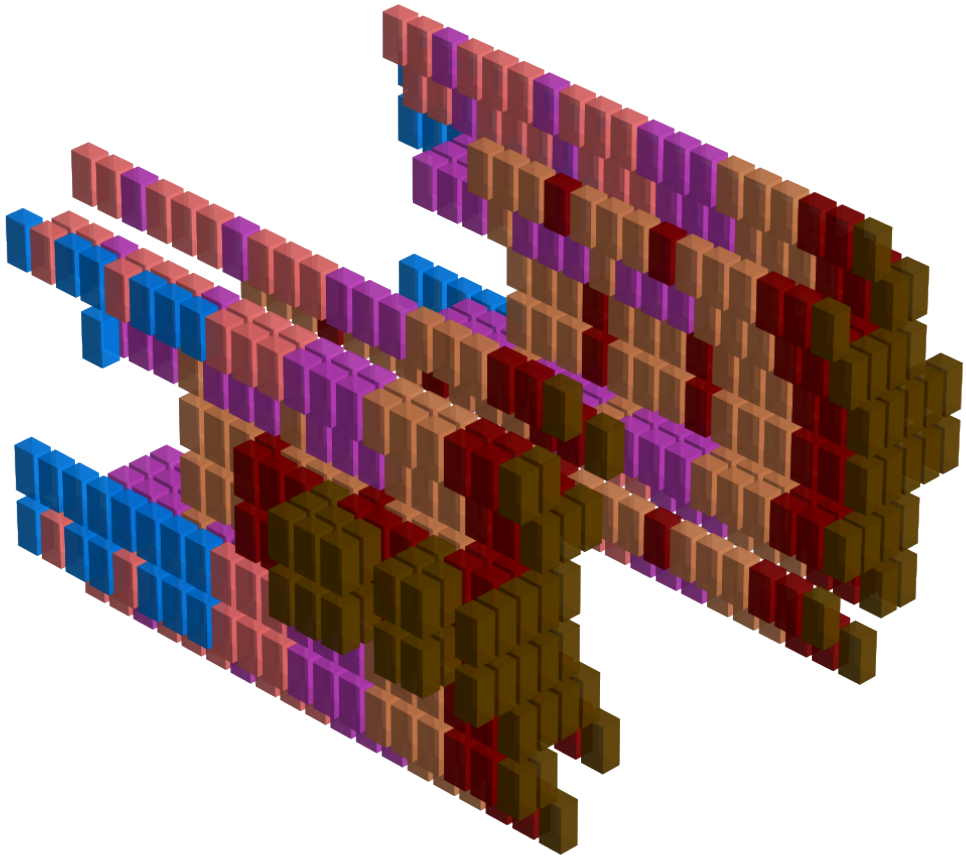


Figure 4.2: Evolution of Preset 1.

```
preset1
        ; unusedPresetByte: Unused Byte
        .BYTE $00
        ; smoothingDelay: 'Because of the time taken to draw larger patterns speed
        ; increase/decrease is not linear. You can adjust the 'compensating delay'
        ; which often smooths out jerky patterns. Can be used just for special FX),
        ; though. Suck it and see.'
        .BYTE $0C
        ; cursorSpeed: 'Gives you a slow or fast little cursor, according to setting.'
        .BYTE $02
        ; bufferLength: 'Larger patterns flow more smoothly with a shorter
        ; Buffer Length - not so many positions are retained so less plotting to do.
        ; Small patterns with a long Buffer Length are good for 'steamer' effects.
        ; N.B. Cannot be adjusted whilst patterns are actually onscreen.'
        .BYTE $28
        ; pulseSpeed: 'Usually if you hold down the button you get a continuous
        ; stream. Setting the Pulse Speed allows you to generate a pulsed stream, as
        ; if you were rapidly pressing and releasing the FIRE button.'
        .BYTE $01
        ; indexForColorBarDisplay: 'The initial index for the color displayed
        ; in the color bar when adjusting the colors for each step.'
        .BYTE $0E
        ; lineWidth: 'Sets the width of the lines produced in Line Mode.'
        .BYTE $07
        ; sequencerSpeed: 'Controls the rate at which sequencer feeds in its data. '
        .BYTE $08
        ; pulseWidth: 'Sets the length of the pulses in a pulsed stream output.
        ; Don't worry about what that means - just get in there and mess with it.'
        .BYTE $01
        ; baseLevel: 'Controls how many 'levels' of pattern are plotted.'
        .BYTE $07
        ; presetColorValuesArray: 'Allows you to set the colour for each of the
        ; seven pattern steps. Set up the colour you want, press RETURN, and the
        ; command offers the next colour along, up to no. 7, then ends. Cannot be
        ; adjusted while patterns being generated.'
        .BYTE BLACK,BROWN,RED,ORANGE,PURPLE,LTRED,BLUE,LTBLUE
        ; trackingActivated: 'Controls whether logic-seeking is used in the
        ; buffer or not. The upshot of this for you is a slightly different feel -
        ; continuous but fragmented when ON, or together-ish bursts when OFF. Try it.'
        .BYTE $FF
        ; lineModeActivated: 'A bit like drawing with the Aurora Borealis'
        .BYTE $00
        ; presetIndex: 'This calls in one of the 16 presets, stored Lightsynth
        ; parameters which give different effects. Try them all out io see some uf
        ; the multitude of effects which you cai achieve using the system. Some are
        ; fast, some slow, some pulse, others swirl. Play with them all, try them to
        ; different music.'
        .BYTE $01
        ; currentPatternElement: 'Initial pattern used by this preset.'
        .BYTE $01
        ; currentSymmetrySetting: 'Current symmetry setting.'
        ; Possible values are 0 - 4:
        ; 'NO SYMMETRY     '
        ; 'Y-AXIS SYMMETRY '
        ; 'X-Y SYMMETRY    '
        ; 'X-AXIS SYMMETRY '
        ; 'QUAD SYMMETRY   '
        .BYTE $04
        ; Unused Data.
        .BYTE $FF,$00,$FF,$FF,$00,$FF,$00,$FF,$00
```

Listing 4.2: Source code for Preset 1.

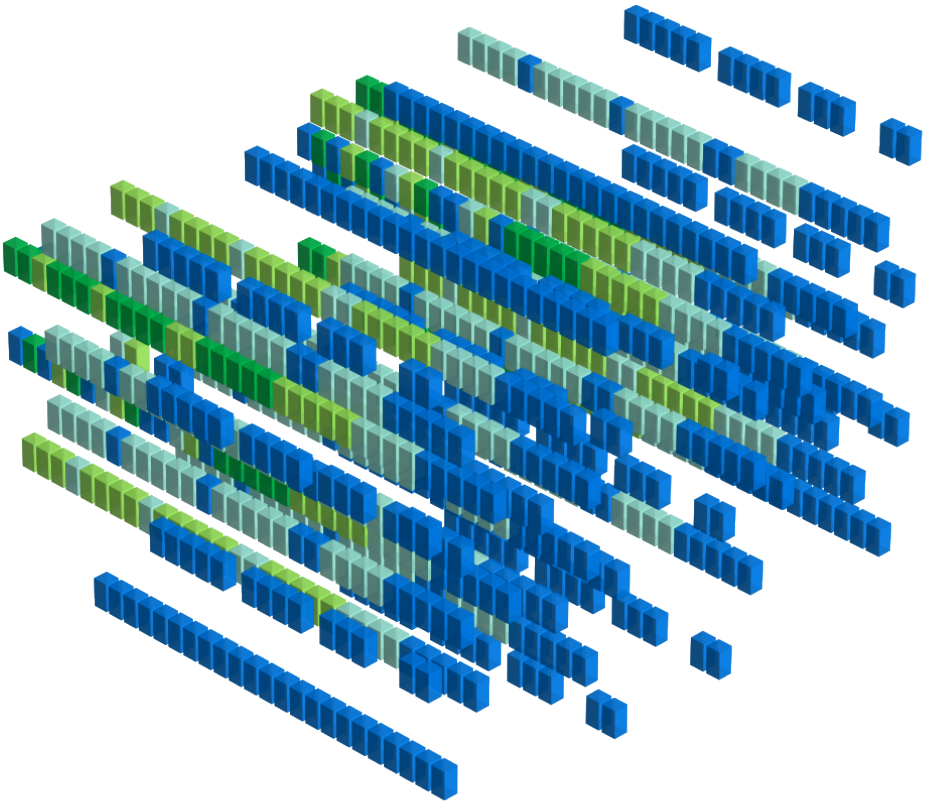


Figure 4.3: Evolution of Preset 2.

```
preset2
        ; unusedPresetByte: Unused Byte
        .BYTE $00
        ; smoothingDelay: 'Because of the time taken to draw larger patterns speed
        ; increase/decrease is not linear. You can adjust the 'compensating delay'
        ; which often smooths out jerky patterns. Can be used just for special FX),
        ; though. Suck it and see.'
        .BYTE $0B
        ; cursorSpeed: 'Gives you a slow or fast little cursor, according to setting.'
        .BYTE $02
        ; bufferLength: 'Larger patterns flow more smoothly with a shorter
        ; Buffer Length - not so many positions are retained so less plotting to do.
        ; Small patterns with a long Buffer Length are good for 'steamer' effects.
        ; N.B. Cannot be adjusted whilst patterns are actually onscreen.'
        .BYTE $28
        ; pulseSpeed: 'Usually if you hold down the button you get a continuous
        ; stream. Setting the Pulse Speed allows you to generate a pulsed stream, as
        ; if you were rapidly pressing and releasing the FIRE button.'
        .BYTE $01
        ; indexForColorBarDisplay: 'The initial index for the color displayed
        ; in the color bar when adjusting the colors for each step.'
        .BYTE $01
        ; lineWidth: 'Sets the width of the lines produced in Line Mode.'
        .BYTE $07
        ; sequencerSpeed: 'Controls the rate at which sequencer feeds in its data. '
        .BYTE $0B
        ; pulseWidth: 'Sets the length of the pulses in a pulsed stream output.
        ; Don't worry about what that means - just get in there and mess with it.'
        .BYTE $01
        ; baseLevel: 'Controls how many 'levels' of pattern are plotted.'
        .BYTE $07
        ; presetColorValuesArray: 'Allows you to set the colour for each of the
        ; seven pattern steps. Set up the colour you want, press RETURN, and the
        ; command offers the next colour along, up to no. 7, then ends. Cannot be
        ; adjusted while patterns being generated.'
        .BYTE BLACK,BLUE,LTBLUE,CYAN,LTGREEN,GREEN,LTBLUE,BLUE
        ; trackingActivated: 'Controls whether logic-seeking is used in the
        ; buffer or not. The upshot of this for you is a slightly different feel -
        ; continuous but fragmented when ON, or together-ish bursts when OFF. Try it.'
        .BYTE $FF
        ; lineModeActivated: 'A bit like drawing with the Aurora Borealis'
        .BYTE $00
        ; presetIndex: 'This calls in one of the 16 presets, stored Lightsynth
        ; parameters which give different effects. Try them all out io see some uf
        ; the multitude of effects which you cai achieve using the system. Some are
        ; fast, some slow, some pulse, others swirl. Play with them all, try them to
        ; different music.'
        .BYTE $05
        ; currentPatternElement: 'Initial pattern used by this preset.'
        .BYTE $05
        ; currentSymmetrySetting: 'Current symmetry setting.'
        ; Possible values are 0 - 4:
        ; 'NO SYMMETRY     '
        ; 'Y-AXIS SYMMETRY '
        ; 'X-Y SYMMETRY    '
        ; 'X-AXIS SYMMETRY '
        ; 'QUAD SYMMETRY   '
        .BYTE $01
        ; Unused Data.
        .BYTE $FF,$00,$FF,$FF,$00,$FF,$00,$FF,$00
```

Listing 4.3: Source code for Preset 2.

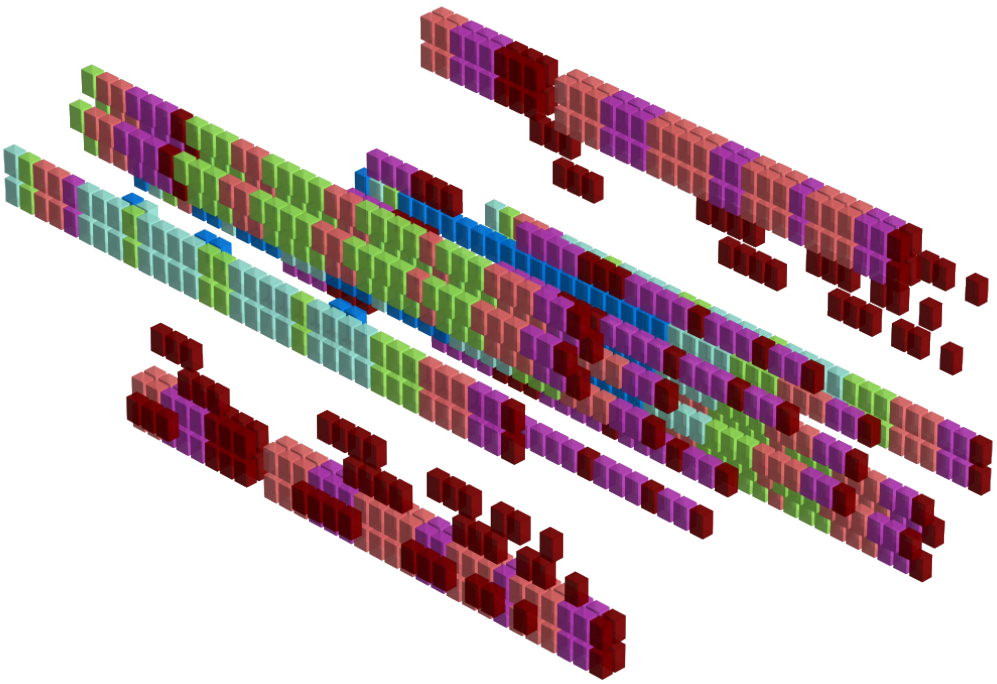


Figure 4.4: Evolution of Preset 3.

```
preset3
        ; unusedPresetByte: Unused Byte
        .BYTE $00
        ; smoothingDelay: 'Because of the time taken to draw larger patterns speed
        ; increase/decrease is not linear. You can adjust the 'compensating delay'
        ; which often smooths out jerky patterns. Can be used just for special FX),
        ; though. Suck it and see.'
        .BYTE $04
        ; cursorSpeed: 'Gives you a slow or fast little cursor, according to setting.'
        .BYTE $02
        ; bufferLength: 'Larger patterns flow more smoothly with a shorter
        ; Buffer Length - not so many positions are retained so less plotting to do.
        ; Small patterns with a long Buffer Length are good for 'steamer' effects.
        ; N.B. Cannot be adjusted whilst patterns are actually onscreen.'
        .BYTE $26
        ; pulseSpeed: 'Usually if you hold down the button you get a continuous
        ; stream. Setting the Pulse Speed allows you to generate a pulsed stream, as
        ; if you were rapidly pressing and releasing the FIRE button.'
        .BYTE $01
        ; indexForColorBarDisplay: 'The initial index for the color displayed
        ; in the color bar when adjusting the colors for each step.'
        .BYTE $01
        ; lineWidth: 'Sets the width of the lines produced in Line Mode.'
        .BYTE $07
        ; sequencerSpeed: 'Controls the rate at which sequencer feeds in its data. '
        .BYTE $0A
        ; pulseWidth: 'Sets the length of the pulses in a pulsed stream output.
        ; Don't worry about what that means - just get in there and mess with it.'
        .BYTE $01
        ; baseLevel: 'Controls how many 'levels' of pattern are plotted.'
        .BYTE $07
        ; presetColorValuesArray: 'Allows you to set the colour for each of the
        ; seven pattern steps. Set up the colour you want, press RETURN, and the
        ; command offers the next colour along, up to no. 7, then ends. Cannot be
        ; adjusted while patterns being generated.'
        .BYTE BLACK,RED,PURPLE,LTRED,LTGREEN,CYAN,LTBLUE,BLUE
        ; trackingActivated: 'Controls whether logic-seeking is used in the
        ; buffer or not. The upshot of this for you is a slightly different feel -
        ; continuous but fragmented when ON, or together-ish bursts when OFF. Try it.'
        .BYTE $00
        ; lineModeActivated: 'A bit like drawing with the Aurora Borealis'
        .BYTE $00
        ; presetIndex: 'This calls in one of the 16 presets, stored Lightsynth
        ; parameters which give different effects. Try them all out io see some uf
        ; the multitude of effects which you cai achieve using the system. Some are
        ; fast, some slow, some pulse, others swirl. Play with them all, try them to
        ; different music.'
        .BYTE $0E
        ; currentPatternElement: 'Initial pattern used by this preset.'
        .BYTE $0E
        ; currentSymmetrySetting: 'Current symmetry setting.'
        ; Possible values are 0 - 4:
        ; 'NO SYMMETRY     '
        ; 'Y-AXIS SYMMETRY '
        ; 'X-Y SYMMETRY    '
        ; 'X-AXIS SYMMETRY '
        ; 'QUAD SYMMETRY   '
        .BYTE $02
        ; Unused Data.
        .BYTE $FF,$00,$FF,$FF,$00,$FF,$00,$EA,$10
```

Listing 4.4: Source code for Preset 3.

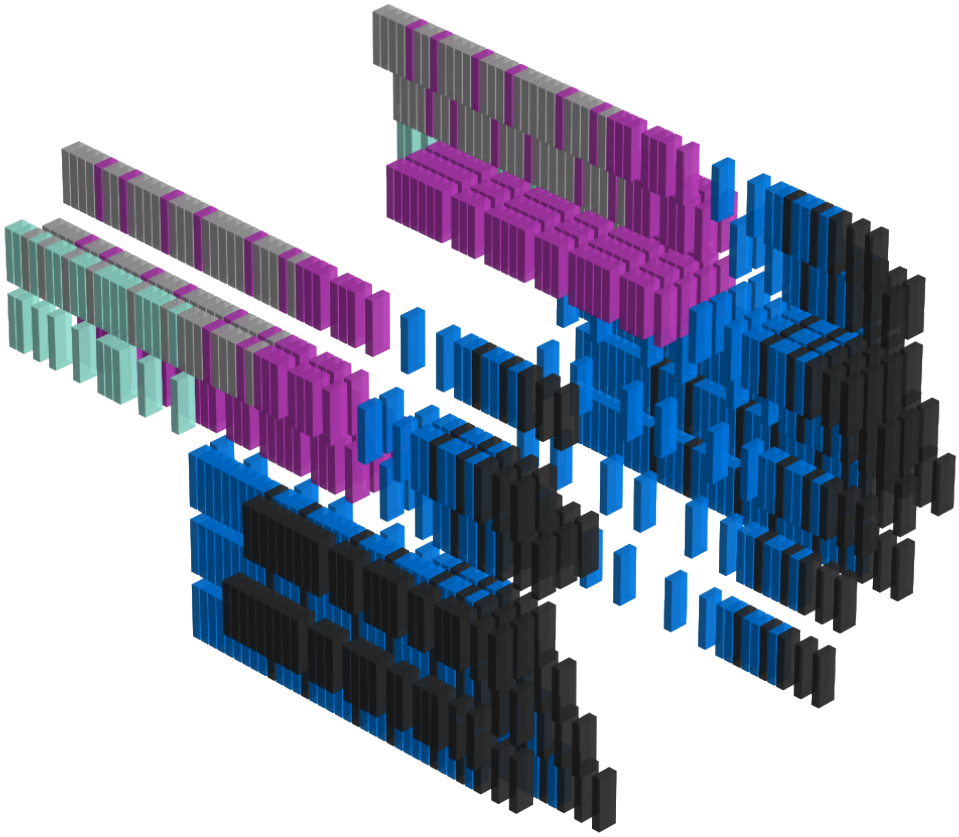


Figure 4.5: Evolution of Preset 4.

```
preset4
        ; unusedPresetByte: Unused Byte
        .BYTE $00
        ; smoothingDelay: 'Because of the time taken to draw larger patterns speed
        ; increase/decrease is not linear. You can adjust the 'compensating delay'
        ; which often smooths out jerky patterns. Can be used just for special FX),
        ; though. Suck it and see.'
        .BYTE $0C
        ; cursorSpeed: 'Gives you a slow or fast little cursor, according to setting.'
        .BYTE $01
        ; bufferLength: 'Larger patterns flow more smoothly with a shorter
        ; Buffer Length - not so many positions are retained so less plotting to do.
        ; Small patterns with a long Buffer Length are good for 'steamer' effects.
        ; N.B. Cannot be adjusted whilst patterns are actually onscreen.'
        .BYTE $2B
        ; pulseSpeed: 'Usually if you hold down the button you get a continuous
        ; stream. Setting the Pulse Speed allows you to generate a pulsed stream, as
        ; if you were rapidly pressing and releasing the FIRE button.'
        .BYTE $01
        ; indexForColorBarDisplay: 'The initial index for the color displayed
        ; in the color bar when adjusting the colors for each step.'
        .BYTE $07
        ; lineWidth: 'Sets the width of the lines produced in Line Mode.'
        .BYTE $07
        ; sequencerSpeed: 'Controls the rate at which sequencer feeds in its data. '
        .BYTE $08
        ; pulseWidth: 'Sets the length of the pulses in a pulsed stream output.
        ; Don't worry about what that means - just get in there and mess with it.'
        .BYTE $01
        ; baseLevel: 'Controls how many 'levels' of pattern are plotted.'
        .BYTE $07
        ; presetColorValuesArray: 'Allows you to set the colour for each of the
        ; seven pattern steps. Set up the colour you want, press RETURN, and the
        ; command offers the next colour along, up to no. 7, then ends. Cannot be
        ; adjusted while patterns being generated.'
        .BYTE BLACK,GRAY1,BLUE,GRAY2,PURPLE,GRAY3,CYAN,WHITE
        ; trackingActivated: 'Controls whether logic-seeking is used in the
        ; buffer or not. The upshot of this for you is a slightly different feel -
        ; continuous but fragmented when ON, or together-ish bursts when OFF. Try it.'
        .BYTE $00
        ; lineModeActivated: 'A bit like drawing with the Aurora Borealis'
        .BYTE $00
        ; presetIndex: 'This calls in one of the 16 presets, stored Lightsynth
        ; parameters which give different effects. Try them all out io see some uf
        ; the multitude of effects which you cai achieve using the system. Some are
        ; fast, some slow, some pulse, others swirl. Play with them all, try them to
        ; different music.'
        .BYTE $01
        ; currentPatternElement: 'Initial pattern used by this preset.'
        .BYTE $01
        ; currentSymmetrySetting: 'Current symmetry setting.'
        ; Possible values are 0 - 4:
        ; 'NO SYMMETRY     '
        ; 'Y-AXIS SYMMETRY '
        ; 'X-Y SYMMETRY    '
        ; 'X-AXIS SYMMETRY '
        ; 'QUAD SYMMETRY   '
        .BYTE $01
        ; Unused Data.
        .BYTE $00,$FF,$00,$00,$FF,$00,$FF,$00,$FF
```

Listing 4.5: Source code for Preset 4.

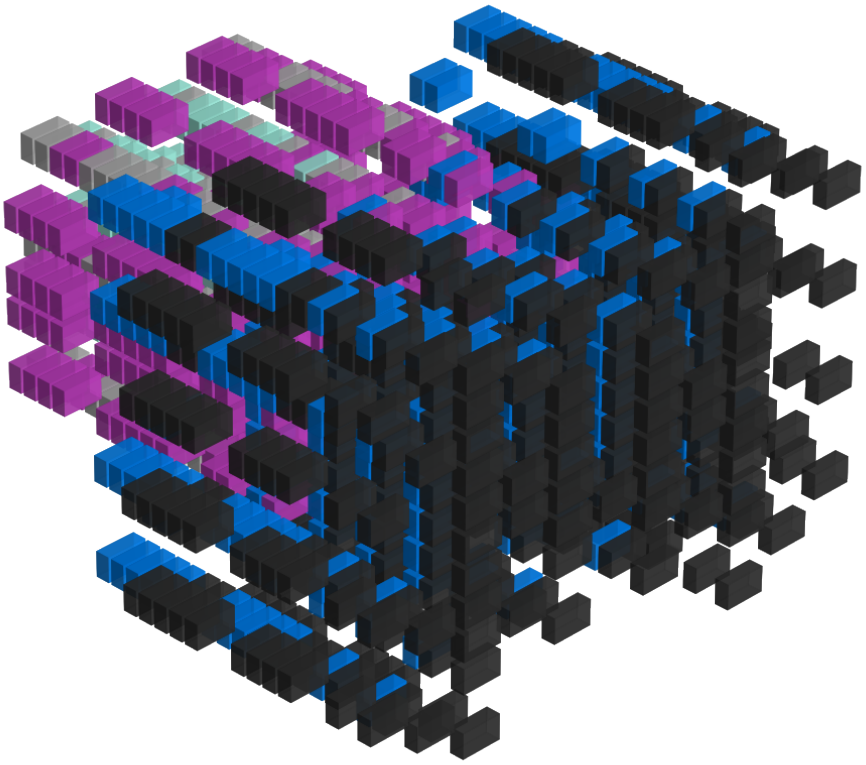


Figure 4.6: Evolution of Preset 5.

```
preset5
        ; unusedPresetByte: Unused Byte
        .BYTE $00
        ; smoothingDelay: 'Because of the time taken to draw larger patterns speed
        ; increase/decrease is not linear. You can adjust the 'compensating delay'
        ; which often smooths out jerky patterns. Can be used just for special FX),
        ; though. Suck it and see.'
        .BYTE $0C
        ; cursorSpeed: 'Gives you a slow or fast little cursor, according to setting.'
        .BYTE $02
        ; bufferLength: 'Larger patterns flow more smoothly with a shorter
        ; Buffer Length - not so many positions are retained so less plotting to do.
        ; Small patterns with a long Buffer Length are good for 'steamer' effects.
        ; N.B. Cannot be adjusted whilst patterns are actually onscreen.'
        .BYTE $2B
        ; pulseSpeed: 'Usually if you hold down the button you get a continuous
        ; stream. Setting the Pulse Speed allows you to generate a pulsed stream, as
        ; if you were rapidly pressing and releasing the FIRE button.'
        .BYTE $01
        ; indexForColorBarDisplay: 'The initial index for the color displayed
        ; in the color bar when adjusting the colors for each step.'
        .BYTE $07
        ; lineWidth: 'Sets the width of the lines produced in Line Mode.'
        .BYTE $07
        ; sequencerSpeed: 'Controls the rate at which sequencer feeds in its data. '
        .BYTE $0C
        ; pulseWidth: 'Sets the length of the pulses in a pulsed stream output.
        ; Don't worry about what that means - just get in there and mess with it.'
        .BYTE $01
        ; baseLevel: 'Controls how many 'levels' of pattern are plotted.'
        .BYTE $07
        ; presetColorValuesArray: 'Allows you to set the colour for each of the
        ; seven pattern steps. Set up the colour you want, press RETURN, and the
        ; command offers the next colour along, up to no. 7, then ends. Cannot be
        ; adjusted while patterns being generated.'
        .BYTE BLACK,GRAY1,BLUE,GRAY2,PURPLE,GRAY3,CYAN,WHITE
        ; trackingActivated: 'Controls whether logic-seeking is used in the
        ; buffer or not. The upshot of this for you is a slightly different feel -
        ; continuous but fragmented when ON, or together-ish bursts when OFF. Try it.'
        .BYTE $00
        ; lineModeActivated: 'A bit like drawing with the Aurora Borealis'
        .BYTE $00
        ; presetIndex: 'This calls in one of the 16 presets, stored Lightsynth
        ; parameters which give different effects. Try them all out io see some uf
        ; the multitude of effects which you cai achieve using the system. Some are
        ; fast, some slow, some pulse, others swirl. Play with them all, try them to
        ; different music.'
        .BYTE $06
        ; currentPatternElement: 'Initial pattern used by this preset.'
        .BYTE $06
        ; currentSymmetrySetting: 'Current symmetry setting.'
        ; Possible values are 0 - 4:
        ; 'NO SYMMETRY     '
        ; 'Y-AXIS SYMMETRY '
        ; 'X-Y SYMMETRY    '
        ; 'X-AXIS SYMMETRY '
        ; 'QUAD SYMMETRY   '
        .BYTE $03
        ; Unused Data.
        .BYTE $00,$FF,$00,$00,$FF,$00,$FF,$00,$F7
```

Listing 4.6: Source code for Preset 5.

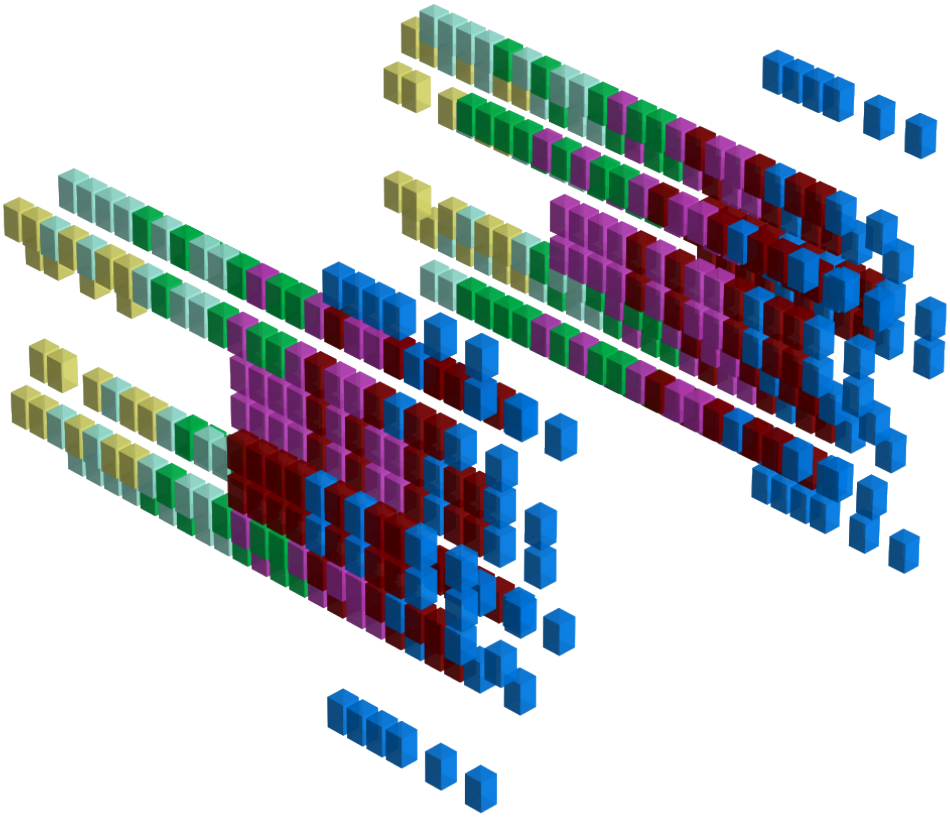


Figure 4.7: Evolution of Preset 6.

```
preset6
        ; unusedPresetByte: Unused Byte
        .BYTE $00
        ; smoothingDelay: 'Because of the time taken to draw larger patterns speed
        ; increase/decrease is not linear. You can adjust the 'compensating delay'
        ; which often smooths out jerky patterns. Can be used just for special FX),
        ; though. Suck it and see.'
        .BYTE $0F
        ; cursorSpeed: 'Gives you a slow or fast little cursor, according to setting.'
        .BYTE $02
        ; bufferLength: 'Larger patterns flow more smoothly with a shorter
        ; Buffer Length - not so many positions are retained so less plotting to do.
        ; Small patterns with a long Buffer Length are good for 'steamer' effects.
        ; N.B. Cannot be adjusted whilst patterns are actually onscreen.'
        .BYTE $3F
        ; pulseSpeed: 'Usually if you hold down the button you get a continuous
        ; stream. Setting the Pulse Speed allows you to generate a pulsed stream, as
        ; if you were rapidly pressing and releasing the FIRE button.'
        .BYTE $01
        ; indexForColorBarDisplay: 'The initial index for the color displayed
        ; in the color bar when adjusting the colors for each step.'
        .BYTE $01
        ; lineWidth: 'Sets the width of the lines produced in Line Mode.'
        .BYTE $07
        ; sequencerSpeed: 'Controls the rate at which sequencer feeds in its data. '
        .BYTE $0F
        ; pulseWidth: 'Sets the length of the pulses in a pulsed stream output.
        ; Don't worry about what that means - just get in there and mess with it.'
        .BYTE $01
        ; baseLevel: 'Controls how many 'levels' of pattern are plotted.'
        .BYTE $07
        ; presetColorValuesArray: 'Allows you to set the colour for each of the
        ; seven pattern steps. Set up the colour you want, press RETURN, and the
        ; command offers the next colour along, up to no. 7, then ends. Cannot be
        ; adjusted while patterns being generated.'
        .BYTE BLACK,BLUE,RED,PURPLE,GREEN,CYAN,YELLOW,WHITE
        ; trackingActivated: 'Controls whether logic-seeking is used in the
        ; buffer or not. The upshot of this for you is a slightly different feel -
        ; continuous but fragmented when ON, or together-ish bursts when OFF. Try it.'
        .BYTE $FF
        ; lineModeActivated: 'A bit like drawing with the Aurora Borealis'
        .BYTE $00
        ; presetIndex: 'This calls in one of the 16 presets, stored Lightsynth
        ; parameters which give different effects. Try them all out io see some uf
        ; the multitude of effects which you cai achieve using the system. Some are
        ; fast, some slow, some pulse, others swirl. Play with them all, try them to
        ; different music.'
        .BYTE $03
        ; currentPatternElement: 'Initial pattern used by this preset.'
        .BYTE $03
        ; currentSymmetrySetting: 'Current symmetry setting.'
        ; Possible values are 0 - 4:
        ; 'NO SYMMETRY     '
        ; 'Y-AXIS SYMMETRY '
        ; 'X-Y SYMMETRY    '
        ; 'X-AXIS SYMMETRY '
        ; 'QUAD SYMMETRY   '
        .BYTE $04
        ; Unused Data.
        .BYTE $00,$FF,$00,$00,$FF,$00,$FF,$00,$FF
```

Listing 4.7: Source code for Preset 6.

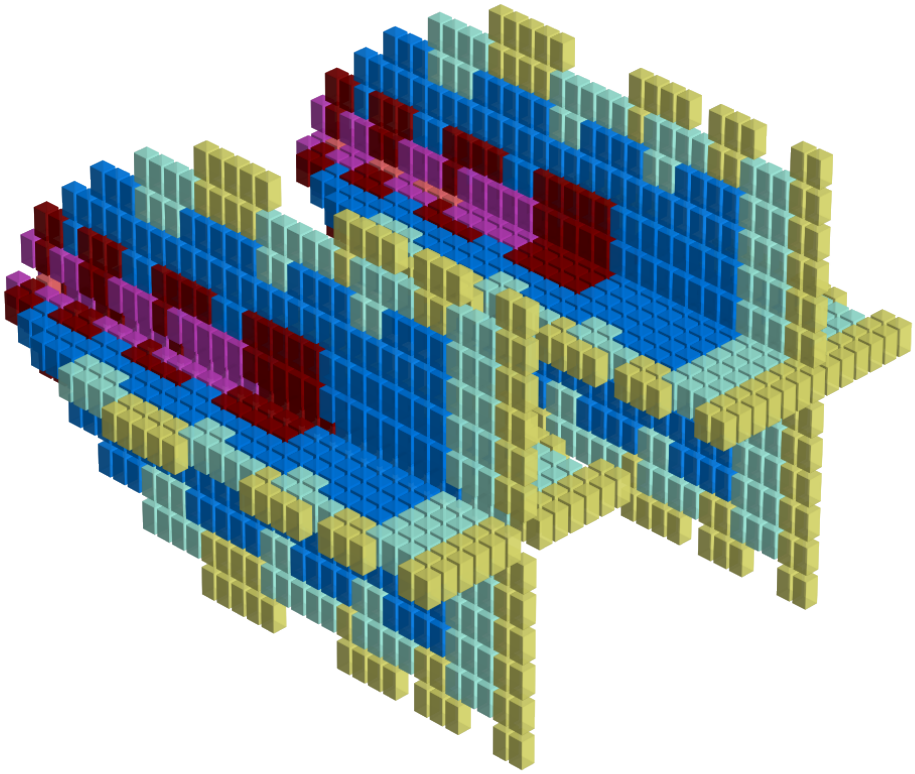


Figure 4.8: Evolution of Preset 7.

```
preset7
        ; unusedPresetByte: Unused Byte
        .BYTE $00
        ; smoothingDelay: 'Because of the time taken to draw larger patterns speed
        ; increase/decrease is not linear. You can adjust the 'compensating delay'
        ; which often smooths out jerky patterns. Can be used just for special FX),
        ; though. Suck it and see.'
        .BYTE $0B
        ; cursorSpeed: 'Gives you a slow or fast little cursor, according to setting.'
        .BYTE $01
        ; bufferLength: 'Larger patterns flow more smoothly with a shorter
        ; Buffer Length - not so many positions are retained so less plotting to do.
        ; Small patterns with a long Buffer Length are good for 'steamer' effects.
        ; N.B. Cannot be adjusted whilst patterns are actually onscreen.'
        .BYTE $1C
        ; pulseSpeed: 'Usually if you hold down the button you get a continuous
        ; stream. Setting the Pulse Speed allows you to generate a pulsed stream, as
        ; if you were rapidly pressing and releasing the FIRE button.'
        .BYTE $02
        ; indexForColorBarDisplay: 'The initial index for the color displayed
        ; in the color bar when adjusting the colors for each step.'
        .BYTE $0A
        ; lineWidth: 'Sets the width of the lines produced in Line Mode.'
        .BYTE $07
        ; sequencerSpeed: 'Controls the rate at which sequencer feeds in its data. '
        .BYTE $09
        ; pulseWidth: 'Sets the length of the pulses in a pulsed stream output.
        ; Don't worry about what that means - just get in there and mess with it.'
        .BYTE $01
        ; baseLevel: 'Controls how many 'levels' of pattern are plotted.'
        .BYTE $07
        ; presetColorValuesArray: 'Allows you to set the colour for each of the
        ; seven pattern steps. Set up the colour you want, press RETURN, and the
        ; command offers the next colour along, up to no. 7, then ends. Cannot be
        ; adjusted while patterns being generated.'
        .BYTE BLACK,YELLOW,CYAN,LTBLUE,BLUE,RED,PURPLE,LTRED
        ; trackingActivated: 'Controls whether logic-seeking is used in the
        ; buffer or not. The upshot of this for you is a slightly different feel -
        ; continuous but fragmented when ON, or together-ish bursts when OFF. Try it.'
        .BYTE $00
        ; lineModeActivated: 'A bit like drawing with the Aurora Borealis'
        .BYTE $00
        ; presetIndex: 'This calls in one of the 16 presets, stored Lightsynth
        ; parameters which give different effects. Try them all out io see some uf
        ; the multitude of effects which you cai achieve using the system. Some are
        ; fast, some slow, some pulse, others swirl. Play with them all, try them to
        ; different music.'
        .BYTE $07
        ; currentPatternElement: 'Initial pattern used by this preset.'
        .BYTE $07
        ; currentSymmetrySetting: 'Current symmetry setting.'
        ; Possible values are 0 - 4:
        ; 'NO SYMMETRY     '
        ; 'Y-AXIS SYMMETRY '
        ; 'X-Y SYMMETRY    '
        ; 'X-AXIS SYMMETRY '
        ; 'QUAD SYMMETRY   '
        .BYTE $01
        ; Unused Data.
        .BYTE $00,$FF,$00,$00,$FF,$00,$FF,$15,$EF
```

Listing 4.8: Source code for Preset 7.

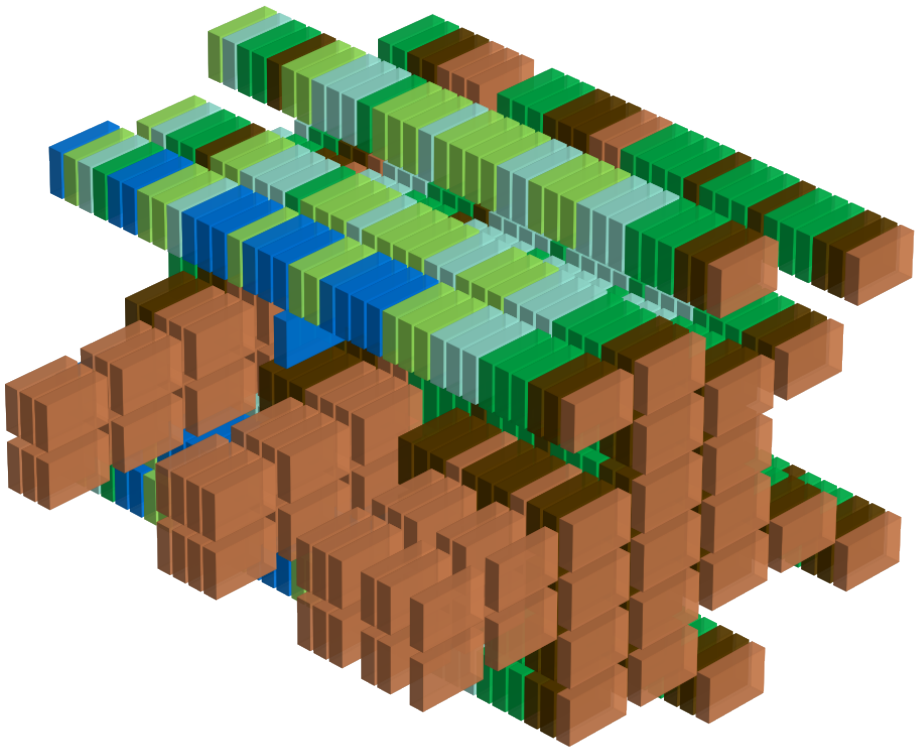


Figure 4.9: Evolution of Preset 8.

```
preset8
        ; unusedPresetByte: Unused Byte
        .BYTE $00
        ; smoothingDelay: 'Because of the time taken to draw larger patterns speed
        ; increase/decrease is not linear. You can adjust the 'compensating delay'
        ; which often smooths out jerky patterns. Can be used just for special FX),
        ; though. Suck it and see.'
        .BYTE $04
        ; cursorSpeed: 'Gives you a slow or fast little cursor, according to setting.'
        .BYTE $01
        ; bufferLength: 'Larger patterns flow more smoothly with a shorter
        ; Buffer Length - not so many positions are retained so less plotting to do.
        ; Small patterns with a long Buffer Length are good for 'steamer' effects.
        ; N.B. Cannot be adjusted whilst patterns are actually onscreen.'
        .BYTE $28
        ; pulseSpeed: 'Usually if you hold down the button you get a continuous
        ; stream. Setting the Pulse Speed allows you to generate a pulsed stream, as
        ; if you were rapidly pressing and releasing the FIRE button.'
        .BYTE $02
        ; indexForColorBarDisplay: 'The initial index for the color displayed
        ; in the color bar when adjusting the colors for each step.'
        .BYTE $01
        ; lineWidth: 'Sets the width of the lines produced in Line Mode.'
        .BYTE $07
        ; sequencerSpeed: 'Controls the rate at which sequencer feeds in its data. '
        .BYTE $0A
        ; pulseWidth: 'Sets the length of the pulses in a pulsed stream output.
        ; Don't worry about what that means - just get in there and mess with it.'
        .BYTE $01
        ; baseLevel: 'Controls how many 'levels' of pattern are plotted.'
        .BYTE $07
        ; presetColorValuesArray: 'Allows you to set the colour for each of the
        ; seven pattern steps. Set up the colour you want, press RETURN, and the
        ; command offers the next colour along, up to no. 7, then ends. Cannot be
        ; adjusted while patterns being generated.'
        .BYTE BLACK,ORANGE,BROWN,GREEN,CYAN,LTGREEN,LTBLUE,BLUE
        ; trackingActivated: 'Controls whether logic-seeking is used in the
        ; buffer or not. The upshot of this for you is a slightly different feel -
        ; continuous but fragmented when ON, or together-ish bursts when OFF. Try it.'
        .BYTE $FF
        ; lineModeActivated: 'A bit like drawing with the Aurora Borealis'
        .BYTE $00
        ; presetIndex: 'This calls in one of the 16 presets, stored Lightsynth
        ; parameters which give different effects. Try them all out io see some uf
        ; the multitude of effects which you cai achieve using the system. Some are
        ; fast, some slow, some pulse, others swirl. Play with them all, try them to
        ; different music.'
        .BYTE $01
        ; currentPatternElement: 'Initial pattern used by this preset.'
        .BYTE $01
        ; currentSymmetrySetting: 'Current symmetry setting.'
        ; Possible values are 0 - 4:
        ; 'NO SYMMETRY     '
        ; 'Y-AXIS SYMMETRY '
        ; 'X-Y SYMMETRY    '
        ; 'X-AXIS SYMMETRY '
        ; 'QUAD SYMMETRY   '
        .BYTE $03
        ; Unused Data.
        .BYTE $FF,$00,$FF,$FF,$00,$FF,$00,$FF,$00
```

Listing 4.9: Source code for Preset 8.

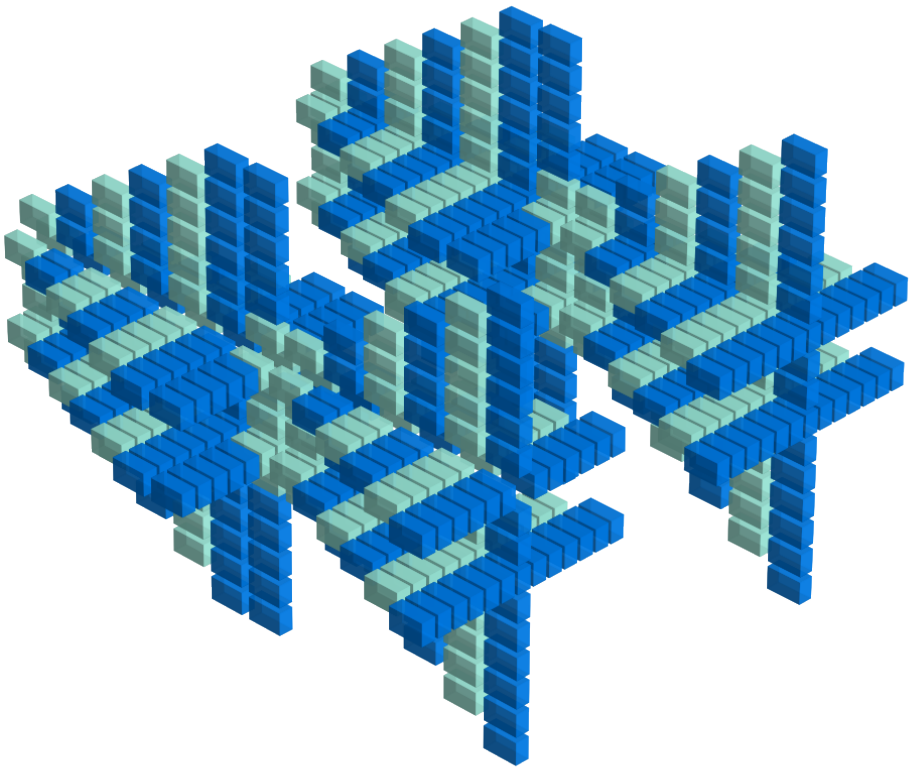


Figure 4.10: Evolution of Preset 9.

```
preset9
        ; unusedPresetByte: Unused Byte
        .BYTE $00
        ; smoothingDelay: 'Because of the time taken to draw larger patterns speed
        ; increase/decrease is not linear. You can adjust the 'compensating delay'
        ; which often smooths out jerky patterns. Can be used just for special FX),
        ; though. Suck it and see.'
        .BYTE $11
        ; cursorSpeed: 'Gives you a slow or fast little cursor, according to setting.'
        .BYTE $01
        ; bufferLength: 'Larger patterns flow more smoothly with a shorter
        ; Buffer Length - not so many positions are retained so less plotting to do.
        ; Small patterns with a long Buffer Length are good for 'steamer' effects.
        ; N.B. Cannot be adjusted whilst patterns are actually onscreen.'
        .BYTE $0D
        ; pulseSpeed: 'Usually if you hold down the button you get a continuous
        ; stream. Setting the Pulse Speed allows you to generate a pulsed stream, as
        ; if you were rapidly pressing and releasing the FIRE button.'
        .BYTE $07
        ; indexForColorBarDisplay: 'The initial index for the color displayed
        ; in the color bar when adjusting the colors for each step.'
        .BYTE $01
        ; lineWidth: 'Sets the width of the lines produced in Line Mode.'
        .BYTE $07
        ; sequencerSpeed: 'Controls the rate at which sequencer feeds in its data. '
        .BYTE $0C
        ; pulseWidth: 'Sets the length of the pulses in a pulsed stream output.
        ; Don't worry about what that means - just get in there and mess with it.'
        .BYTE $01
        ; baseLevel: 'Controls how many 'levels' of pattern are plotted.'
        .BYTE $07
        ; presetColorValuesArray: 'Allows you to set the colour for each of the
        ; seven pattern steps. Set up the colour you want, press RETURN, and the
        ; command offers the next colour along, up to no. 7, then ends. Cannot be
        ; adjusted while patterns being generated.'
        .BYTE BLACK,BLUE,CYAN,BLUE,CYAN,BLUE,CYAN,BLUE
        ; trackingActivated: 'Controls whether logic-seeking is used in the
        ; buffer or not. The upshot of this for you is a slightly different feel -
        ; continuous but fragmented when ON, or together-ish bursts when OFF. Try it.'
        .BYTE $FF
        ; lineModeActivated: 'A bit like drawing with the Aurora Borealis'
        .BYTE $00
        ; presetIndex: 'This calls in one of the 16 presets, stored Lightsynth
        ; parameters which give different effects. Try them all out io see some uf
        ; the multitude of effects which you cai achieve using the system. Some are
        ; fast, some slow, some pulse, others swirl. Play with them all, try them to
        ; different music.'
        .BYTE $07
        ; currentPatternElement: 'Initial pattern used by this preset.'
        .BYTE $07
        ; currentSymmetrySetting: 'Current symmetry setting.'
        ; Possible values are 0 - 4:
        ; 'NO SYMMETRY     '
        ; 'Y-AXIS SYMMETRY '
        ; 'X-Y SYMMETRY    '
        ; 'X-AXIS SYMMETRY '
        ; 'QUAD SYMMETRY   '
        .BYTE $04
        ; Unused Data.
        .BYTE $FF,$00,$FF,$FF,$00,$FF,$00,$FF,$00
```

Listing 4.10: Source code for Preset 9.

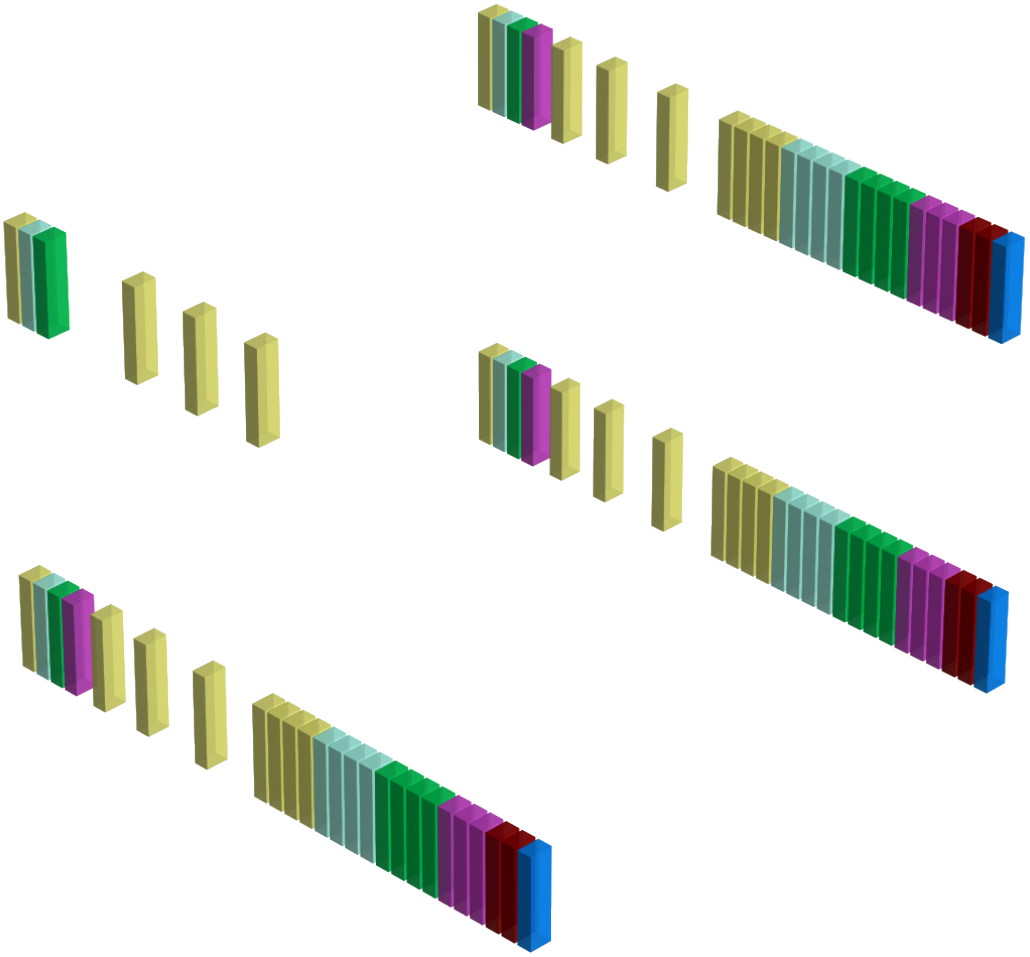


Figure 4.11: Evolution of Preset 10.

```
preset10
        ; unusedPresetByte: Unused Byte
        .BYTE $00
        ; smoothingDelay: 'Because of the time taken to draw larger patterns speed
        ; increase/decrease is not linear. You can adjust the 'compensating delay'
        ; which often smooths out jerky patterns. Can be used just for special FX),
        ; though. Suck it and see.'
        .BYTE $01
        ; cursorSpeed: 'Gives you a slow or fast little cursor, according to setting.'
        .BYTE $02
        ; bufferLength: 'Larger patterns flow more smoothly with a shorter
        ; Buffer Length - not so many positions are retained so less plotting to do.
        ; Small patterns with a long Buffer Length are good for 'steamer' effects.
        ; N.B. Cannot be adjusted whilst patterns are actually onscreen.'
        .BYTE $1F
        ; pulseSpeed: 'Usually if you hold down the button you get a continuous
        ; stream. Setting the Pulse Speed allows you to generate a pulsed stream, as
        ; if you were rapidly pressing and releasing the FIRE button.'
        .BYTE $02
        ; indexForColorBarDisplay: 'The initial index for the color displayed
        ; in the color bar when adjusting the colors for each step.'
        .BYTE $09
        ; lineWidth: 'Sets the width of the lines produced in Line Mode.'
        .BYTE $04
        ; sequencerSpeed: 'Controls the rate at which sequencer feeds in its data. '
        .BYTE $08
        ; pulseWidth: 'Sets the length of the pulses in a pulsed stream output.
        ; Don't worry about what that means - just get in there and mess with it.'
        .BYTE $01
        ; baseLevel: 'Controls how many 'levels' of pattern are plotted.'
        .BYTE $07
        ; presetColorValuesArray: 'Allows you to set the colour for each of the
        ; seven pattern steps. Set up the colour you want, press RETURN, and the
        ; command offers the next colour along, up to no. 7, then ends. Cannot be
        ; adjusted while patterns being generated.'
        .BYTE BLACK,BLUE,RED,RED,PURPLE,LTRED,ORANGE,BROWN
        ; trackingActivated: 'Controls whether logic-seeking is used in the
        ; buffer or not. The upshot of this for you is a slightly different feel -
        ; continuous but fragmented when ON, or together-ish bursts when OFF. Try it.'
        .BYTE $FF
        ; lineModeActivated: 'A bit like drawing with the Aurora Borealis'
        .BYTE $01
        ; presetIndex: 'This calls in one of the 16 presets, stored Lightsynth
        ; parameters which give different effects. Try them all out io see some uf
        ; the multitude of effects which you cai achieve using the system. Some are
        ; fast, some slow, some pulse, others swirl. Play with them all, try them to
        ; different music.'
        .BYTE $00
        ; currentPatternElement: 'Initial pattern used by this preset.'
        .BYTE $00
        ; currentSymmetrySetting: 'Current symmetry setting.'
        ; Possible values are 0 - 4:
        ; 'NO SYMMETRY     '
        ; 'Y-AXIS SYMMETRY '
        ; 'X-Y SYMMETRY    '
        ; 'X-AXIS SYMMETRY '
        ; 'QUAD SYMMETRY   '
        .BYTE $04
        ; Unused Data.
        .BYTE $FF,$00,$FF,$FF,$00,$FF,$00,$FF,$00
```

Listing 4.11: Source code for Preset 10.

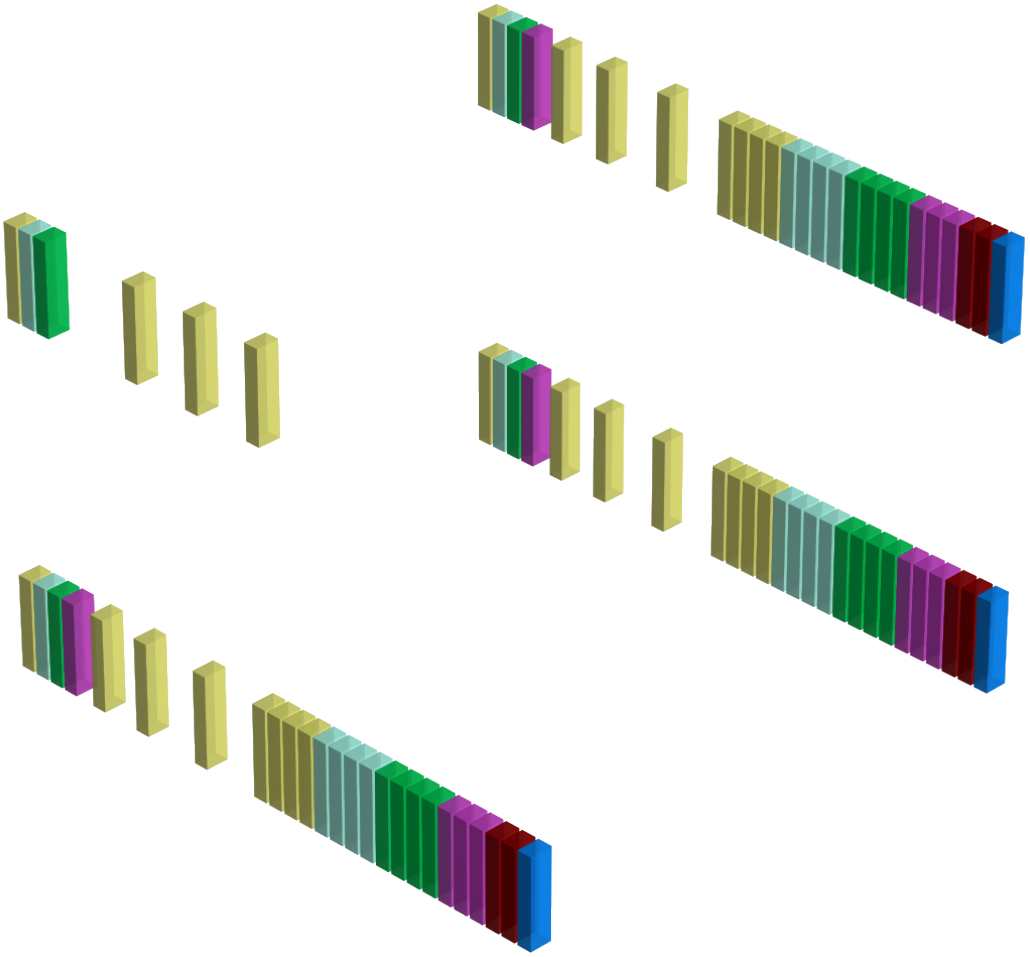


Figure 4.12: Evolution of Preset 11.

```
preset11
        ; unusedPresetByte: Unused Byte
        .BYTE $00
        ; smoothingDelay: 'Because of the time taken to draw larger patterns speed
        ; increase/decrease is not linear. You can adjust the 'compensating delay'
        ; which often smooths out jerky patterns. Can be used just for special FX),
        ; though. Suck it and see.'
        .BYTE $01
        ; cursorSpeed: 'Gives you a slow or fast little cursor, according to setting.'
        .BYTE $01
        ; bufferLength: 'Larger patterns flow more smoothly with a shorter
        ; Buffer Length - not so many positions are retained so less plotting to do.
        ; Small patterns with a long Buffer Length are good for 'steamer' effects.
        ; N.B. Cannot be adjusted whilst patterns are actually onscreen.'
        .BYTE $13
        ; pulseSpeed: 'Usually if you hold down the button you get a continuous
        ; stream. Setting the Pulse Speed allows you to generate a pulsed stream, as
        ; if you were rapidly pressing and releasing the FIRE button.'
        .BYTE $06
        ; indexForColorBarDisplay: 'The initial index for the color displayed
        ; in the color bar when adjusting the colors for each step.'
        .BYTE $01
        ; lineWidth: 'Sets the width of the lines produced in Line Mode.'
        .BYTE $07
        ; sequencerSpeed: 'Controls the rate at which sequencer feeds in its data. '
        .BYTE $08
        ; pulseWidth: 'Sets the length of the pulses in a pulsed stream output.
        ; Don't worry about what that means - just get in there and mess with it.'
        .BYTE $05
        ; baseLevel: 'Controls how many 'levels' of pattern are plotted.'
        .BYTE $07
        ; presetColorValuesArray: 'Allows you to set the colour for each of the
        ; seven pattern steps. Set up the colour you want, press RETURN, and the
        ; command offers the next colour along, up to no. 7, then ends. Cannot be
        ; adjusted while patterns being generated.'
        .BYTE BLACK,BLUE,RED,PURPLE,GREEN,CYAN,YELLOW,WHITE
        ; trackingActivated: 'Controls whether logic-seeking is used in the
        ; buffer or not. The upshot of this for you is a slightly different feel -
        ; continuous but fragmented when ON, or together-ish bursts when OFF. Try it.'
        .BYTE $FF
        ; lineModeActivated: 'A bit like drawing with the Aurora Borealis'
        .BYTE $00
        ; presetIndex: 'This calls in one of the 16 presets, stored Lightsynth
        ; parameters which give different effects. Try them all out io see some uf
        ; the multitude of effects which you cai achieve using the system. Some are
        ; fast, some slow, some pulse, others swirl. Play with them all, try them to
        ; different music.'
        .BYTE $0F
        ; currentPatternElement: 'Initial pattern used by this preset.'
        .BYTE $0F
        ; currentSymmetrySetting: 'Current symmetry setting.'
        ; Possible values are 0 - 4:
        ; 'NO SYMMETRY     '
        ; 'Y-AXIS SYMMETRY '
        ; 'X-Y SYMMETRY    '
        ; 'X-AXIS SYMMETRY '
        ; 'QUAD SYMMETRY   '
        .BYTE $04
        ; Unused Data.
        .BYTE $FF,$00,$FF,$FF,$00,$FF,$00,$EA,$10
```

Listing 4.12: Source code for Preset 11.

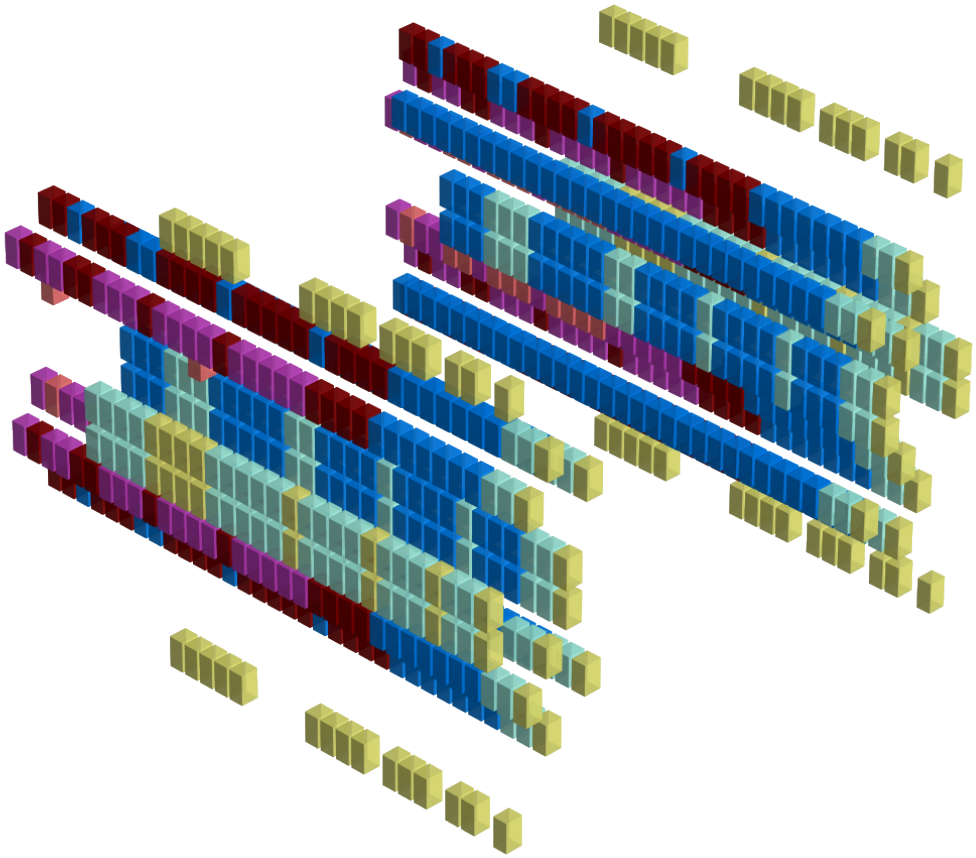


Figure 4.13: Evolution of Preset 12.

```
preset12
        ; unusedPresetByte: Unused Byte
        .BYTE $00
        ; smoothingDelay: 'Because of the time taken to draw larger patterns speed
        ; increase/decrease is not linear. You can adjust the 'compensating delay'
        ; which often smooths out jerky patterns. Can be used just for special FX),
        ; though. Suck it and see.'
        .BYTE $0C
        ; cursorSpeed: 'Gives you a slow or fast little cursor, according to setting.'
        .BYTE $02
        ; bufferLength: 'Larger patterns flow more smoothly with a shorter
        ; Buffer Length - not so many positions are retained so less plotting to do.
        ; Small patterns with a long Buffer Length are good for 'steamer' effects.
        ; N.B. Cannot be adjusted whilst patterns are actually onscreen.'
        .BYTE $28
        ; pulseSpeed: 'Usually if you hold down the button you get a continuous
        ; stream. Setting the Pulse Speed allows you to generate a pulsed stream, as
        ; if you were rapidly pressing and releasing the FIRE button.'
        .BYTE $01
        ; indexForColorBarDisplay: 'The initial index for the color displayed
        ; in the color bar when adjusting the colors for each step.'
        .BYTE $02
        ; lineWidth: 'Sets the width of the lines produced in Line Mode.'
        .BYTE $07
        ; sequencerSpeed: 'Controls the rate at which sequencer feeds in its data. '
        .BYTE $09
        ; pulseWidth: 'Sets the length of the pulses in a pulsed stream output.
        ; Don't worry about what that means - just get in there and mess with it.'
        .BYTE $01
        ; baseLevel: 'Controls how many 'levels' of pattern are plotted.'
        .BYTE $07
        ; presetColorValuesArray: 'Allows you to set the colour for each of the
        ; seven pattern steps. Set up the colour you want, press RETURN, and the
        ; command offers the next colour along, up to no. 7, then ends. Cannot be
        ; adjusted while patterns being generated.'
        .BYTE BLACK,BLUE,LTBLUE,CYAN,LTGREEN,YELLOW,PURPLE,RED
        ; trackingActivated: 'Controls whether logic-seeking is used in the
        ; buffer or not. The upshot of this for you is a slightly different feel -
        ; continuous but fragmented when ON, or together-ish bursts when OFF. Try it.'
        .BYTE $00
        ; lineModeActivated: 'A bit like drawing with the Aurora Borealis'
        .BYTE $00
        ; presetIndex: 'This calls in one of the 16 presets, stored Lightsynth
        ; parameters which give different effects. Try them all out io see some uf
        ; the multitude of effects which you cai achieve using the system. Some are
        ; fast, some slow, some pulse, others swirl. Play with them all, try them to
        ; different music.'
        .BYTE $0A
        ; currentPatternElement: 'Initial pattern used by this preset.'
        .BYTE $0A
        ; currentSymmetrySetting: 'Current symmetry setting.'
        ; Possible values are 0 - 4:
        ; 'NO SYMMETRY     '
        ; 'Y-AXIS SYMMETRY '
        ; 'X-Y SYMMETRY    '
        ; 'X-AXIS SYMMETRY '
        ; 'QUAD SYMMETRY   '
        .BYTE $01
        ; Unused Data.
        .BYTE $00,$FF,$00,$00,$FF,$00,$FF,$00,$FF
```

Listing 4.13: Source code for Preset 12.

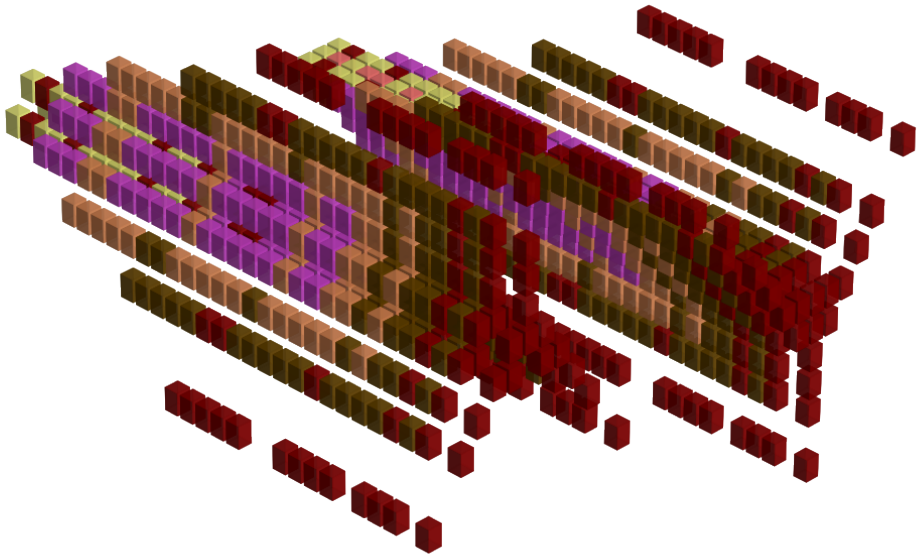


Figure 4.14: Evolution of Preset 13.

```
preset13
        ; unusedPresetByte: Unused Byte
        .BYTE $00
        ; smoothingDelay: 'Because of the time taken to draw larger patterns speed
        ; increase/decrease is not linear. You can adjust the 'compensating delay'
        ; which often smooths out jerky patterns. Can be used just for special FX),
        ; though. Suck it and see.'
        .BYTE $0B
        ; cursorSpeed: 'Gives you a slow or fast little cursor, according to setting.'
        .BYTE $01
        ; bufferLength: 'Larger patterns flow more smoothly with a shorter
        ; Buffer Length - not so many positions are retained so less plotting to do.
        ; Small patterns with a long Buffer Length are good for 'steamer' effects.
        ; N.B. Cannot be adjusted whilst patterns are actually onscreen.'
        .BYTE $1C
        ; pulseSpeed: 'Usually if you hold down the button you get a continuous
        ; stream. Setting the Pulse Speed allows you to generate a pulsed stream, as
        ; if you were rapidly pressing and releasing the FIRE button.'
        .BYTE $02
        ; indexForColorBarDisplay: 'The initial index for the color displayed
        ; in the color bar when adjusting the colors for each step.'
        .BYTE $0A
        ; lineWidth: 'Sets the width of the lines produced in Line Mode.'
        .BYTE $07
        ; sequencerSpeed: 'Controls the rate at which sequencer feeds in its data. '
        .BYTE $09
        ; pulseWidth: 'Sets the length of the pulses in a pulsed stream output.
        ; Don't worry about what that means - just get in there and mess with it.'
        .BYTE $01
        ; baseLevel: 'Controls how many 'levels' of pattern are plotted.'
        .BYTE $07
        ; presetColorValuesArray: 'Allows you to set the colour for each of the
        ; seven pattern steps. Set up the colour you want, press RETURN, and the
        ; command offers the next colour along, up to no. 7, then ends. Cannot be
        ; adjusted while patterns being generated.'
        .BYTE BLACK,YELLOW,CYAN,LTBLUE,BLUE,RED,PURPLE,LTRED
        ; trackingActivated: 'Controls whether logic-seeking is used in the
        ; buffer or not. The upshot of this for you is a slightly different feel -
        ; continuous but fragmented when ON, or together-ish bursts when OFF. Try it.'
        .BYTE $00
        ; lineModeActivated: 'A bit like drawing with the Aurora Borealis'
        .BYTE $00
        ; presetIndex: 'This calls in one of the 16 presets, stored Lightsynth
        ; parameters which give different effects. Try them all out io see some uf
        ; the multitude of effects which you cai achieve using the system. Some are
        ; fast, some slow, some pulse, others swirl. Play with them all, try them to
        ; different music.'
        .BYTE $03
        ; currentPatternElement: 'Initial pattern used by this preset.'
        .BYTE $03
        ; currentSymmetrySetting: 'Current symmetry setting.'
        ; Possible values are 0 - 4:
        ; 'NO SYMMETRY     '
        ; 'Y-AXIS SYMMETRY '
        ; 'X-Y SYMMETRY    '
        ; 'X-AXIS SYMMETRY '
        ; 'QUAD SYMMETRY   '
        .BYTE $04
        ; Unused Data.
        .BYTE $00,$FF,$00,$00,$FF,$00,$FF,$00,$FF
```

Listing 4.14: Source code for Preset 13.

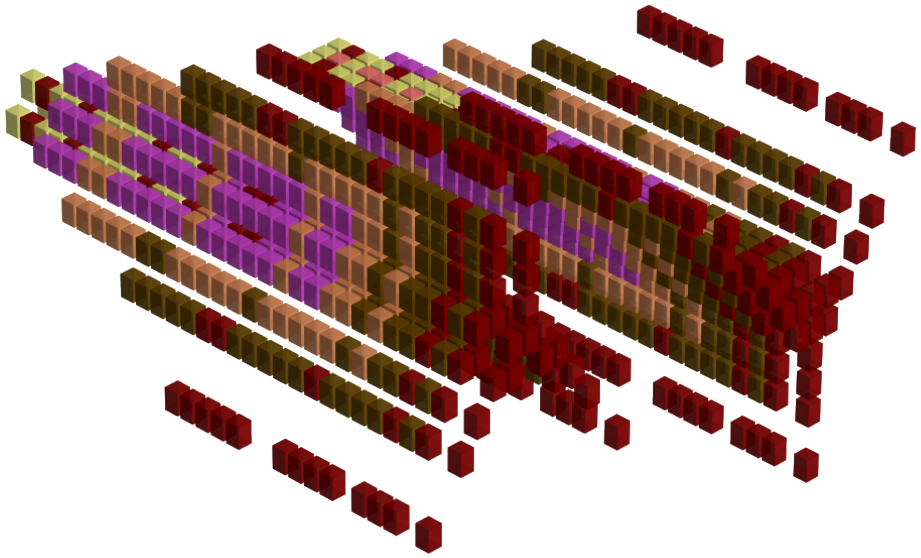


Figure 4.15: Evolution of Preset 14.

```
preset14
        ; unusedPresetByte: Unused Byte
        .BYTE $00
        ; smoothingDelay: 'Because of the time taken to draw larger patterns speed
        ; increase/decrease is not linear. You can adjust the 'compensating delay'
        ; which often smooths out jerky patterns. Can be used just for special FX),
        ; though. Suck it and see.'
        .BYTE $0C
        ; cursorSpeed: 'Gives you a slow or fast little cursor, according to setting.'
        .BYTE $02
        ; bufferLength: 'Larger patterns flow more smoothly with a shorter
        ; Buffer Length - not so many positions are retained so less plotting to do.
        ; Small patterns with a long Buffer Length are good for 'steamer' effects.
        ; N.B. Cannot be adjusted whilst patterns are actually onscreen.'
        .BYTE $2B
        ; pulseSpeed: 'Usually if you hold down the button you get a continuous
        ; stream. Setting the Pulse Speed allows you to generate a pulsed stream, as
        ; if you were rapidly pressing and releasing the FIRE button.'
        .BYTE $01
        ; indexForColorBarDisplay: 'The initial index for the color displayed
        ; in the color bar when adjusting the colors for each step.'
        .BYTE $0A
        ; lineWidth: 'Sets the width of the lines produced in Line Mode.'
        .BYTE $07
        ; sequencerSpeed: 'Controls the rate at which sequencer feeds in its data. '
        .BYTE $08
        ; pulseWidth: 'Sets the length of the pulses in a pulsed stream output.
        ; Don't worry about what that means - just get in there and mess with it.'
        .BYTE $01
        ; baseLevel: 'Controls how many 'levels' of pattern are plotted.'
        .BYTE $07
        ; presetColorValuesArray: 'Allows you to set the colour for each of the
        ; seven pattern steps. Set up the colour you want, press RETURN, and the
        ; command offers the next colour along, up to no. 7, then ends. Cannot be
        ; adjusted while patterns being generated.'
        .BYTE BLACK,RED,BROWN,ORANGE,PURPLE,RED,YELLOW,LTRED
        ; trackingActivated: 'Controls whether logic-seeking is used in the
        ; buffer or not. The upshot of this for you is a slightly different feel -
        ; continuous but fragmented when ON, or together-ish bursts when OFF. Try it.'
        .BYTE $FF
        ; lineModeActivated: 'A bit like drawing with the Aurora Borealis'
        .BYTE $00
        ; presetIndex: 'This calls in one of the 16 presets, stored Lightsynth
        ; parameters which give different effects. Try them all out io see some uf
        ; the multitude of effects which you cai achieve using the system. Some are
        ; fast, some slow, some pulse, others swirl. Play with them all, try them to
        ; different music.'
        .BYTE $04
        ; currentPatternElement: 'Initial pattern used by this preset.'
        .BYTE $04
        ; currentSymmetrySetting: 'Current symmetry setting.'
        ; Possible values are 0 - 4:
        ; 'NO SYMMETRY     '
        ; 'Y-AXIS SYMMETRY '
        ; 'X-Y SYMMETRY    '
        ; 'X-AXIS SYMMETRY '
        ; 'QUAD SYMMETRY   '
        .BYTE $02
        ; Unused Data.
        .BYTE $00,$FF,$00,$00,$FF,$00,$FF,$00,$FF
```

Listing 4.15: Source code for Preset 14.

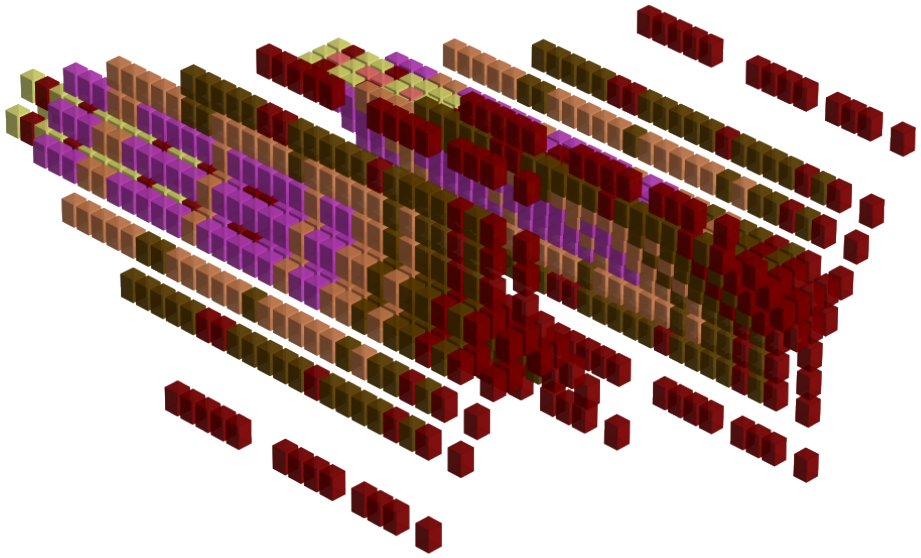
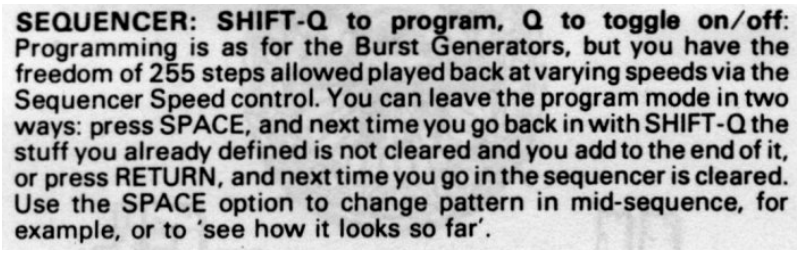


Figure 4.16: Evolution of Preset 15.

```
preset15
        ; unusedPresetByte: Unused Byte
        .BYTE $00
        ; smoothingDelay: 'Because of the time taken to draw larger patterns speed
        ; increase/decrease is not linear. You can adjust the 'compensating delay'
        ; which often smooths out jerky patterns. Can be used just for special FX),
        ; though. Suck it and see.'
        .BYTE $03
        ; cursorSpeed: 'Gives you a slow or fast little cursor, according to setting.'
        .BYTE $01
        ; bufferLength: 'Larger patterns flow more smoothly with a shorter
        ; Buffer Length - not so many positions are retained so less plotting to do.
        ; Small patterns with a long Buffer Length are good for 'steamer' effects.
        ; N.B. Cannot be adjusted whilst patterns are actually onscreen.'
        .BYTE $1F
        ; pulseSpeed: 'Usually if you hold down the button you get a continuous
        ; stream. Setting the Pulse Speed allows you to generate a pulsed stream, as
        ; if you were rapidly pressing and releasing the FIRE button.'
        .BYTE $06
        ; indexForColorBarDisplay: 'The initial index for the color displayed
        ; in the color bar when adjusting the colors for each step.'
        .BYTE $01
        ; lineWidth: 'Sets the width of the lines produced in Line Mode.'
        .BYTE $07
        ; sequencerSpeed: 'Controls the rate at which sequencer feeds in its data. '
        .BYTE $00
        ; pulseWidth: 'Sets the length of the pulses in a pulsed stream output.
        ; Don't worry about what that means - just get in there and mess with it.'
        .BYTE $01
        ; baseLevel: 'Controls how many 'levels' of pattern are plotted.'
        .BYTE $07
        ; presetColorValuesArray: 'Allows you to set the colour for each of the
        ; seven pattern steps. Set up the colour you want, press RETURN, and the
        ; command offers the next colour along, up to no. 7, then ends. Cannot be
        ; adjusted while patterns being generated.'
        .BYTE BLACK,BLUE,RED,PURPLE,GREEN,CYAN,YELLOW,WHITE
        ; trackingActivated: 'Controls whether logic-seeking is used in the
        ; buffer or not. The upshot of this for you is a slightly different feel -
        ; continuous but fragmented when ON, or together-ish bursts when OFF. Try it.'
        .BYTE $FF
        ; lineModeActivated: 'A bit like drawing with the Aurora Borealis'
        .BYTE $00
        ; presetIndex: 'This calls in one of the 16 presets, stored Lightsynth
        ; parameters which give different effects. Try them all out io see some uf
        ; the multitude of effects which you cai achieve using the system. Some are
        ; fast, some slow, some pulse, others swirl. Play with them all, try them to
        ; different music.'
        .BYTE $04
        ; currentPatternElement: 'Initial pattern used by this preset.'
        .BYTE $04
        ; currentSymmetrySetting: 'Current symmetry setting.'
        ; Possible values are 0 - 4:
        ; 'NO SYMMETRY     '
        ; 'Y-AXIS SYMMETRY '
        ; 'X-Y SYMMETRY    '
        ; 'X-AXIS SYMMETRY '
        ; 'QUAD SYMMETRY   '
        .BYTE $04
        ; Unused Data.
        .BYTE $00,$FF,$00,$00,$FF,$00,$FF,$15,$EF
```

Listing 4.16: Source code for Preset 15.

Sensitive Sequencer



SEQUENCER: SHIFT-Q to program, Q to toggle on/off: Programming is as for the Burst Generators, but you have the freedom of 255 steps allowed played back at varying speeds via the Sequencer Speed control. You can leave the program mode in two ways: press SPACE, and next time you go back in with SHIFT-Q the stuff you already defined is not cleared and you add to the end of it, or press RETURN, and next time you go in the sequencer is cleared. Use the SPACE option to change pattern in mid-sequence, for example, or to 'see how it looks so far'.

Figure 5.1: Excerpt from Manual (Part No. LC1982-Vb). Sequencer.

```
MaybeQPressed
        CMP #KEY_Q ; Q pressed?
        BNE MaybeVPressed

        ; Q was pressed. Toggle the sequencer on or off.
        LDA sequencerActive
        BNE TurnSequenceOff
        LDA #SEQUENCER_ACTIVE
        STA currentVariableMode
        JMP ActivateSequencer
        ;Returns

        ;Turn the sequencer off.
TurnSequenceOff
        LDA #$00
        STA sequencerActive
        STA stepsRemainingInSequencerSequence
        JMP DisplaySequencerState
```

Listing 5.1: From `CheckKeyboardInput`.

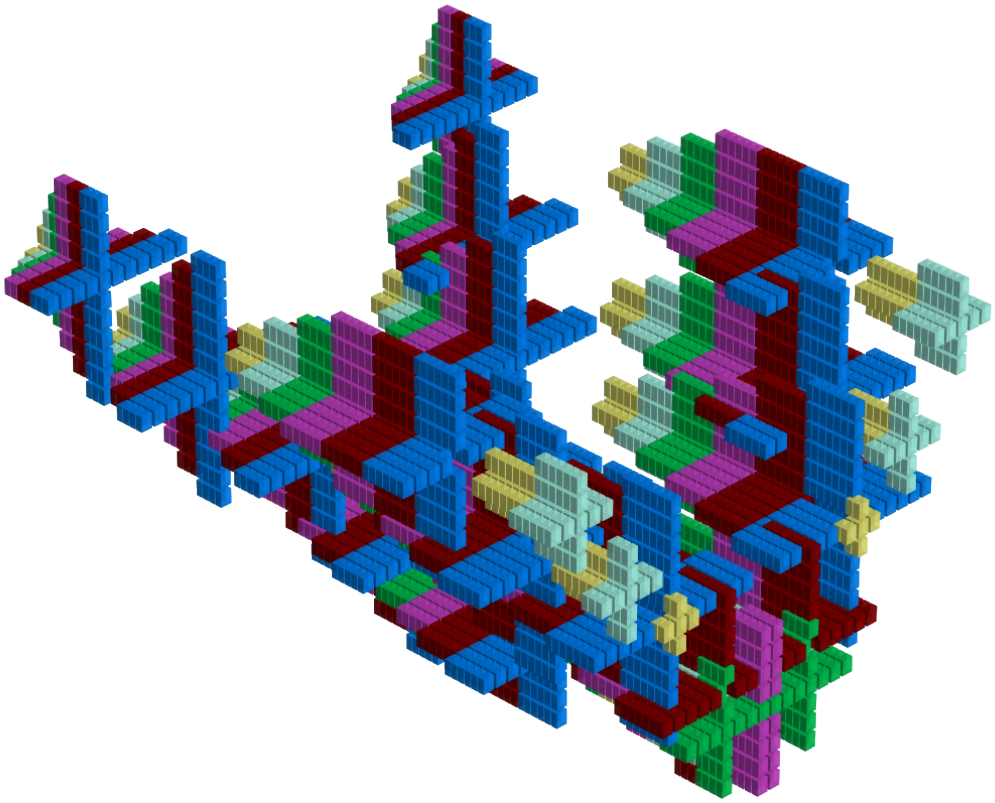


Figure 5.2: Evolution of the Default Sequencer.

```
MULTICROSS      = $06
PULSAR          = $07
startOfSequencerData = $C300
    ; currentSymmetrySetting: 'Current symmetry setting.'
    ; Possible values are 0 - 4:
    ; 'NO SYMMETRY     '
    ; 'Y-AXIS SYMMETRY '
    ; 'X-Y SYMMETRY    '
    ; 'X-AXIS SYMMETRY '
    ; 'QUAD SYMMETRY   '
    .BYTE $01           ; Y-Axis Symmetry

    ; smoothingDelay: 'Because of the time taken to draw larger patterns speed
    ; increase/decrease is not linear. You can adjust the compensating delay
    ; which often smooths out jerky patterns. Can be used just for special FX)
    ; though. Suck it and see.'
    .BYTE $0B

    ; Sequencer Position 1
    .BYTE $04,$04       ; X/Y Co-ordinates: X/Y Position relative to cursor.
    .BYTE PULSAR        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray


    ; Sequencer Position 2
    .BYTE $08,$09       ; X/Y Co-ordinates: X/Y Position relative to cursor.
    .BYTE PULSAR        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray


    ; Sequencer Position 3
    .BYTE $0C,$0C       ; X/Y Co-ordinates: X/Y Position relative to cursor.
    .BYTE PULSAR        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray


    ; Sequencer Position 4
    .BYTE $10,$11       ; X/Y Co-ordinates: X/Y Position relative to cursor.
    .BYTE PULSAR        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray


    ; Sequencer Position 5
    .BYTE $14,$13       ; X/Y Co-ordinates: X/Y Position relative to cursor.
    .BYTE PULSAR        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray


    ; Sequencer Position 6
    .BYTE $17,$13       ; X/Y Co-ordinates: X/Y Position relative to cursor.
    .BYTE PULSAR        ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray


    ; Sequencer Position 7
    .BYTE $FF,$01       ; X/Y Co-ordinates: X/Y Position relative to cursor.
    .BYTE MULTICROSS    ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray


    ; Sequencer Position 8
    .BYTE $41,$FF       ; X/Y Co-ordinates: X/Y Position relative to cursor.
    .BYTE $00           ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray


    ; Sequencer Position 9
    .BYTE $06,$01       ; X/Y Co-ordinates: X/Y Position relative to cursor.
    .BYTE MULTICROSS    ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray

    ...
    ; Sequencer Position 254
    .BYTE $FF,$80       ; X/Y Co-ordinates: X/Y Position relative to cursor.
    .BYTE $EE           ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray


    ; Sequencer Position 255
    .BYTE $FD,$FF       ; X/Y Co-ordinates: X/Y Position relative to cursor.
    .BYTE $FF           ; Index to pattern in pixelXPositionLoPtrArray/pixelXPositionHiPtrArray
```

Listing 5.2: Sequencer definition in `sequencer_data.asm`.

```
ActivateSequencer
        LDA #>startOfSequencerData
        STA currentSequencePtrHi
        LDA #<startOfSequencerData
        STA currentSequencePtrLo
        LDA #$FF
        STA sequencerActive
        LDA shiftPressed
```

```
        AND #$01
        BNE ShiftPressedSoProgramSequencer

        ; Start Playing the Sequencer
        LDA sequencerSpeed
        STA stepsRemainingInSequencerSequence
        LDA #$00
        STA currentVariableMode
        JSR DisplaySequencerState
        RTS

ShiftPressedSoProgramSequencer
        LDA dataFreeForSequencer
        BEQ SetUpNewSequencer
        LDA dataFreeForSequencer
        STA currentDataFree
        LDA prevSequencePtrLo
        STA currentSequencePtrLo
        LDA prevSequencePtrHi
        STA currentSequencePtrHi
        JMP DisplaySequFree
        ;Returns

SetUpNewSequencer
        LDA #$FF
        STA currentDataFree
        LDA currentSymmetrySetting
        LDY #$00
        STA (currentSequencePtrLo),Y
        LDA smoothingDelay
        INY
        STA (currentSequencePtrLo),Y

DisplaySequFree
        JSR ClearLastLineOfScreen

        LDX #$00
SequencerTextLoop
        LDA txtSequFree,X
        STA lastLineBufferPtr,X
        INX
        CPX #$10
        BNE SequencerTextLoop

        JSR WriteLastLineBufferToScreen
        RTS
```

Listing 5.3: `ActivateSequencer` displays the sequencer programming screen (`ShiftPressedSoProgramSequencer`) or activates the sequencer depending on whether the user has pressed 'Shift' in combination with 'Q' or not.

```
MainInterruptHandler
        ; The sequencer is played by the interrupt handler.
        ; Check if it's active.
        LDA stepsRemainingInSequencerSequence
        BEQ b0CFB
        DEC stepsRemainingInSequencerSequence
        BNE b0CFB

        ; If the sequencer is active we'll end up here and
        ; load the sequencer data so that it can be played.
        LDA sequencerSpeed
        STA stepsRemainingInSequencerSequence
        JSR LoadDataForSequencer
```

Listing 5.4: `ActivateSequencer` displays the sequencer programming screen (`ShiftPressedSoProgramSequencer`) or activates the sequencer depending on whether the user has pressed 'Shift' in combination with 'Q' or not.

```
LoadDataForSequencer
        INC currentStepCount
```

```
        LDA currentStepCount
        CMP bufferLength
        BNE b1992

        LDA #$00
        STA currentStepCount
b1992   TAX
        LDA currentIndexForCurrentStepArray,X
        CMP #$FF
        BEQ LoadValuesFromSequencerData

        LDA shouldDrawCursor
        AND trackingActivated
        BEQ MoveToNextPositionInSequencer
        TAX
        LDA currentIndexForCurrentStepArray,X
        CMP #$FF
        BNE MoveToNextPositionInSequencer

LoadValuesFromSequencerData
        LDY #$02
        LDA (currentSequencePtrLo),Y
        CMP #$C0
        BEQ MoveToNextPositionInSequencer

        LDA baseLevel
        STA currentIndexForCurrentStepArray,X

        LDA startOfSequencerData + $01
        STA initialFramesRemainingToNextPaintForStep,X
        STA framesRemainingToNextPaintForStep,X

        LDA startOfSequencerData
        STA symmetrySettingForStepCount,X

        LDY #$02
        LDA (currentSequencePtrLo),Y
        STA pixelXPositionArray,X
        INY
        LDA (currentSequencePtrLo),Y
        STA pixelYPositionArray,X
        INY
        LDA (currentSequencePtrLo),Y
        STA patternIndexArray,X

MoveToNextPositionInSequencer
        LDA currentSequencePtrLo
        CLC
        ADC #$03
        STA currentSequencePtrLo
        LDA currentSequencePtrHi
        ADC #$00
        STA currentSequencePtrHi
        LDY #$02
        LDA (currentSequencePtrLo),Y
        CMP #$FF
        BEQ ResetSequencerToStart
        RTS

ResetSequencerToStart
        LDA #<startOfSequencerData
        STA currentSequencePtrLo
        LDA #>startOfSequencerData
        STA currentSequencePtrHi
        RTS
```

Listing 5.5: `ActivateSequencer` displays the sequencer programming screen (`ShiftPressedSoProgramSequencer`) or activates the sequencer depending on whether the user has pressed 'Shift' in combination with 'Q' or not.

5.0.1 Sequencer Speed

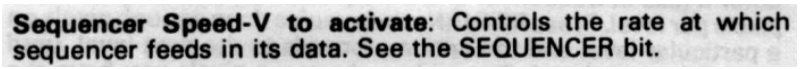


Figure 5.3: Excerpt from Manual (Part No. LC1982-Vb). Sequencer Speed.

```
SEQUENCER_SPEED      = $07
MaybeVPressed
        CMP #KEY_V ; V pressed?
        BNE MaybeOPressed

        ; V pressed.
        ; Sequencer Speed V to activate: Controls the rate at which
    sequencer
        ; feeds in its data. See the SEQUENCER bit.
        LDA #SEQUENCER_SPEED
        STA currentVariableMode
        RTS
```

Listing 5.6: From `CheckKeyboardInput`.

```
UpdateVariableDisplay
        LDA #>SCREEN_RAM + $03D0
        STA colorBarColorRamHiPtr
        LDA #<SCREEN_RAM + $03D0
        STA colorBarColorRamLoPtr

        LDX currentVariableMode
        LDA lastKeyPressed
        CMP #$2C ; > pressed?
        BNE MaybeLeftArrowPressed

        ; > pressed, increase the value bar and write
        ; it to the approiate place in presetValueArray
        INC presetValueArray,X
        LDA presetValueArray,X
        ; Make sure we don't exceed the max value.
        CMP maxValueForPresetValueArray,X
        BNE MaybeInColorMode
        DEC presetValueArray,X
        JMP MaybeInColorMode
```

Listing 5.7: From `CheckKeyboardInputForActiveVariable`. Pressing the ¡ and ¿ keys increments and decrements the value in presetValueArray pointed to by `X` i.e. `currentVariableMode`.

```
; This is where the presets get loaded to. It represents
; the data structure for the presets.
; currentVariableMode is an index into this data structure
; when the user adjusts settings.
```

```
presetValueArray
unusedPresetByte         .BYTE $00
smoothingDelay           .BYTE $0C
cursorSpeed              .BYTE $02
bufferLength             .BYTE $1F
pulseSpeed               .BYTE $01
indexForColorBarDisplay  .BYTE $01
lineWidth                .BYTE $07
sequencerSpeed           .BYTE $04 ; <-- Sequencer Speed is here at
    position \icode{\$07}.
pulseWidth               .BYTE $01
baseLevel                .BYTE $07
presetColorValuesArray   .BYTE BLACK,BLUE,RED,PURPLE,GREEN,CYAN,YELLOW,
    WHITE
trackingActivated        .BYTE $FF
lineModeActivated        .BYTE $00
presetIndex             .BYTE $05
```

Listing 5.8: From `ActivateSequencer`.

```
ActivateSequencer
        LDA #>startOfSequencerData
        STA currentSequencePtrHi
        LDA #<startOfSequencerData
        STA currentSequencePtrLo
        LDA #$FF
        STA sequencerActive
        LDA shiftPressed
        AND #$01
        BNE ShiftPressedSoProgramSequencer

        ; Start Playing the Sequencer.
        ; Load the sequencer speed.
        LDA sequencerSpeed
        STA stepsRemainingInSequencerSequence
        LDA #$00
        STA currentVariableMode
        JSR DisplaySequencerState
        RTS
```

Listing 5.9: From `ActivateSequencer`. `sequencerSpeed` is loaded to `stepsRemainingInSequencerSequence`.

```
MainInterruptHandler
        ; The sequencer is played by the interrupt handler.
        ; Check if it's active.
        LDA stepsRemainingInSequencerSequence
        BEQ SequencerNotActiveCheckJoystickInput
        DEC stepsRemainingInSequencerSequence
        BNE SequencerNotActiveCheckJoystickInput

        ; If the sequencer is active we'll end up here and
        ; load the sequencer data so that it can be played.
```

```
        LDA sequencerSpeed
        STA stepsRemainingInSequencerSequence
        JSR LoadDataForSequencer

SequencerNotActiveCheckJoystickInput
        DEC countStepsBeforeCheckingJoystickInput
        BEQ b0D03
        JMP JumpToCheckKeyboardInput
        ;Returns?
```

Listing 5.10: From `MainInterruptHandler`. `stepsRemainingInSequencerSequence` tells us whether to 'play' a step in the sequencer at each raster interrupt.

Pulse Speed

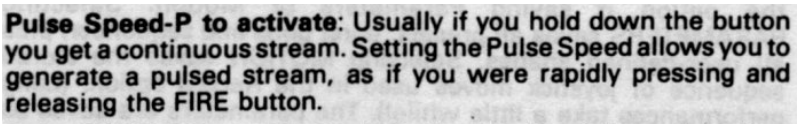


Figure 6.1: Excerpt from Manual (Part No. LC1982-Vb). Pulse Speed.

```
MaybePPressed
        CMP #KEY_P ; P pressed
        BNE MaybeHPressed

        ; P pressed.
        ; Pulse Speed P to activate: Usually if you hold down the
    button you
        ; get a continuous stream. Setting the Pulse Speed allows you
    to
        ; generate a pulsed stream, as if you were rapidly pressing and
        ; releasing the FIRE button.
        LDA #PULSE_SPEED
        STA currentVariableMode
        RTS
```

Listing 6.1: From `CheckKeyboardInput`.

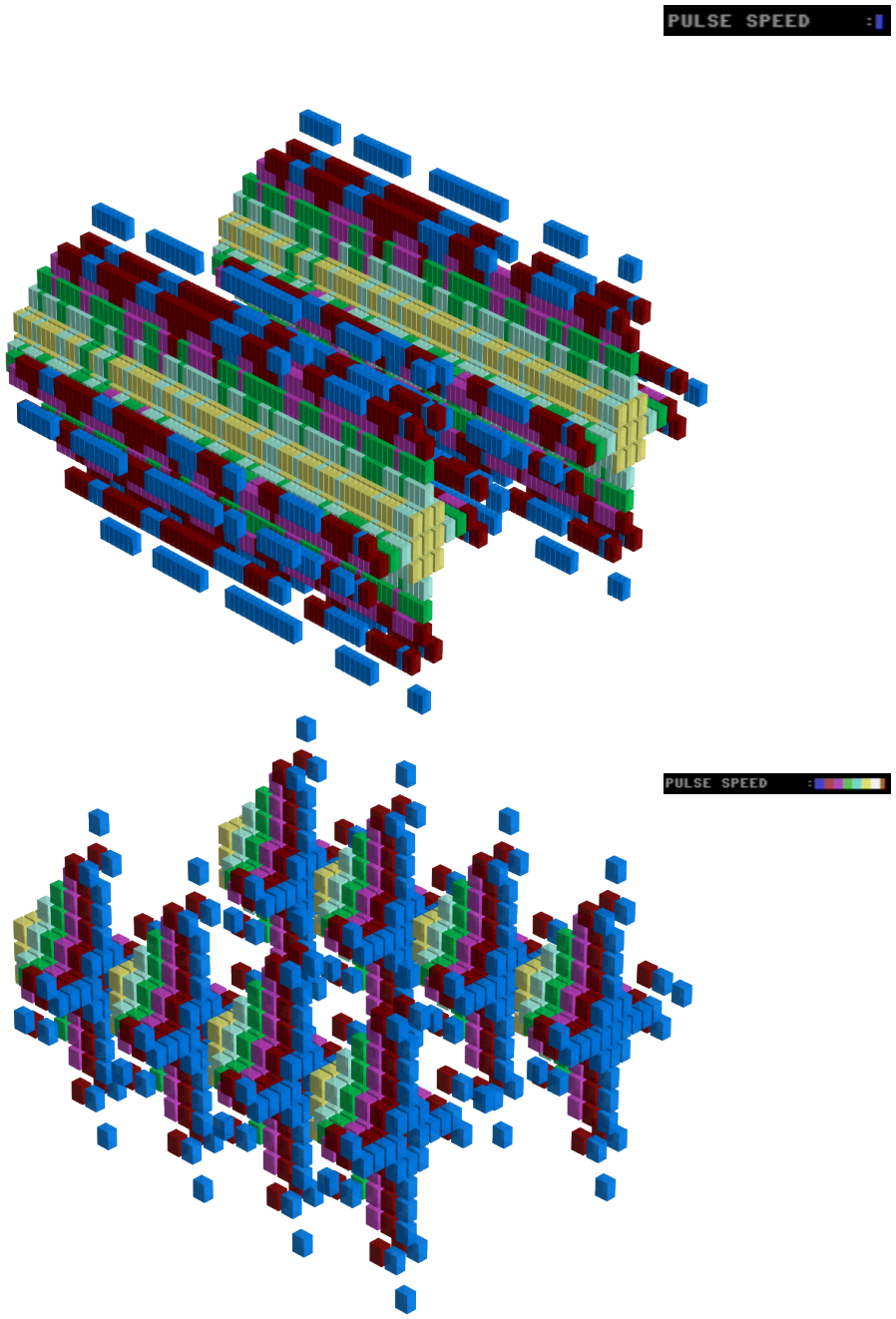


Figure 6.2: Effect of low and high values for Pulse Speed

```
UpdateVariableDisplay
        LDA #>SCREEN_RAM + $03D0
        STA colorBarColorRamHiPtr
        LDA #<SCREEN_RAM + $03D0
        STA colorBarColorRamLoPtr

        LDX currentVariableMode
        LDA lastKeyPressed
        CMP #$2C ; > pressed?
        BNE MaybeLeftArrowPressed

        ; > pressed, increase the value bar and write
        ; it to the approiate place in presetValueArray
        INC presetValueArray,X
        LDA presetValueArray,X
        ; Make sure we don't exceed the max value.
        CMP maxValueForPresetValueArray,X
        BNE MaybeInColorMode
        DEC presetValueArray,X
        JMP MaybeInColorMode
```

Listing 6.2: From `CheckKeyboardInputForActiveVariable`. Pressing the ¡ and ¿ keys increments and decrements the value in presetValueArray pointed to by `X` i.e. `currentVariableMode`.

```
; This is where the presets get loaded to. It represents
; the data structure for the presets.
; currentVariableMode is an index into this data structure
; when the user adjusts settings.
presetValueArray
unusedPresetByte        .BYTE $00
smoothingDelay          .BYTE $0C
cursorSpeed             .BYTE $02
bufferLength            .BYTE $1F
pulseSpeed              .BYTE $01; <-- Pulse Speed is here at position
    \icode{\$04}.
indexForColorBarDisplay .BYTE $01
lineWidth               .BYTE $07
sequencerSpeed          .BYTE $04
pulseWidth              .BYTE $01
baseLevel               .BYTE $07
presetColorValuesArray  .BYTE BLACK,BLUE,RED,PURPLE,GREEN,CYAN,YELLOW,
    WHITE
trackingActivated       .BYTE $FF
lineModeActivated       .BYTE $00
presetIndex            .BYTE $05
```

Listing 6.3: From `ActivateSequencer`.

```
DecrementPulseSpeedCounter
        DEC currentPulseSpeedCounter
        BEQ RefreshPulseSpeed
        JMP DrawCursorAndReturnFromInterrupt
        ; Returns
```

```
RefreshPulseSpeed
        LDA pulseSpeed
        STA currentPulseSpeedCounter
        LDA pulseWidth
        STA currentPulseWidth
```

Listing 6.4: From `MainInterruptHandler`.

Pulse Width

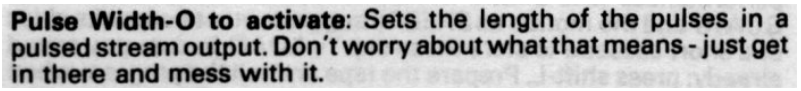


Figure 7.1: Excerpt from Manual (Part No. LC1982-Vb). Pulse Width.

```
MaybeOPressed
        CMP #KEY_O ; O pressed.
        BNE MaybeAsteriskPressed

        ; O pressed.
        ; Pulse Width : Sets the length of the pulses in a pulsed
        ; stream output. Don't worry about what that means - just get
    in there
        ; and mess with it.
        LDA #PULSE_WIDTH
        STA currentVariableMode
        RTS
```

Listing 7.1: From `CheckKeyboardInput`.

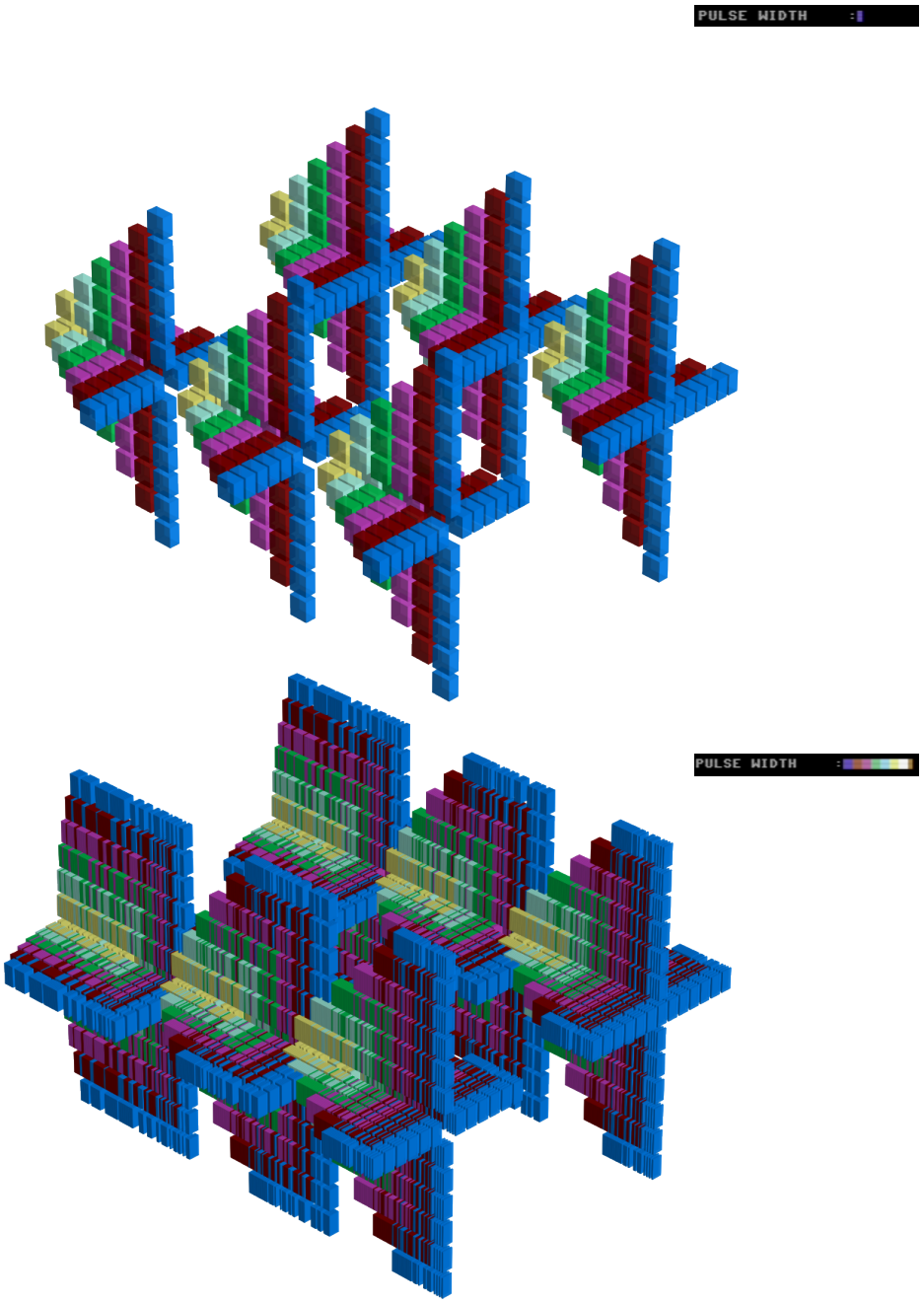


Figure 7.2: Effect of low and high values for Pulse Width

```
UpdateVariableDisplay
        LDA #>SCREEN_RAM + $03D0
        STA colorBarColorRamHiPtr
        LDA #<SCREEN_RAM + $03D0
        STA colorBarColorRamLoPtr

        LDX currentVariableMode
        LDA lastKeyPressed
        CMP #$2C ; > pressed?
        BNE MaybeLeftArrowPressed

        ; > pressed, increase the value bar and write
        ; it to the approiate place in presetValueArray
        INC presetValueArray,X
        LDA presetValueArray,X
        ; Make sure we don't exceed the max value.
        CMP maxValueForPresetValueArray,X
        BNE MaybeInColorMode
        DEC presetValueArray,X
        JMP MaybeInColorMode
```

Listing 7.2: From `CheckKeyboardInputForActiveVariable`. Pressing the ¡ and ¿ keys increments and decrements the value in presetValueArray pointed to by `X` i.e. `currentVariableMode`.

```
; This is where the presets get loaded to. It represents
; the data structure for the presets.
; currentVariableMode is an index into this data structure
; when the user adjusts settings.
presetValueArray
unusedPresetByte        .BYTE $00
smoothingDelay          .BYTE $0C
cursorWidth             .BYTE $02
bufferLength            .BYTE $1F
pulseWidth              .BYTE $01; <-- Pulse Width is here at position
    \icode{\$04}.
indexForColorBarDisplay .BYTE $01
lineWidth               .BYTE $07
sequencerWidth          .BYTE $04
pulseWidth              .BYTE $01
baseLevel               .BYTE $07
presetColorValuesArray  .BYTE BLACK,BLUE,RED,PURPLE,GREEN,CYAN,YELLOW,
    WHITE
trackingActivated       .BYTE $FF
lineModeActivated       .BYTE $00
presetIndex            .BYTE $05
```

Listing 7.3: From `ActivateSequencer`.

```
DecrementPulseWidthCounter
        DEC currentPulseWidthCounter
        BEQ RefreshPulseWidth
        JMP DrawCursorAndReturnFromInterrupt
        ; Returns
```

```
RefreshPulseWidth
        LDA pulseWidth
        STA currentPulseWidthCounter
        LDA pulseWidth
        STA currentPulseWidth
```

Listing 7.4: From `MainInterruptHandler`.

Line Mode

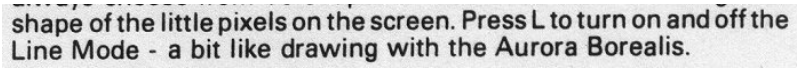

shape of the little pixels on the screen. Press L to turn on and off the Line Mode - a bit like drawing with the Aurora Borealis.

Figure 8.1: Excerpt from Manual (Part No. LC1982-Vb). Line Width.

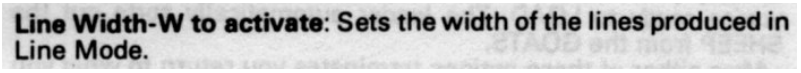

Line Width-W to activate: Sets the width of the lines produced in Line Mode.

Figure 8.2: Excerpt from Manual (Part No. LC1982-Vb). Line Width.

```
        ; 'L' pressed. Turn line mode on or off.
JustLPressed
        LDA lineModeActivated
        EOR #$01
        STA lineModeActivated
        ASL
        ASL
        ASL
        ASL
        TAY

        ; Briefly display the new linemode setting on the bottom of the screen.
        JSR ClearLastLineOfScreen
        LDX #$00
b1043   LDA lineModeSettingDescriptions,Y
        STA lastLineBufferPtr,X
        INY
        INX
        CPX #$10
        BNE b1043
        JMP WriteLastLineBufferToScreen
        ; Returns
```

Listing 8.1: From `CheckKeyboardInput`.

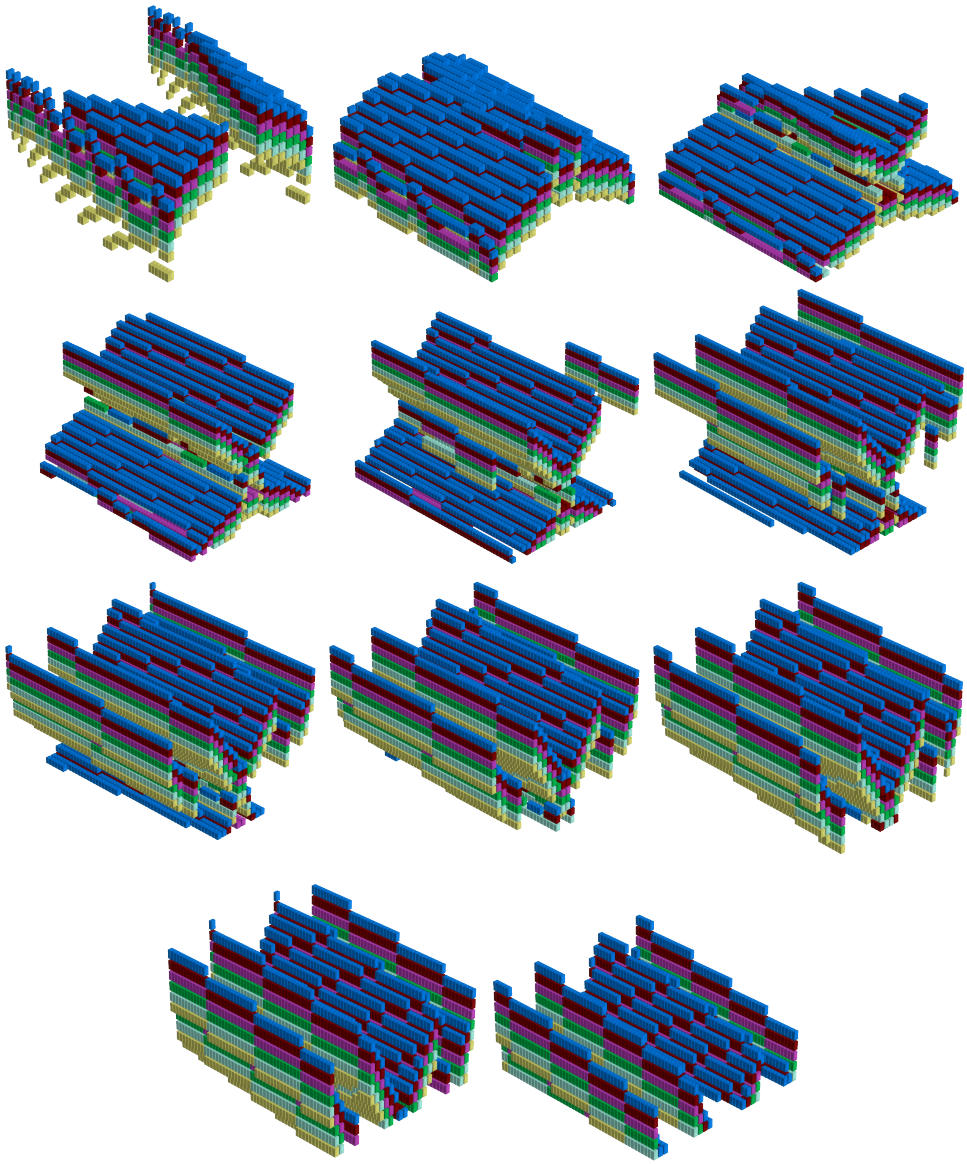


Figure 8.3: Line Mode with Pulse Width at Maximum

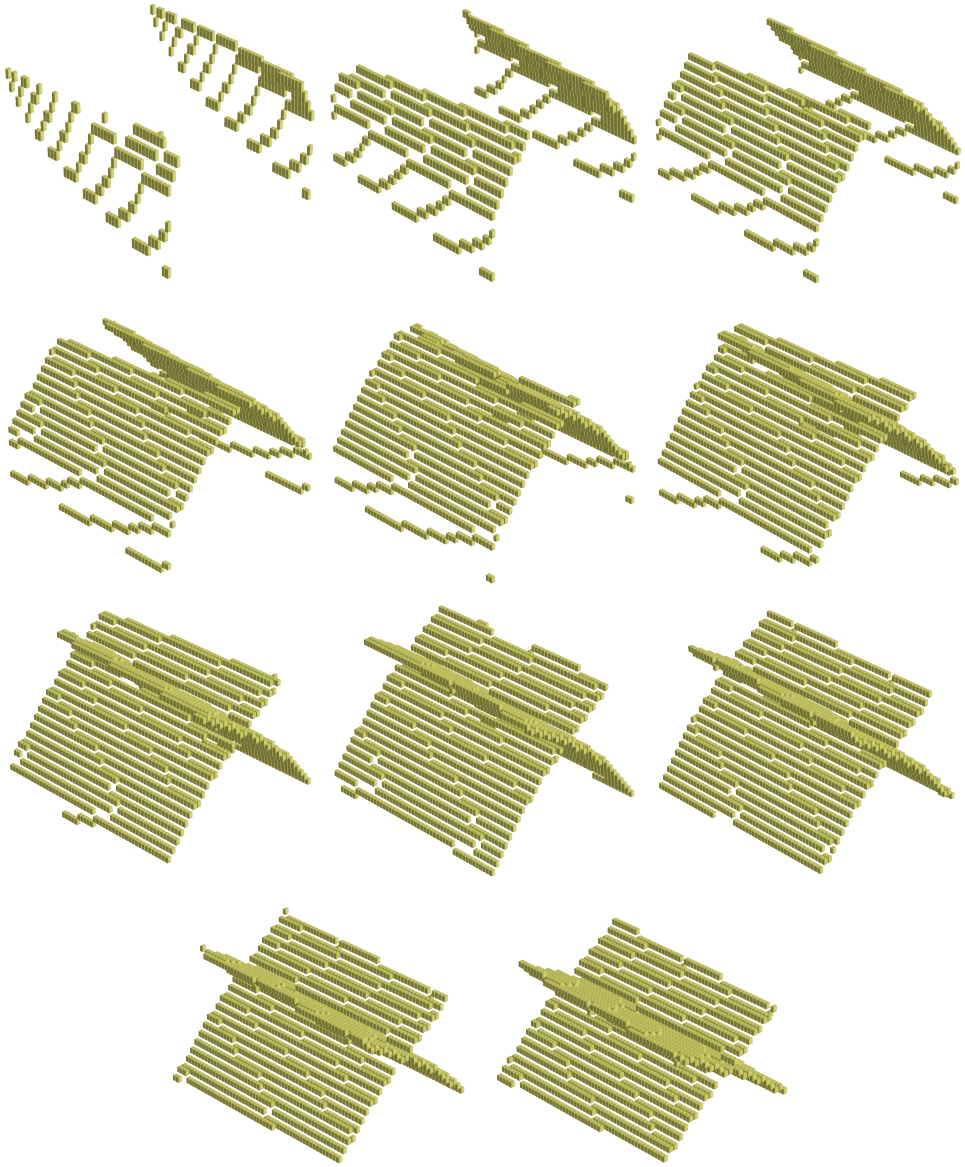


Figure 8.4: Line Mode with Pulse Width at Minimum

```
        ; Line Mode Active
        LDA #$19
        SEC
        SBC colorRAMLineTableIndex
        ORA #$80
        STA currentIndexForCurrentStepArray,X
```

Listing 8.2: From `MainInterruptHandler`.

```
        ; Line Mode sets the top bit of currentIndexToColorValues
        LDA currentIndexToColorValues
        AND #$80
        BNE PaintLineModeAndLoop
        ...
PaintLineModeAndLoop
        ; Loops back to MainPaintLoop
        JMP PaintLineMode
```

Listing 8.3: From `MainPaintLoop`.

```
PaintLineMode
        LDA currentIndexToColorValues
        AND #$7F
        STA offsetForYPos
        LDA #$19
        SEC
        SBC offsetForYPos
        STA pixelYPositionZP
        DEC pixelYPositionZP
        LDA #$00
        STA currentIndexToColorValues
        LDA #$01
        STA skipPixel
        JSR PaintPixelForCurrentSymmetry
        INC pixelYPositionZP
        LDA #$00
        STA skipPixel

        LDA lineWidth
        EOR #$07
        STA currentIndexToColorValues
LineModeLoop
        JSR PaintPixelForCurrentSymmetry
        INC pixelYPositionZP
        INC currentIndexToColorValues
        LDA currentIndexToColorValues
        CMP #$08
        BNE ResetLineModeColorValue
        JMP CleanUpAndExitLineModePaint

        INC currentIndexToColorValues
ResetLineModeColorValue
        STA currentIndexToColorValues
```

```
        LDA pixelYPositionZP
        CMP #$19
        BNE LineModeLoop

CleanUpAndExitLineModePaint
        LDX currentBufferLength
        DEC currentIndexForCurrentStepArray,X
        LDA currentIndexForCurrentStepArray,X
        CMP #$80
        BEQ ResetIndexAndExitLineModePaint
        JMP MainPaintLoop

ResetIndexAndExitLineModePaint
        LDA #$FF
        STA currentIndexForCurrentStepArray,X
        STX shouldDrawCursor
        JMP MainPaintLoop
```

Listing 8.4: From `PaintLineMode`.

Smoothing Delay

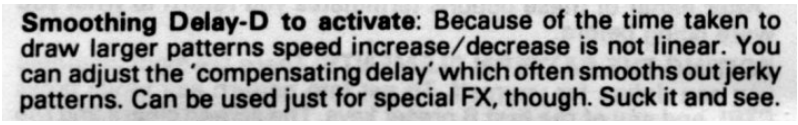


Figure 9.1: Excerpt from Manual (Part No. LC1982-Vb). Smoothing Delay.

```
MaybeDPressed
  CMP #KEY_D ; 'D' pressed?
  BNE MaybeCPressed

  ; Smoothing Delay, D to activate: Because of the time taken to draw
  ; larger patterns speed increase/decrease is not linear. You can
    adjust
  ; the 'compensating delay' which often smooths out jerky patterns.
    Can
  ; be used just for special FX, though. Suck it and see.
  LDA #SMOOTHING_DELAY
  STA currentVariableMode
  RTS
```

Listing 9.1: From `CheckKeyboardInput`.

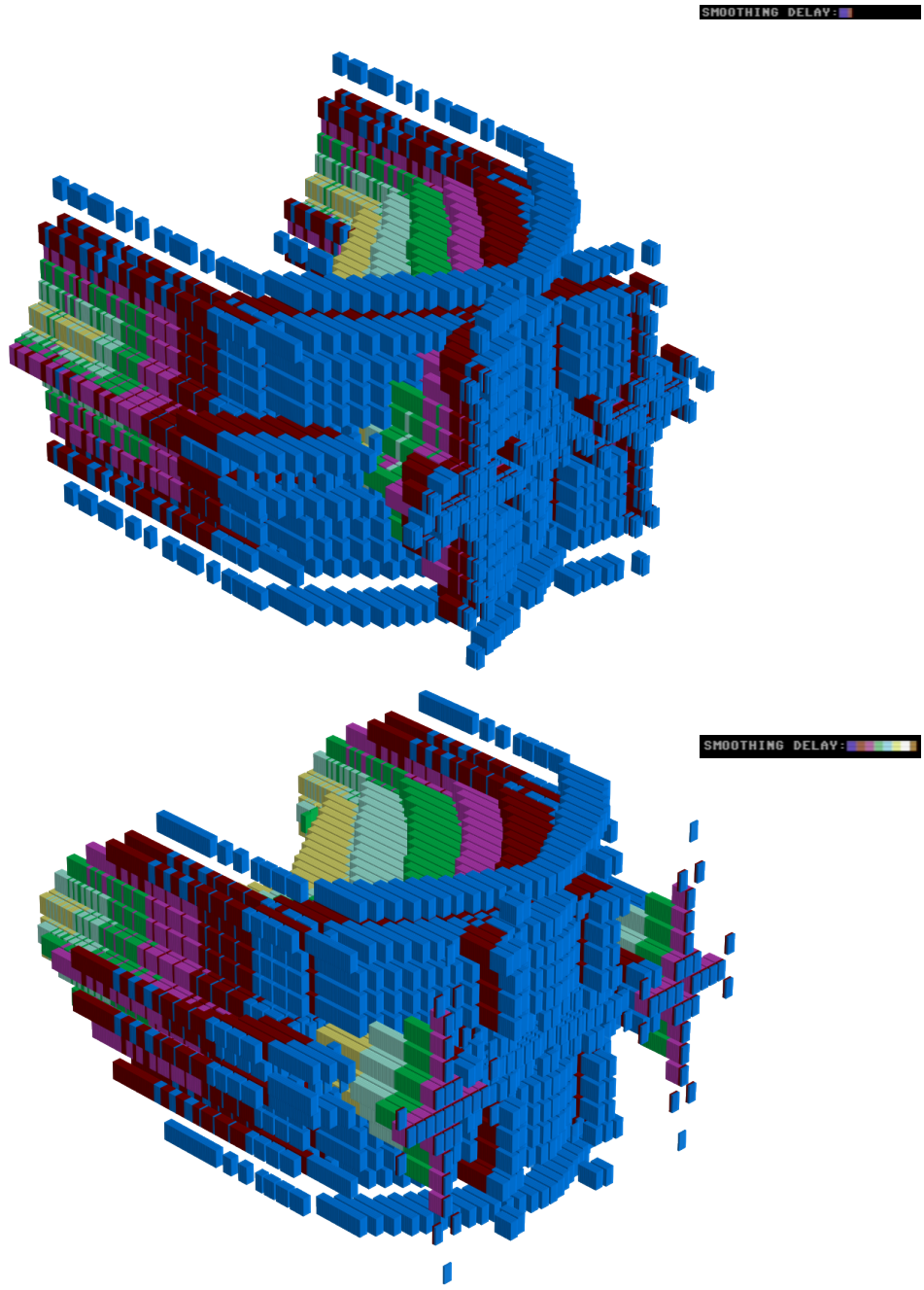


Figure 9.2: Effect of low and high values for Smoothing Delay

```
ApplySmoothingDelay
        LDA smoothingDelay
        STA initialFramesRemainingToNextPaintForStep,X
        STA framesRemainingToNextPaintForStep,X
```

Listing 9.2: From `MainInterruptHandler`.

```
initialFramesRemainingToNextPaintForStep
        .BYTE $00,$00,$00,$00,$00,$00,$00,$00
        .BYTE $00,$00,$00,$00,$00,$00,$00,$00
        .BYTE $00,$00,$00,$00,$00,$00,$00,$00
        .BYTE $00,$00,$FF,$FF,$FF,$FF,$FF,$FF
        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
framesRemainingToNextPaintForStep
        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
        .BYTE $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
```

Listing 9.3: Definition of `framesRemainingToNextPaintForStep` and its counterpart `initialFramesRemainingToNextPaintForStep`..

```
ShouldDoAPaint
        STA currentIndexToColorValues
        DEC framesRemainingToNextPaintForStep,X
        BNE GoBackToStartOfLoop

        ; Actually paint some pixels to the screen.

        ; Reset the delay for this step.
        LDA initialFramesRemainingToNextPaintForStep,X
        STA framesRemainingToNextPaintForStep,X

        ; Get the x and y positions for this pixel.
        LDA pixelXPositionArray,X
        STA pixelXPositionZP
        LDA pixelYPositionArray,X
        STA pixelYPositionZP

        LDA patternIndexArray,X
        STA presetIndex

        LDA symmetrySettingForStepCount,X
        STA currentSymmetrySettingForStep

        LDA currentIndexToColorValues
```

```
        AND #$80
        BNE ResetAndGoBackToStartOfLoop

        TXA
        PHA
        JSR LoopThroughPixelsAndPaint
        PLA
        TAX
```

Listing 9.4: From `MainPaintLoop`.

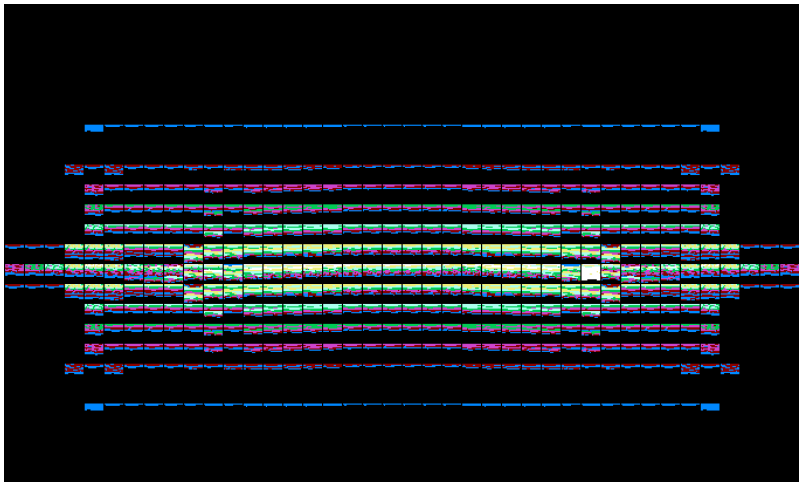


Figure 9.3: 

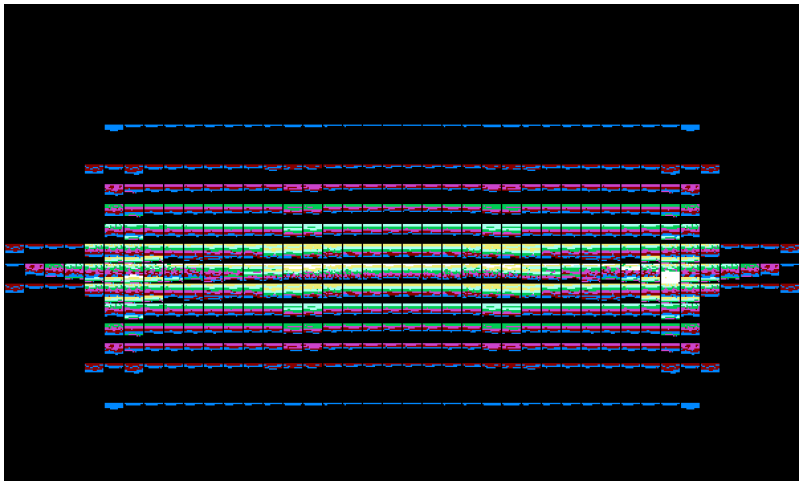
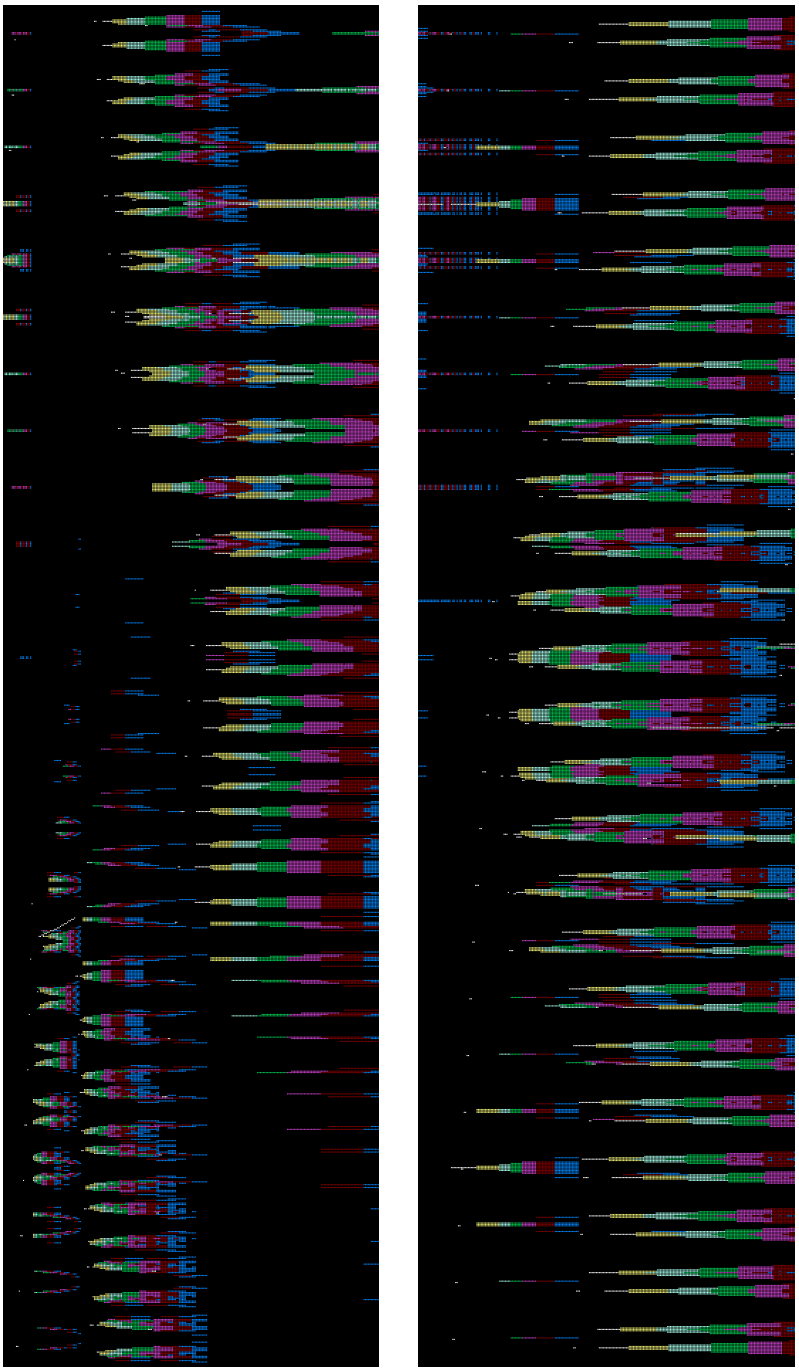
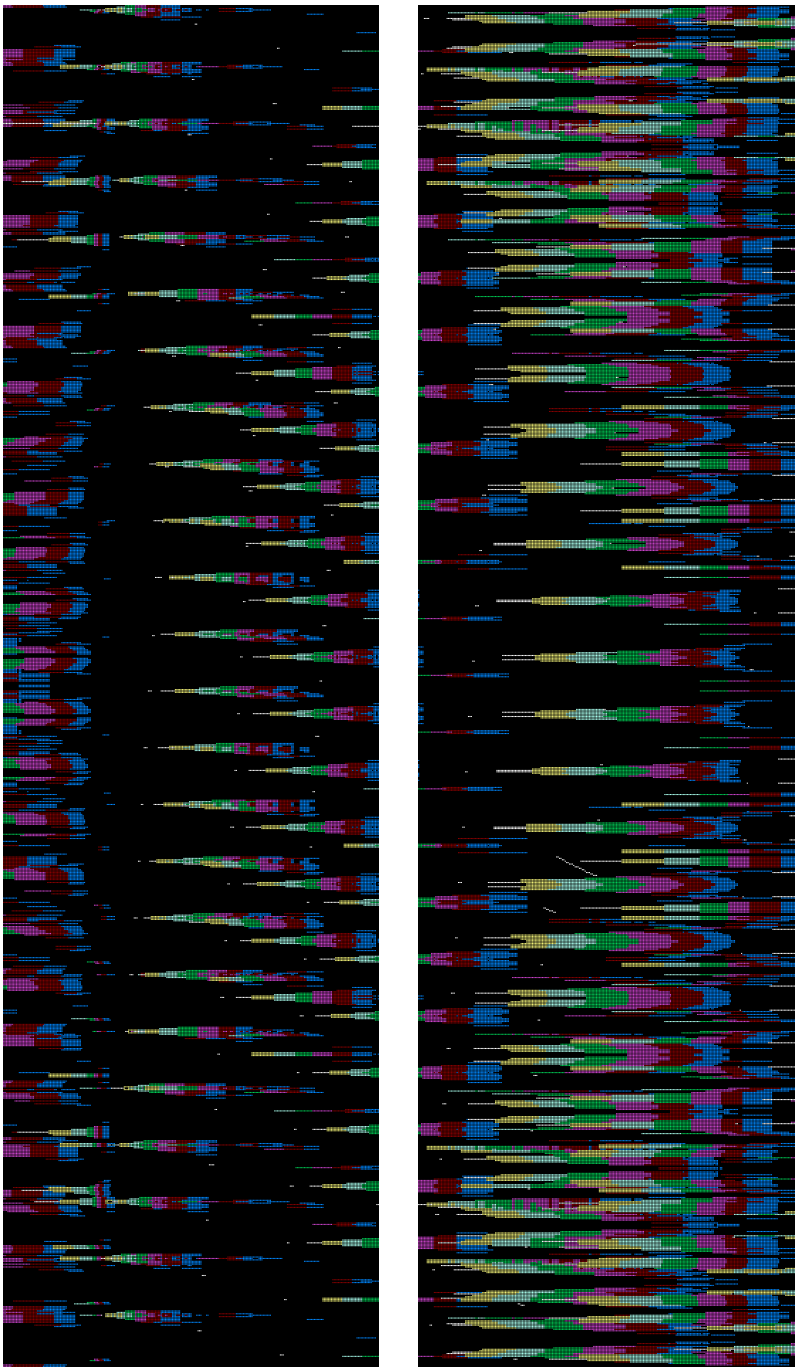


Figure 9.4: 

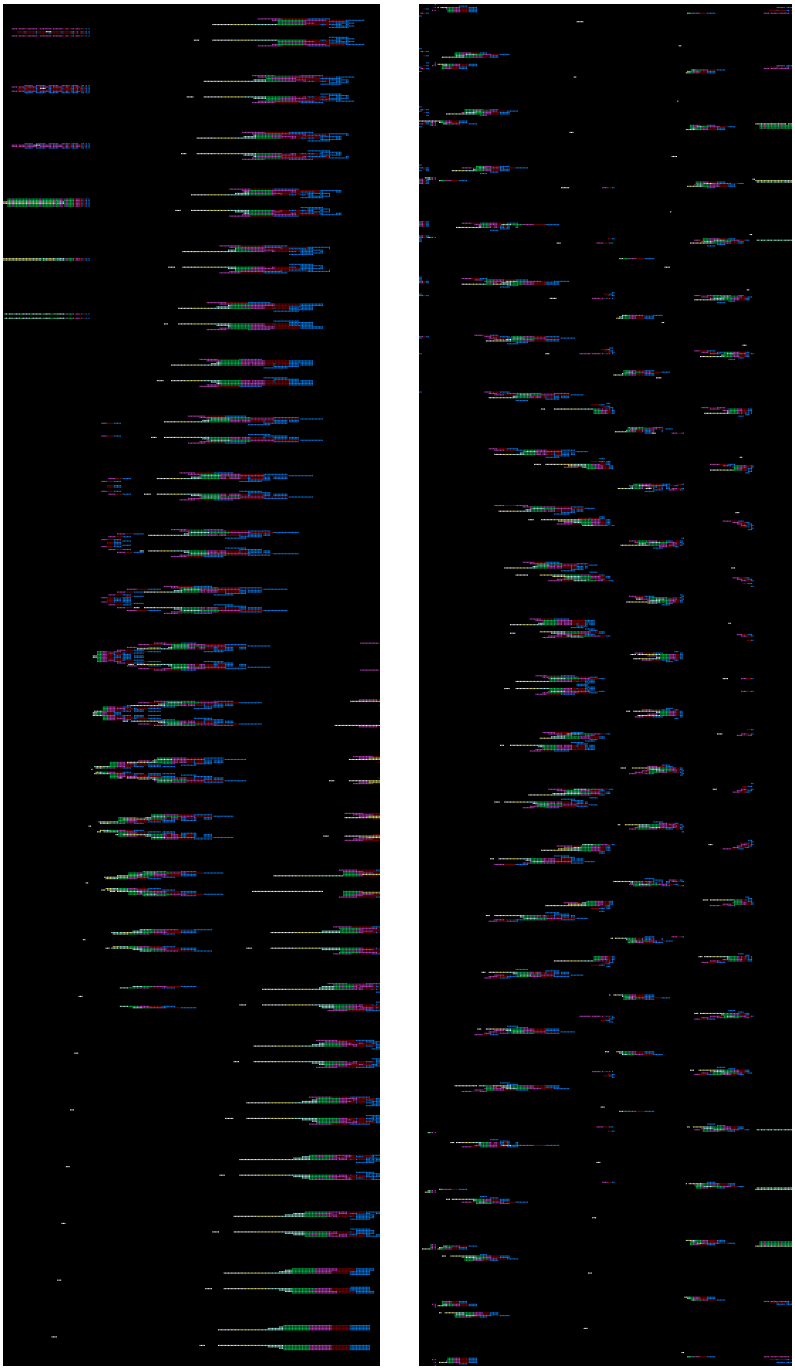
Irksome Evolutions



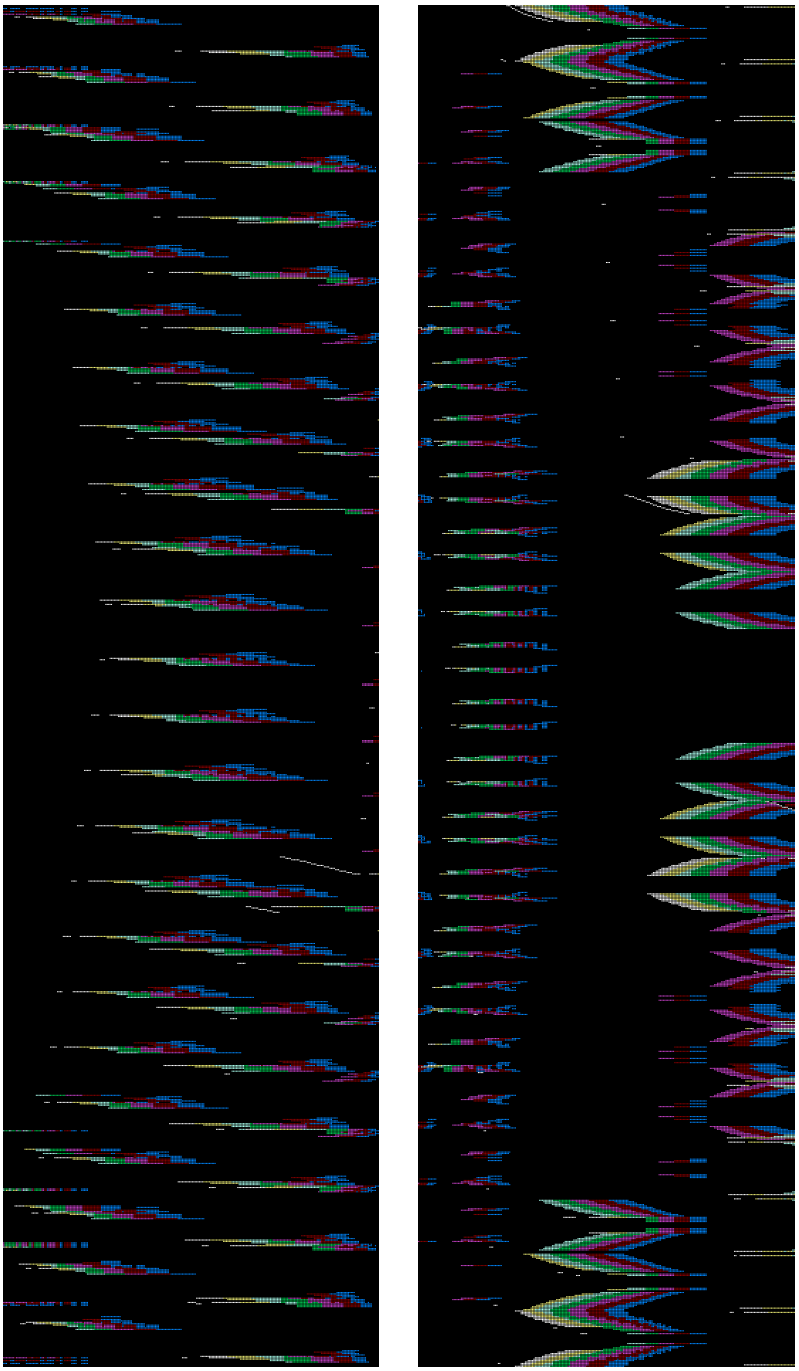
Star One : Y-Axis Symmetry on left, X-Y axis Symmetry on Right



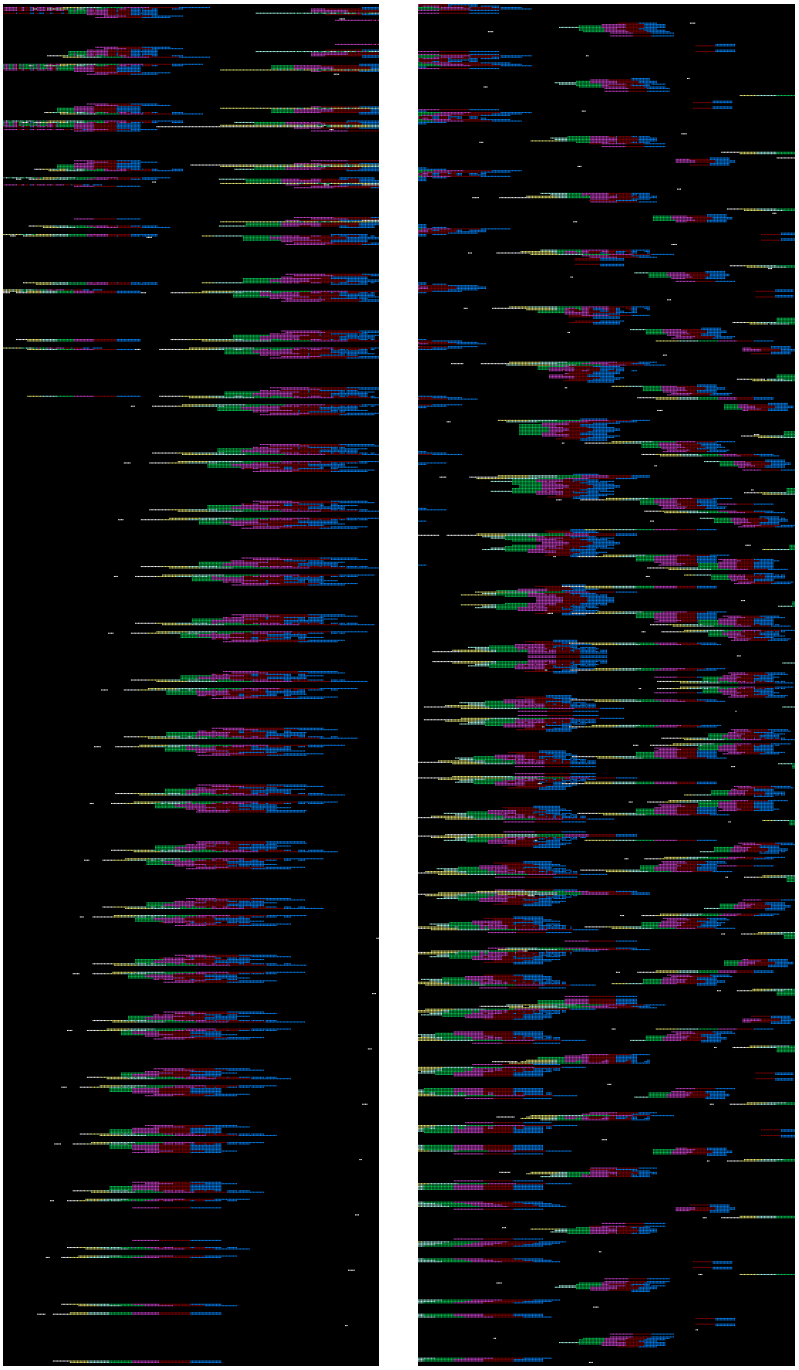
Star One : X-Axis Symmetry on left, Quad Symmetry on Right



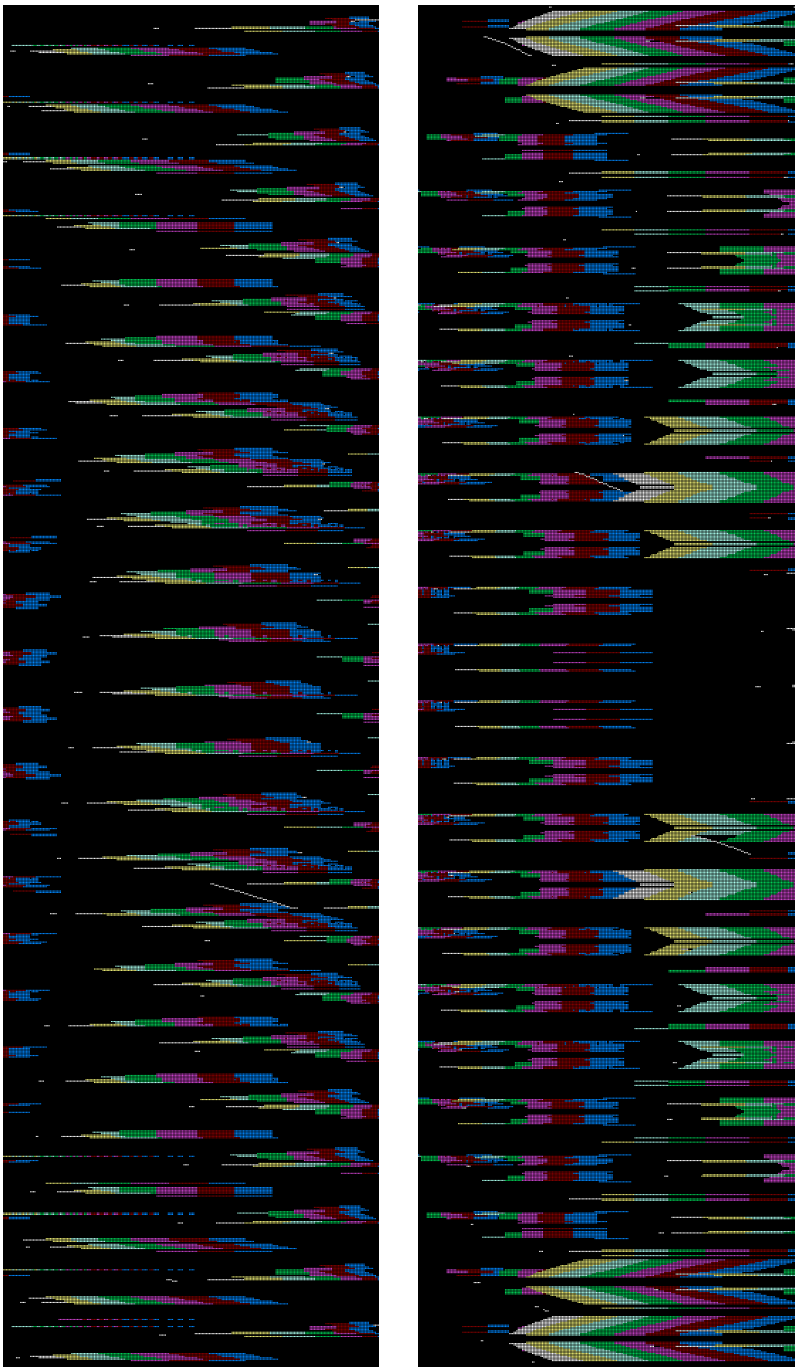
The Twist : Y-Axis Symmetry on left, X-Y axis Symmetry on Right



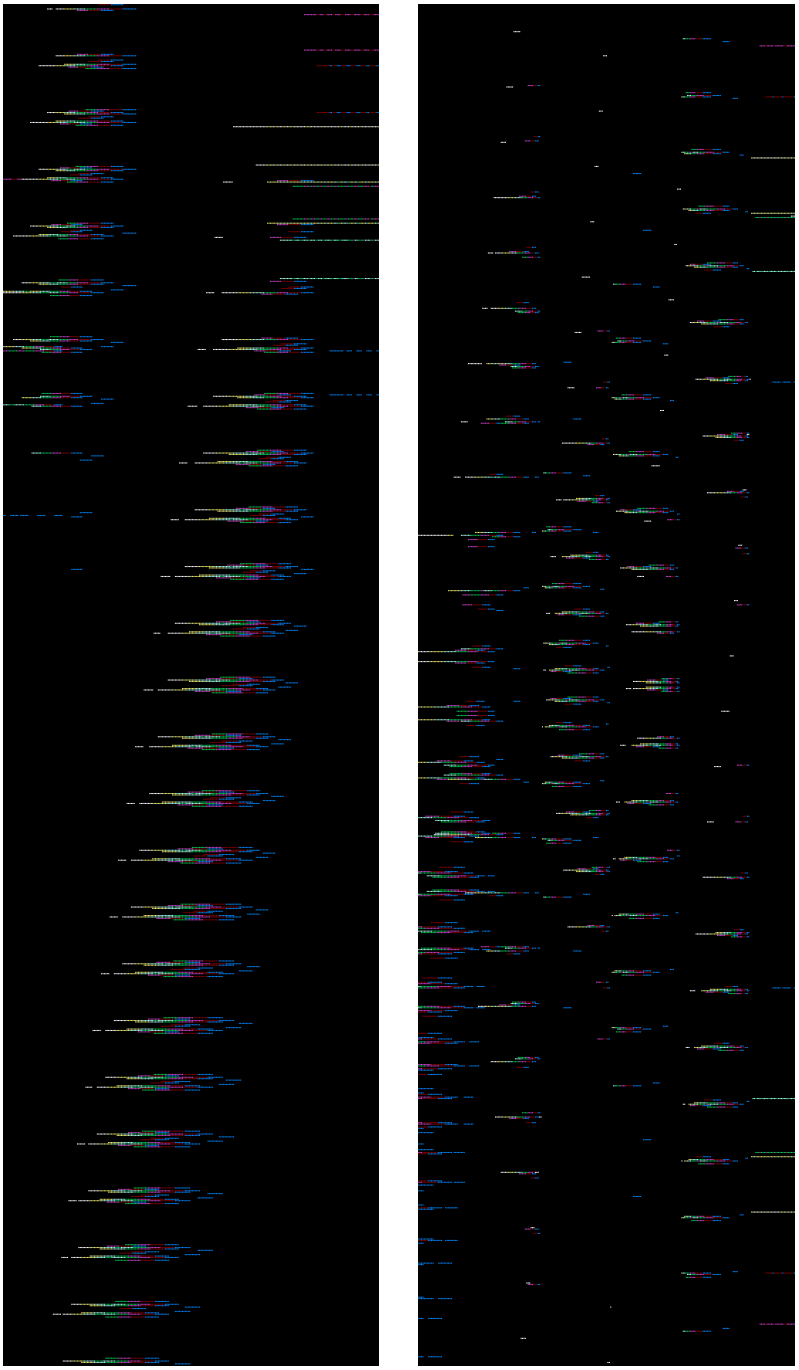
The Twist : X-Axis Symmetry on left, Quad Symmetry on Right



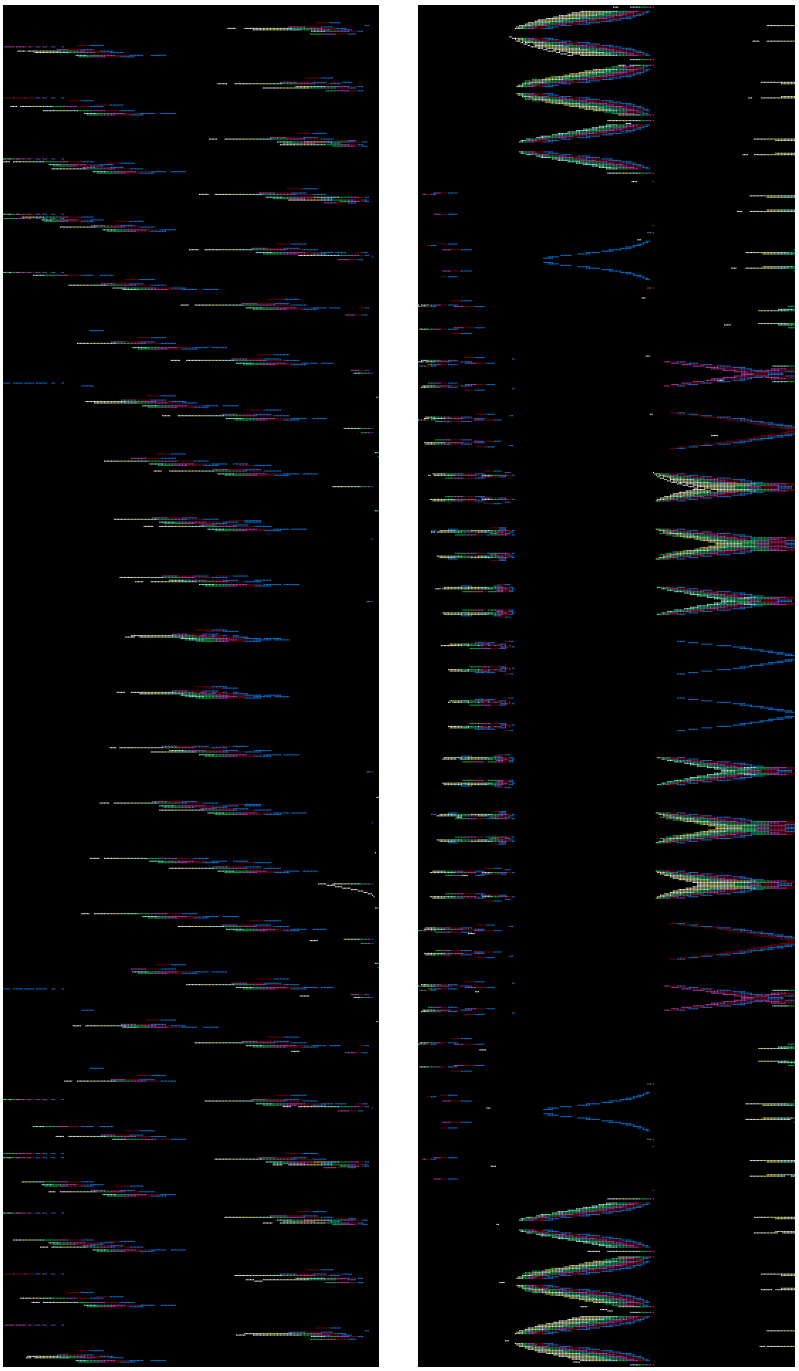
La Llamita : Y-Axis Symmetry on left, X-Y axis Symmetry on Right



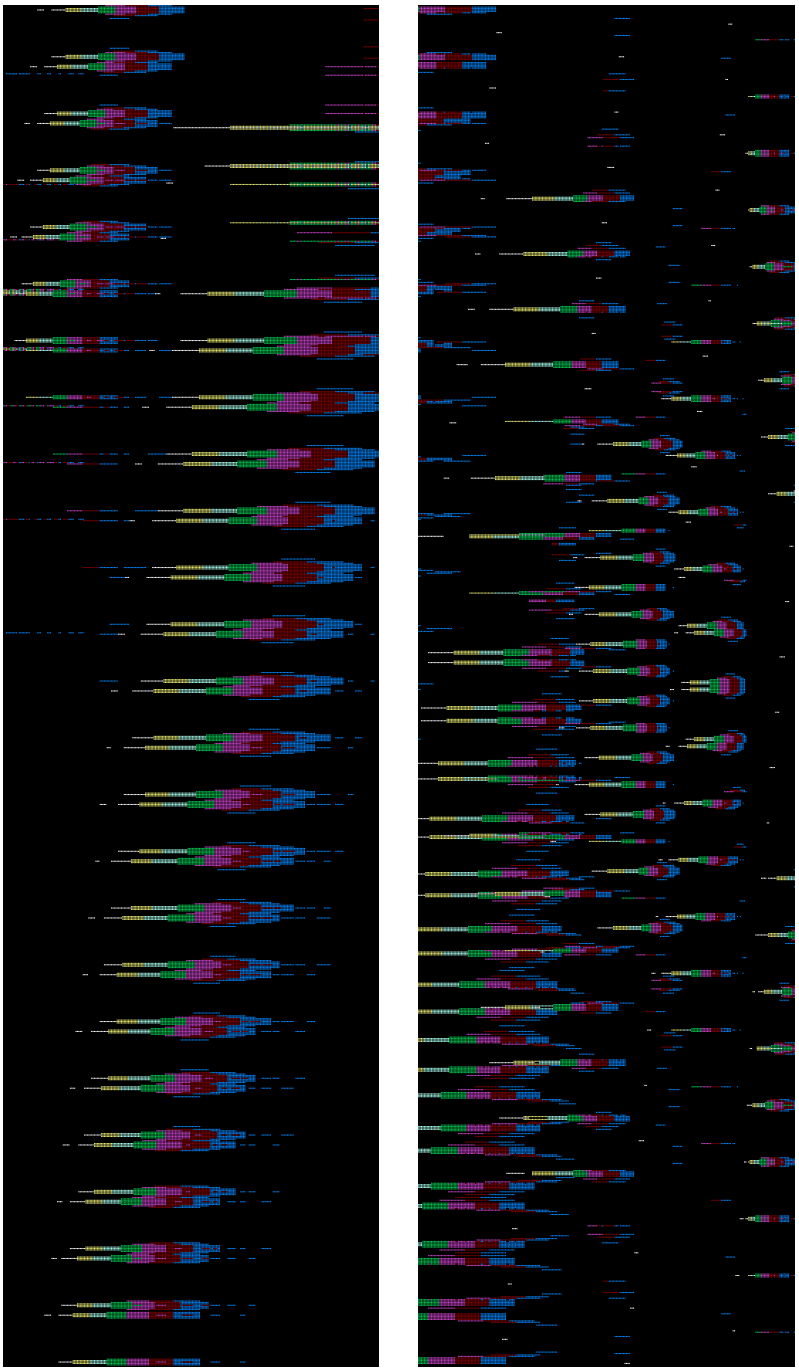
La Llamita : X-Axis Symmetry on left, Quad Symmetry on Right



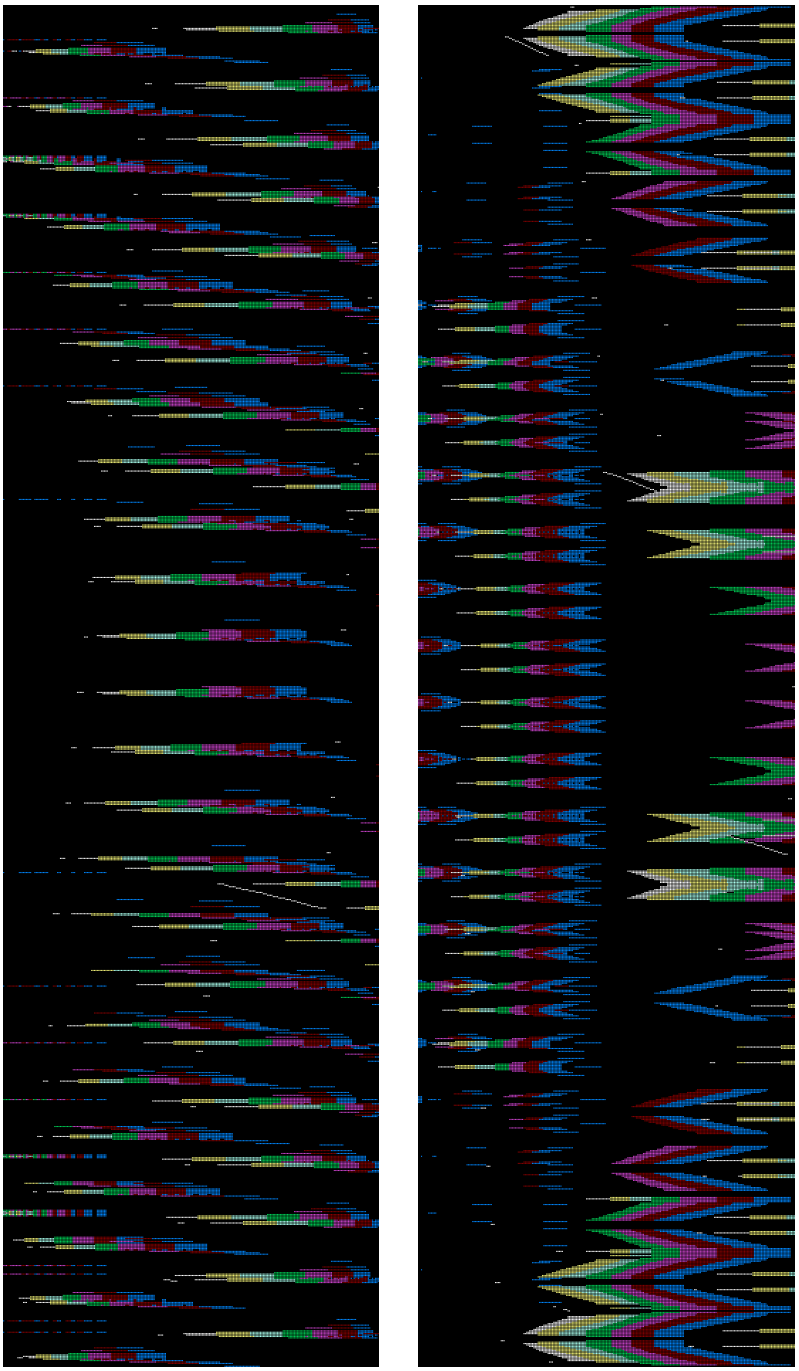
Star Two : Y-Axis Symmetry on left, X-Y axis Symmetry on Right



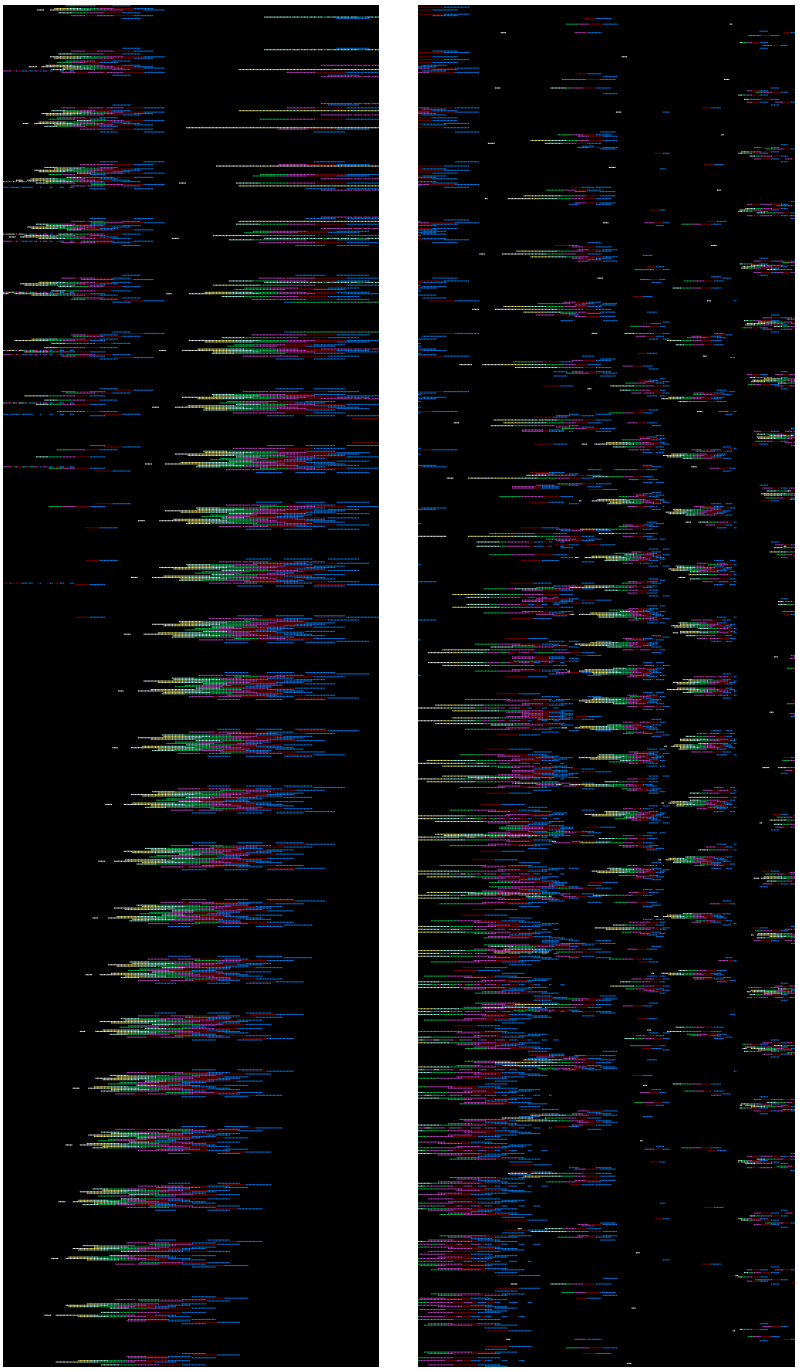
Star Two : X-Axis Symmetry on left, Quad Symmetry on Right



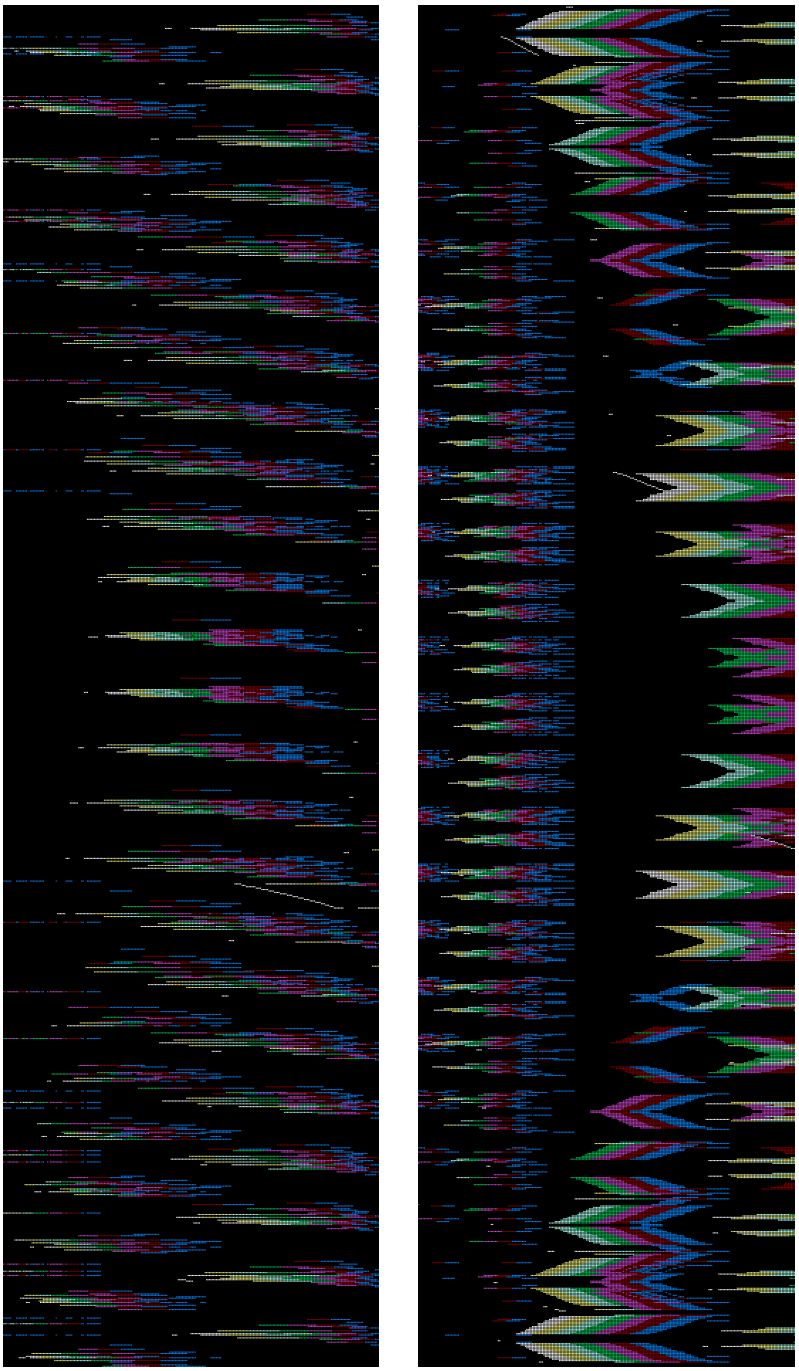
Deltoid : Y-Axis Symmetry on left, X-Y axis Symmetry on Right



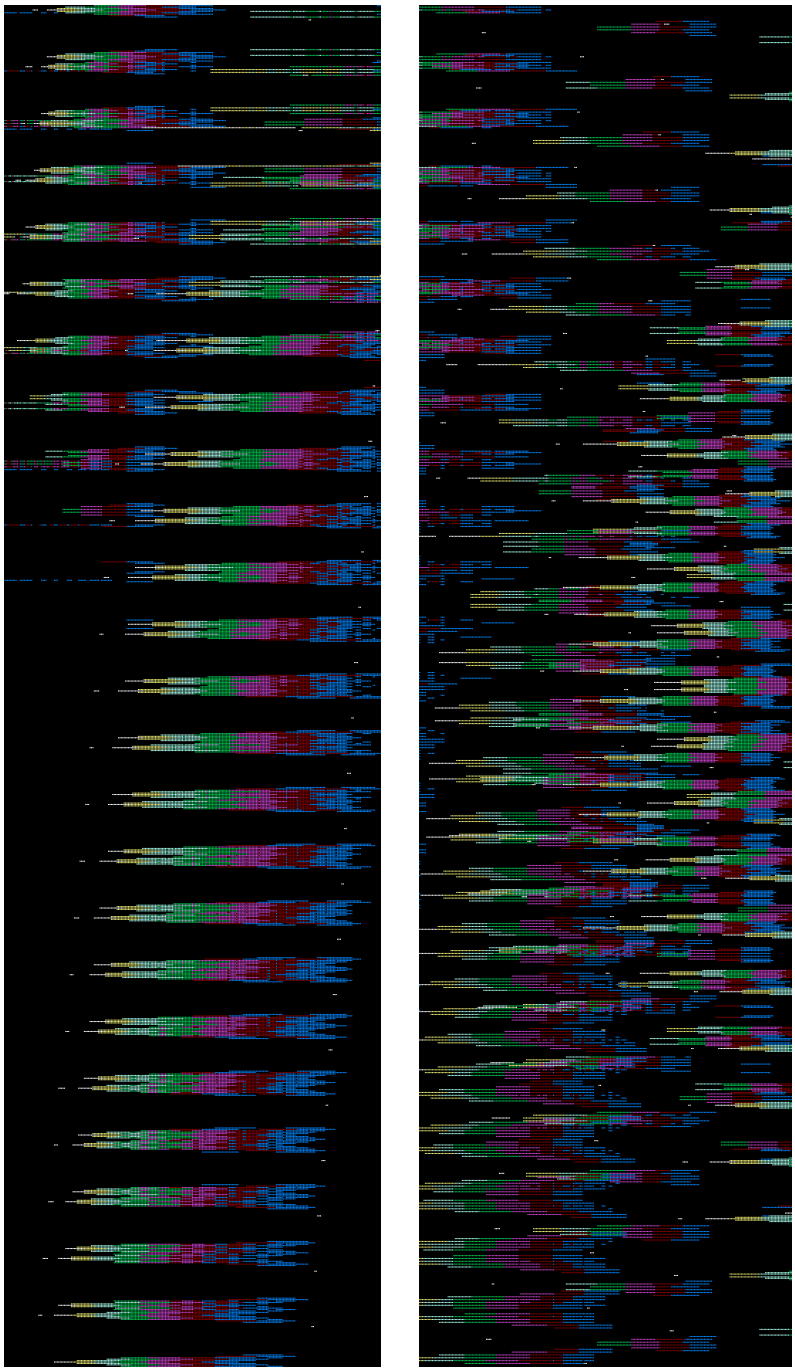
Deltoid : X-Axis Symmetry on left, Quad Symmetry on Right



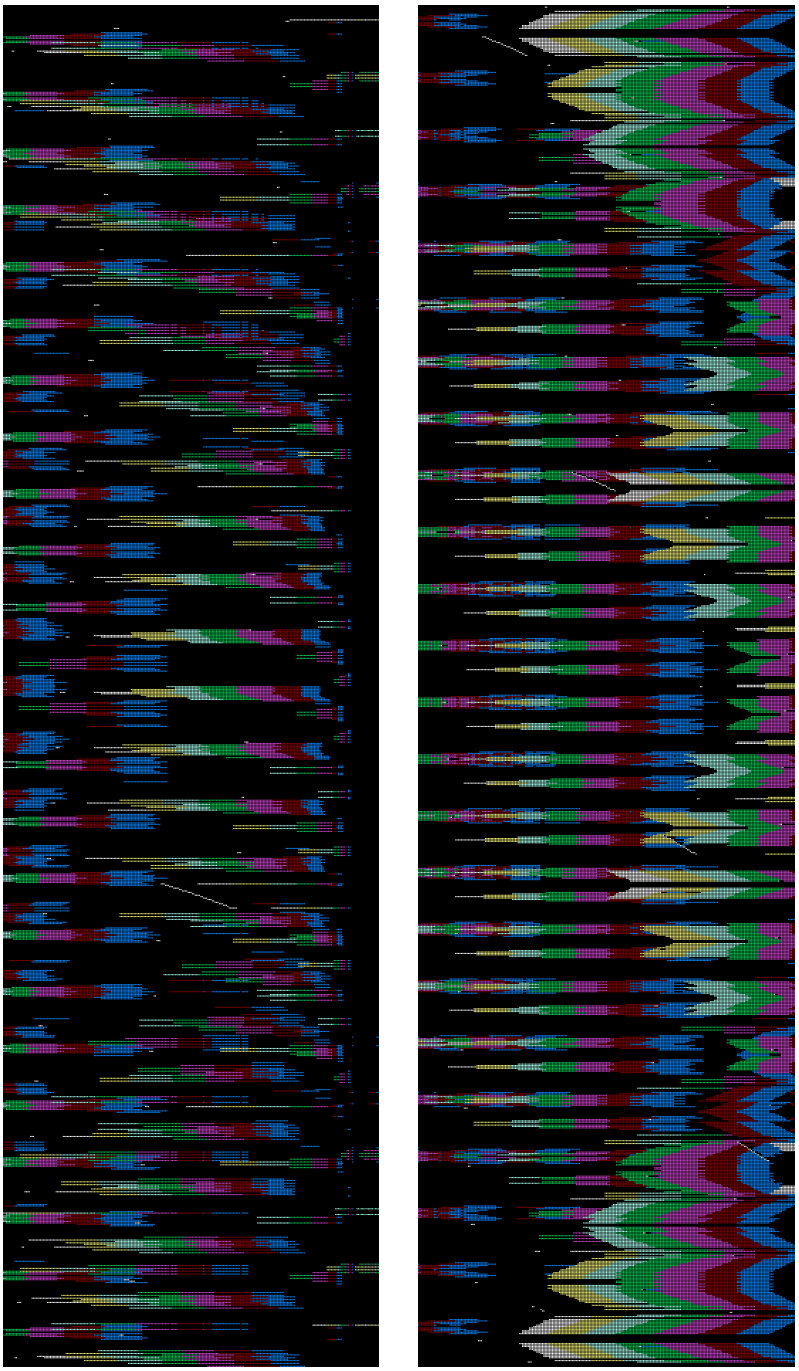
Diffused : Y-Axis Symmetry on left, X-Y axis Symmetry on Right



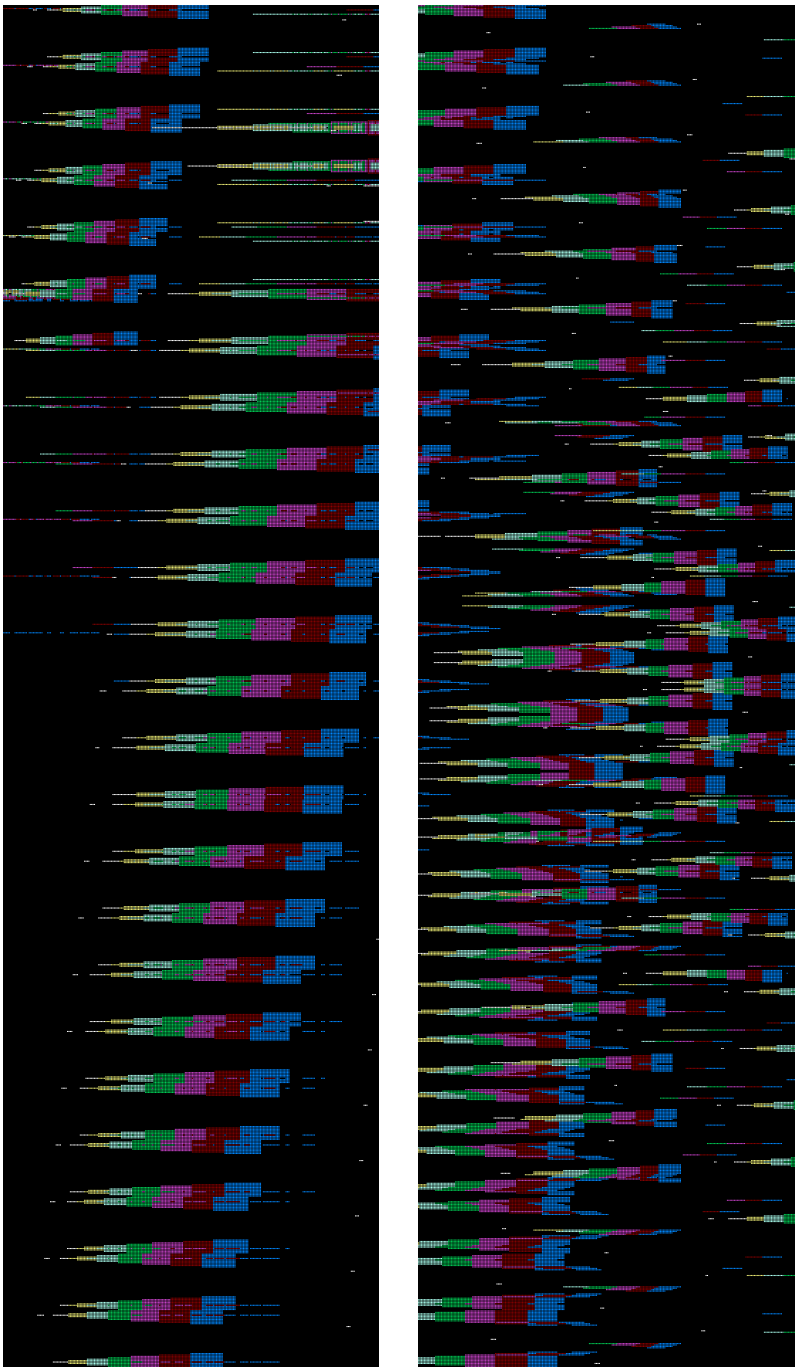
Diffused : X-Axis Symmetry on left, Quad Symmetry on Right



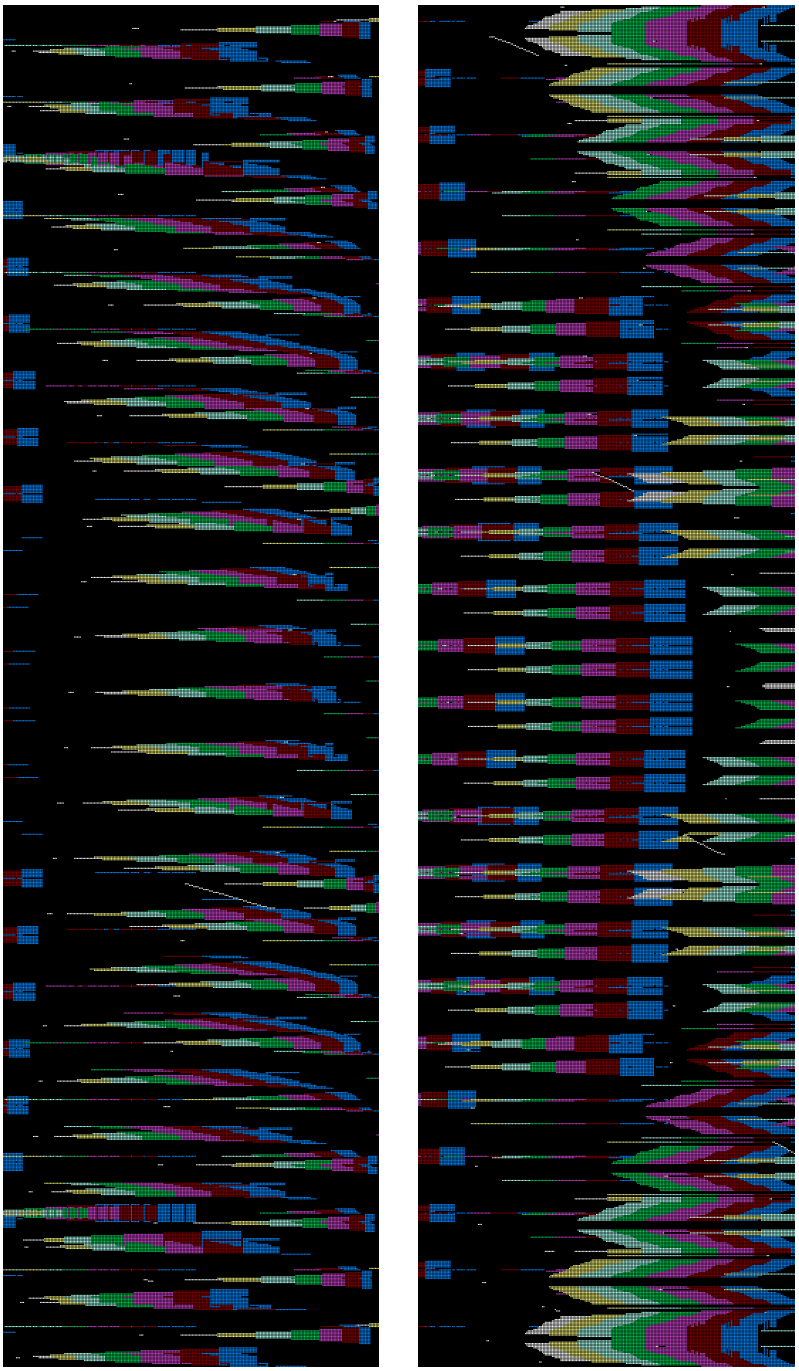
Multi-Cross : Y-Axis Symmetry on left, X-Y axis Symmetry on Right



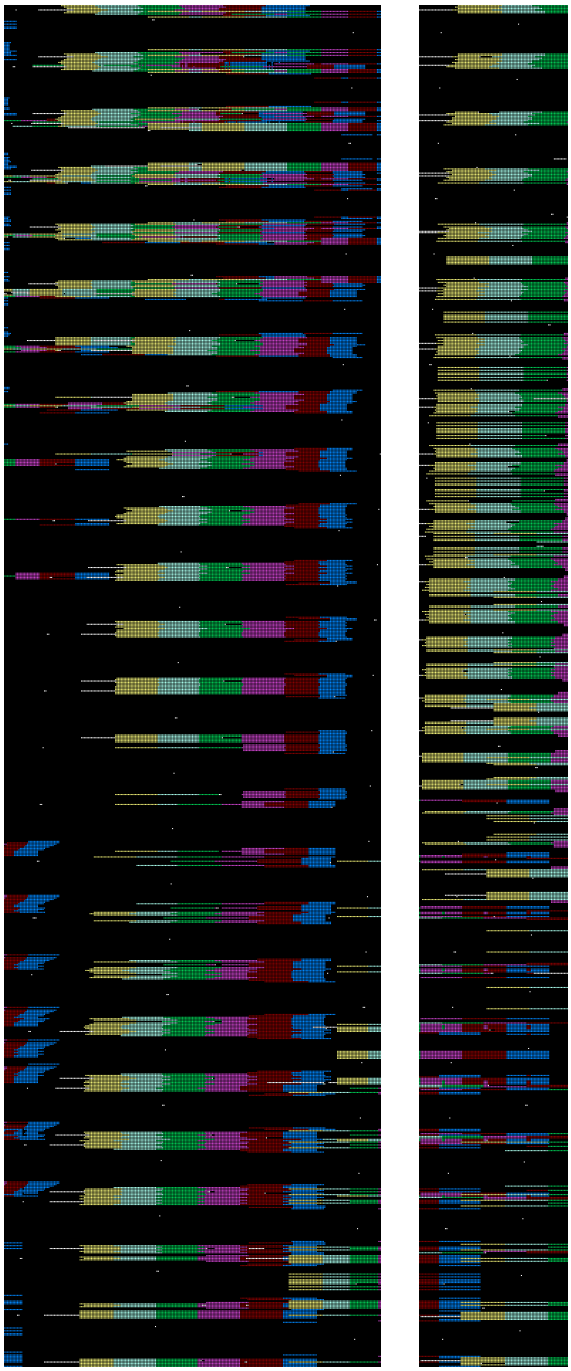
Multi-Cross : X-Axis Symmetry on left, Quad Symmetry on Right



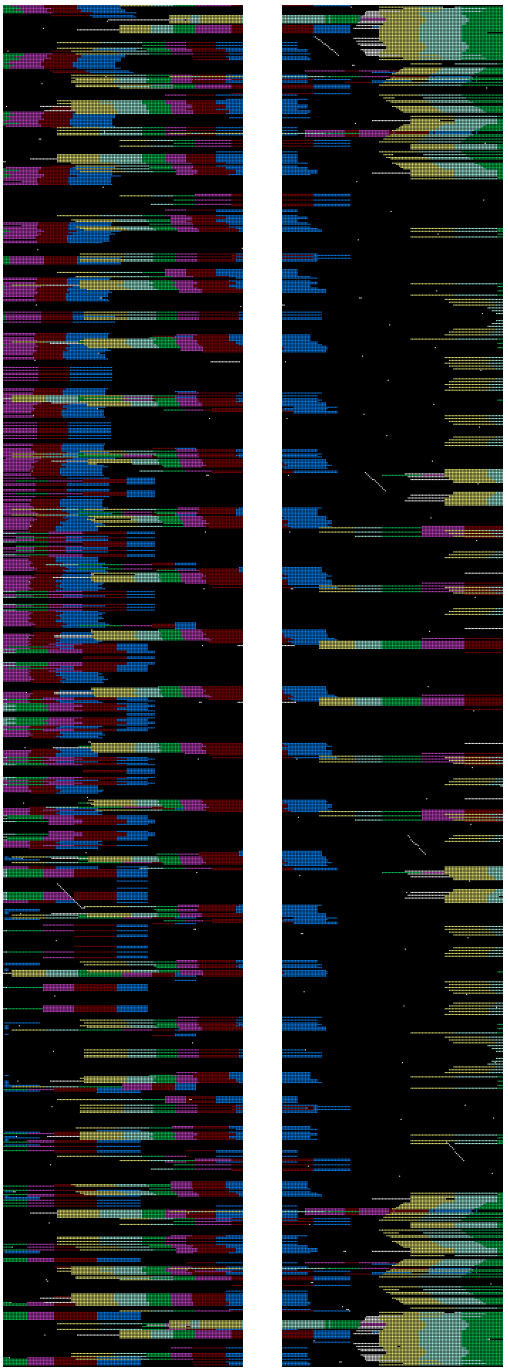
Pulsar : Y-Axis Symmetry on left, X-Y axis Symmetry on Right



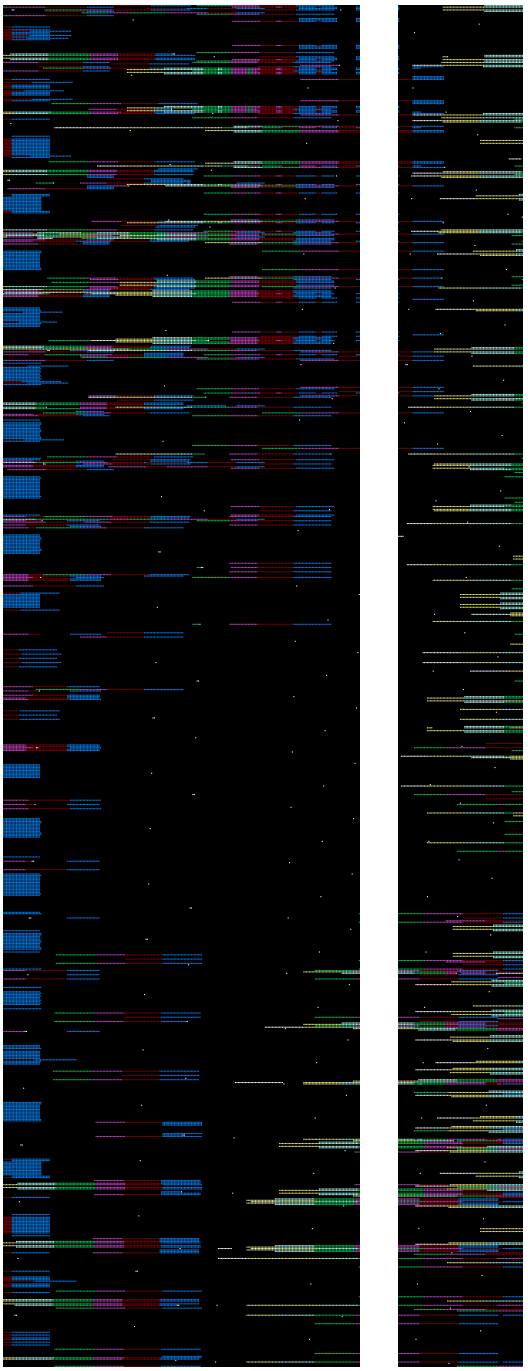
Pulsar : X-Axis Symmetry on left, Quad Symmetry on Right



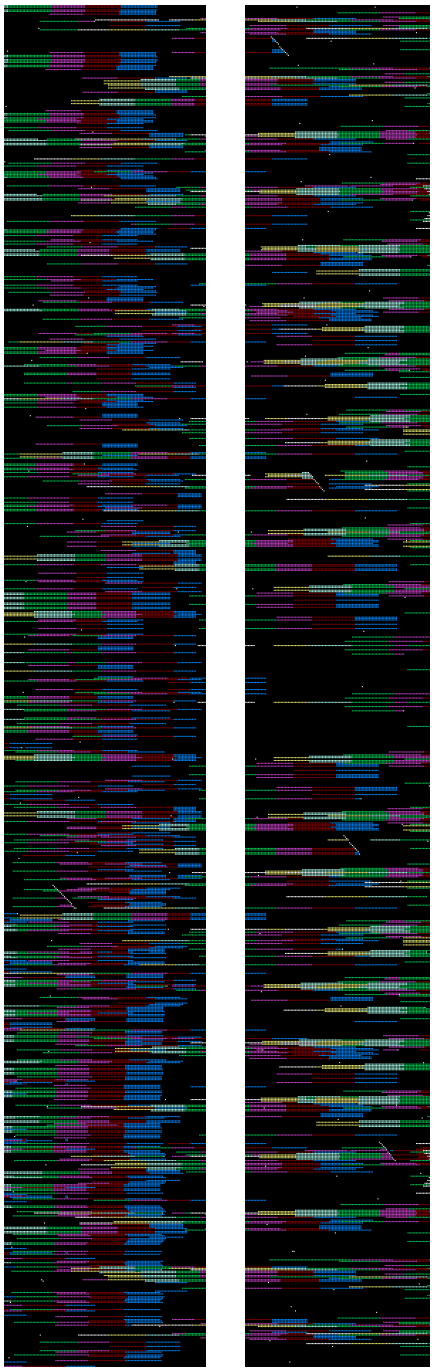
Custom Pattern 1 : Y-Axis Symmetry on left, X-Y axis Symmetry on Right



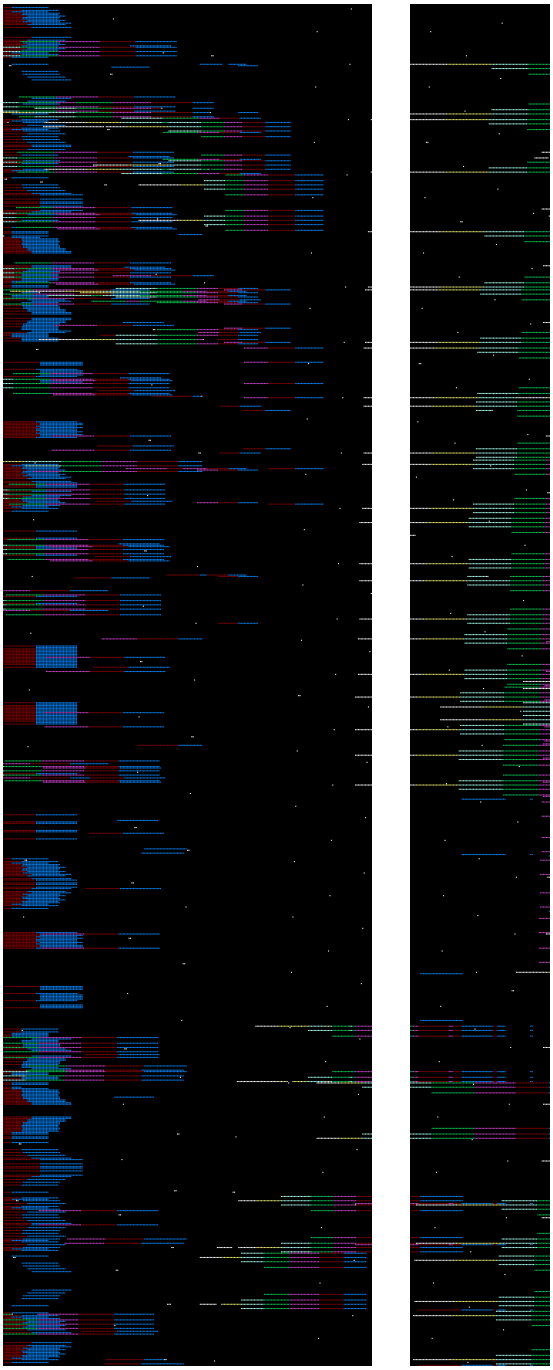
Custom Pattern 1 : X-Axis Symmetry on left, Quad Symmetry on Right



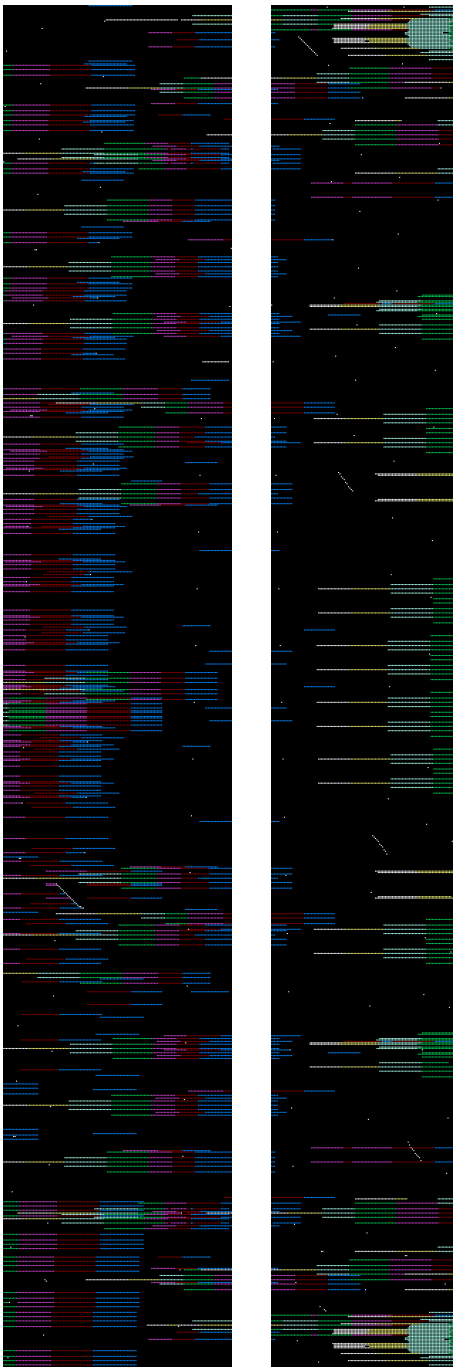
Custom Pattern 2 : Y-Axis Symmetry on left, X-Y axis Symmetry on Right



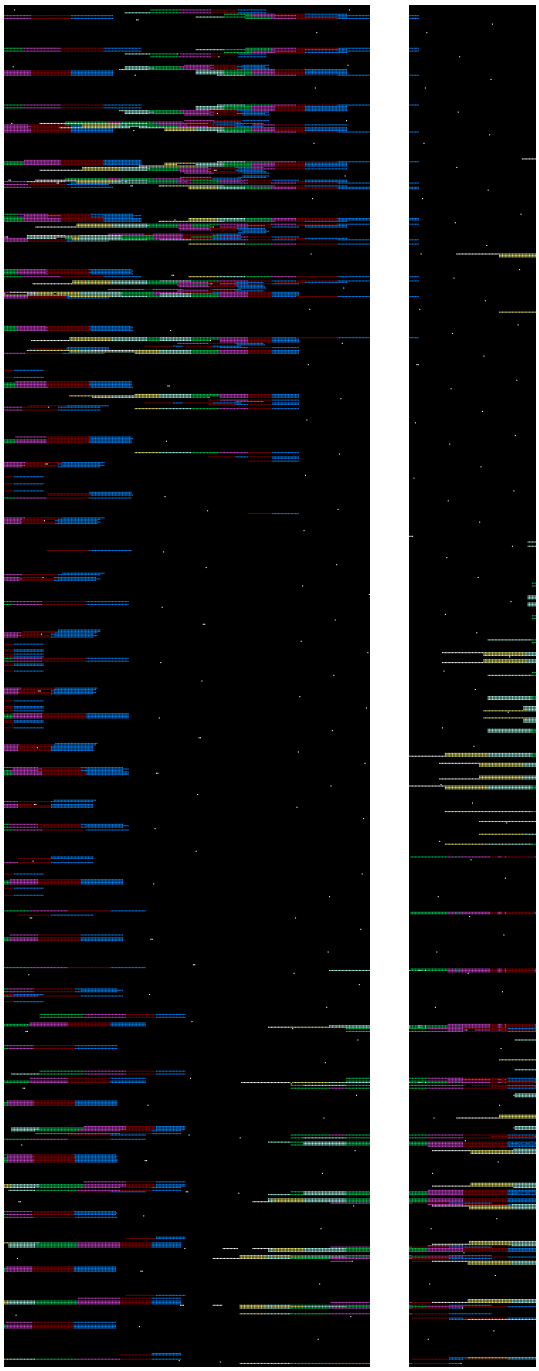
Custom Pattern 2 : X-Axis Symmetry on left, Quad Symmetry on Right



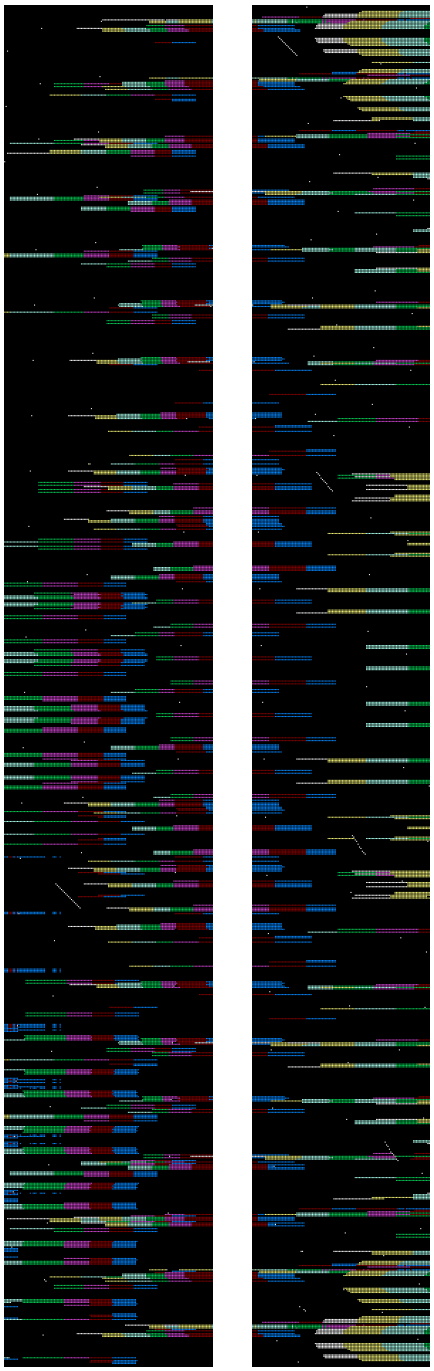
Custom Pattern 3 : Y-Axis Symmetry on left, X-Y axis Symmetry on Right



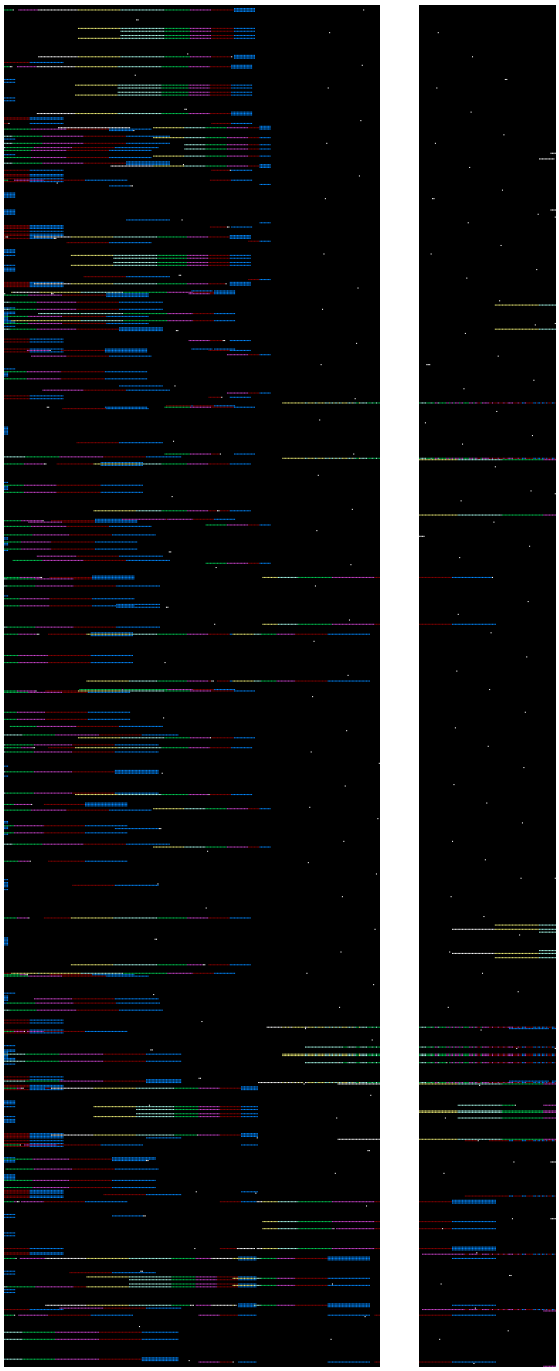
Custom Pattern 3 : X-Axis Symmetry on left, Quad Symmetry on Right



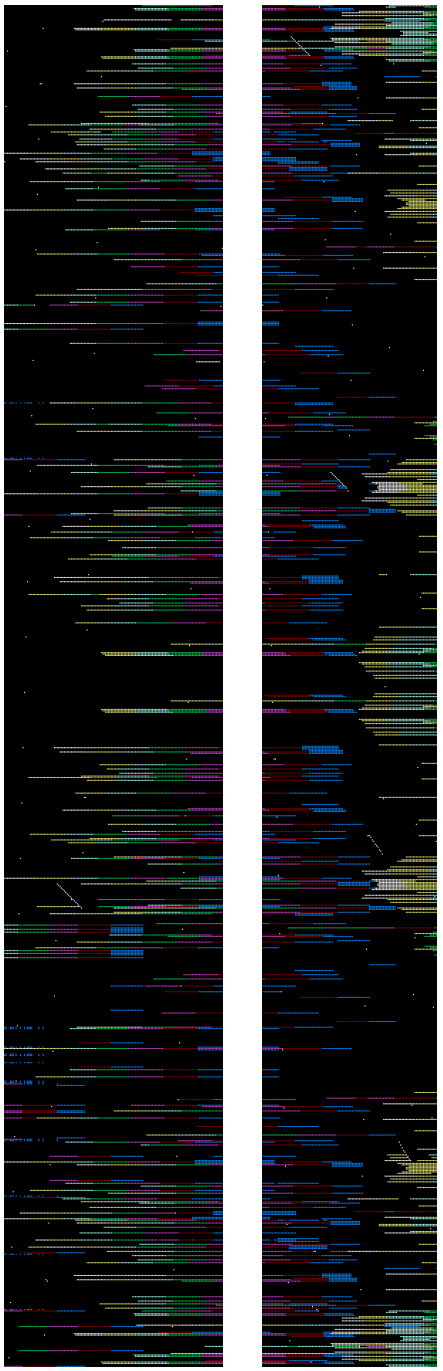
Custom Pattern 4 : Y-Axis Symmetry on left, X-Y axis Symmetry on Right



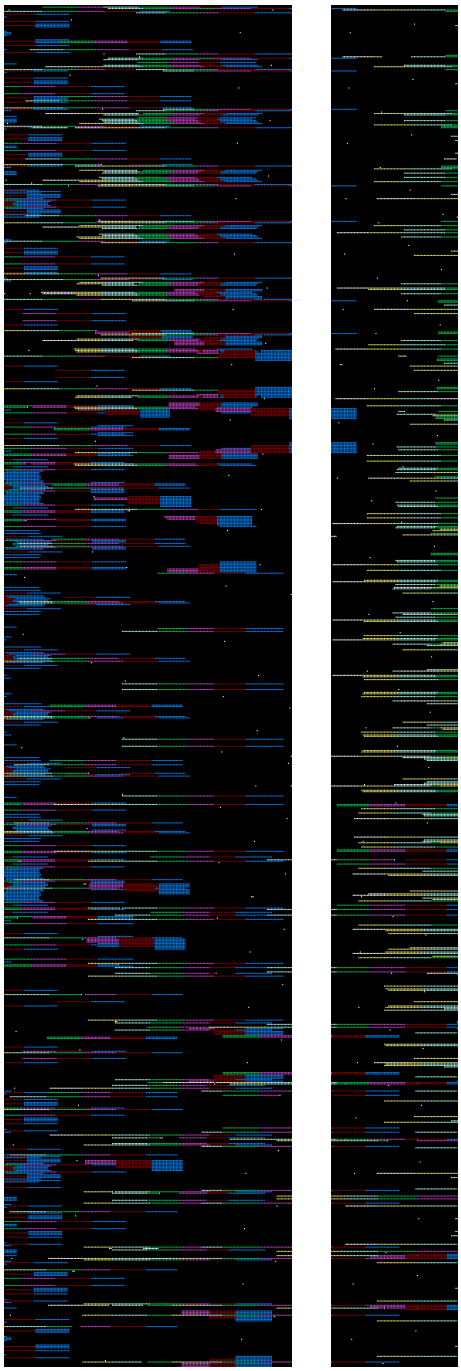
Custom Pattern 4 : X-Axis Symmetry on left, Quad Symmetry on Right



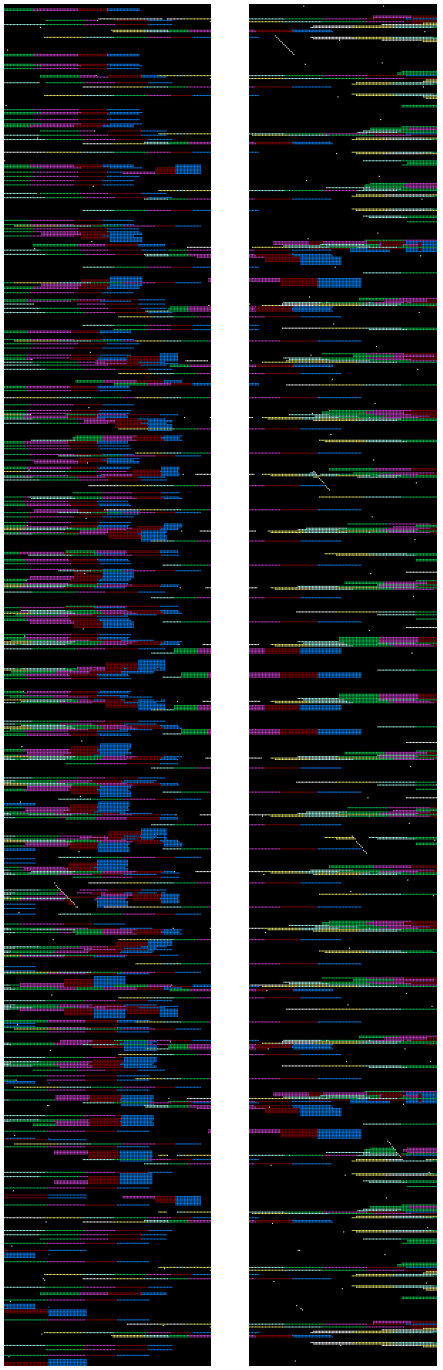
Custom Pattern 5 : Y-Axis Symmetry on left, X-Y axis Symmetry on Right



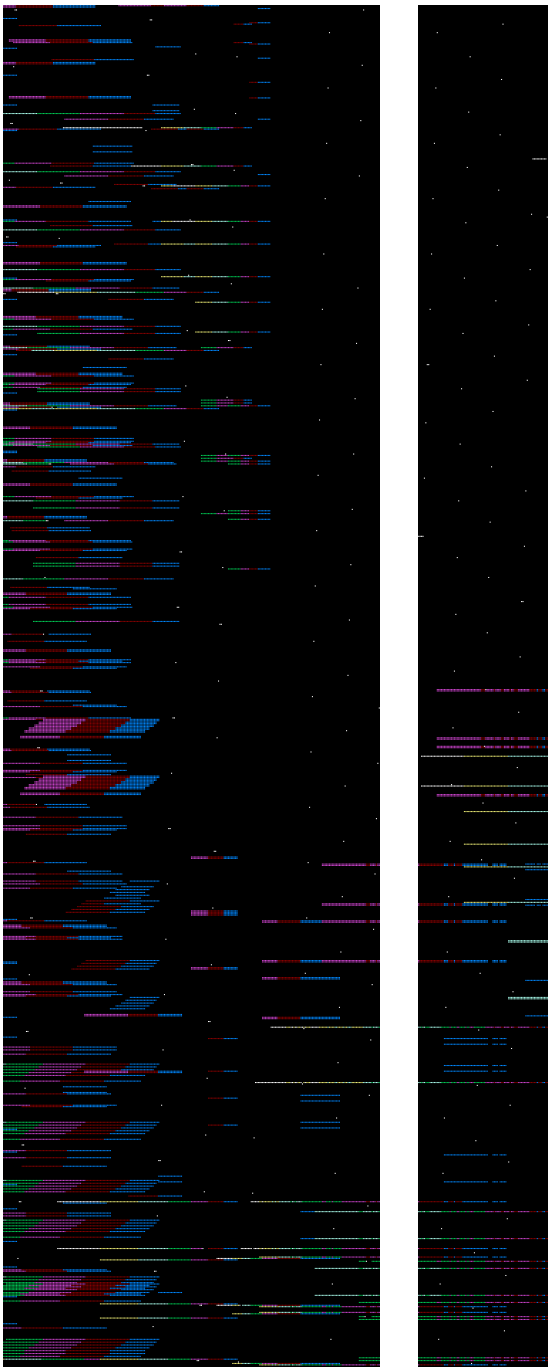
Custom Pattern 5 : X-Axis Symmetry on left, Quad Symmetry on Right



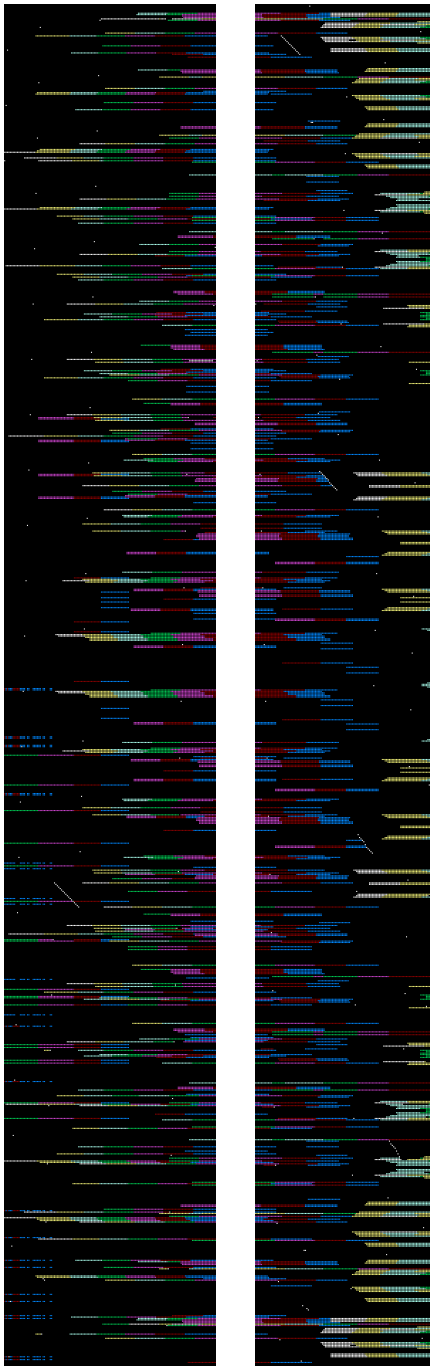
Custom Pattern 6 : Y-Axis Symmetry on left, X-Y axis Symmetry on Right



Custom Pattern 6 : X-Axis Symmetry on left, Quad Symmetry on Right



Custom Pattern 7 : Y-Axis Symmetry on left, X-Y axis Symmetry on Right



Custom Pattern 7 : X-Axis Symmetry on left, Quad Symmetry on Right

Psychedelia Syndrome

Findings and Recommendations

Lamasoft Corporation