

Collective Communication: Theory and Practice

Robert van de Geijn

Acknowledgements

This presentation is based on work in the mid-1990s that was sponsored in part by the Intel Research Council and Intel Scalable Systems Division. At that time, David Payne, Lance Shuler, and Jerrell Watts contributed to the research

Outline

Part I: Theory

- Model of parallel computation
- Collective communications
- A building block approach to library implementation

Part II: Practice

- Implementation on the Paragon
- Performance results

Outline

Part I: Theory

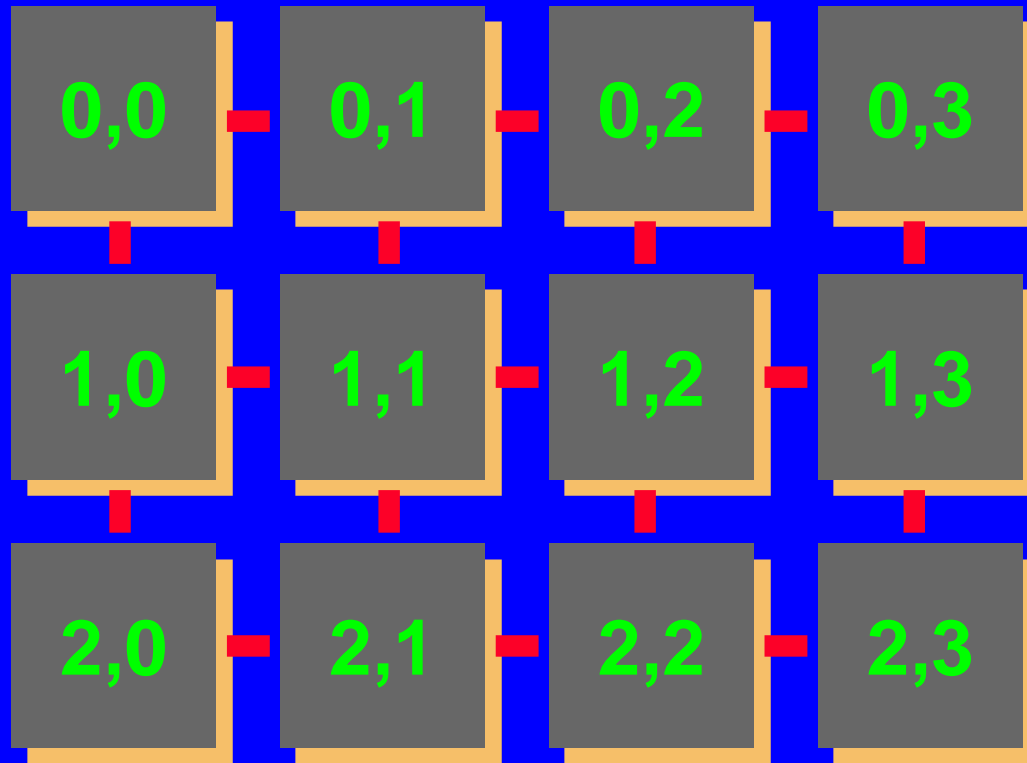
- **Model of parallel computation**
- Collective communications
- A building block approach to library implementation

Part II: Practice

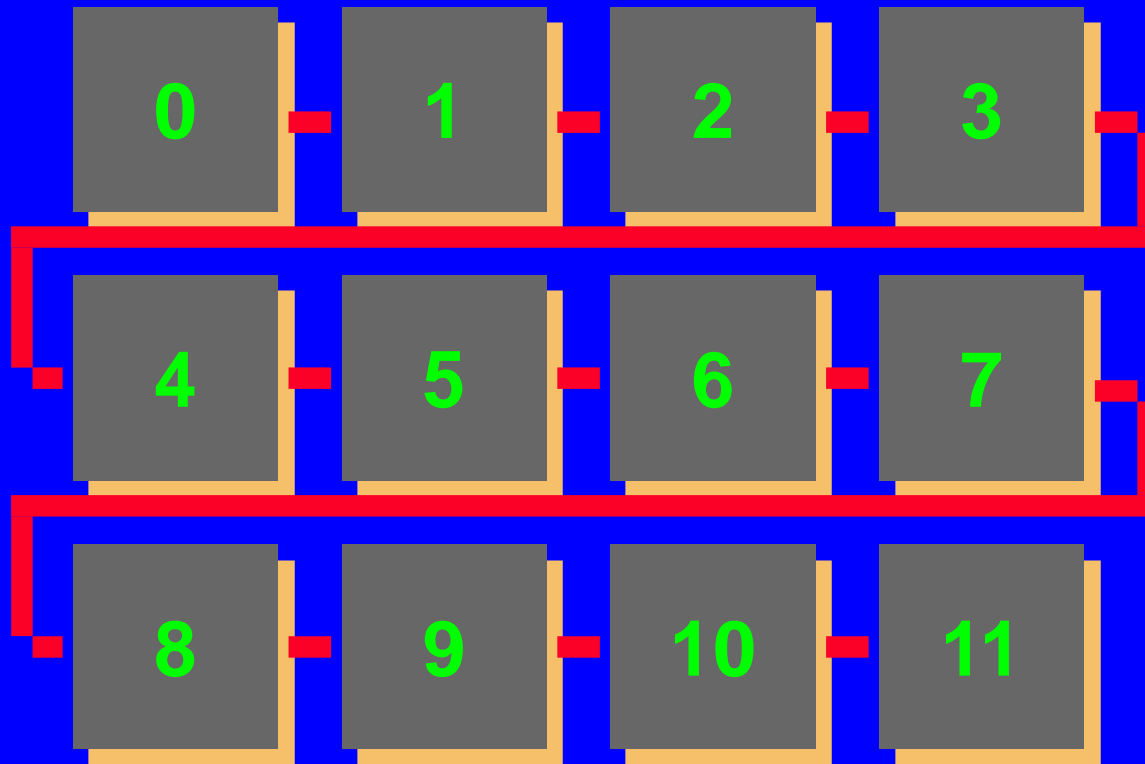
- Implementation on the Paragon
- Performance results
- Applications

Model of Parallel Computation

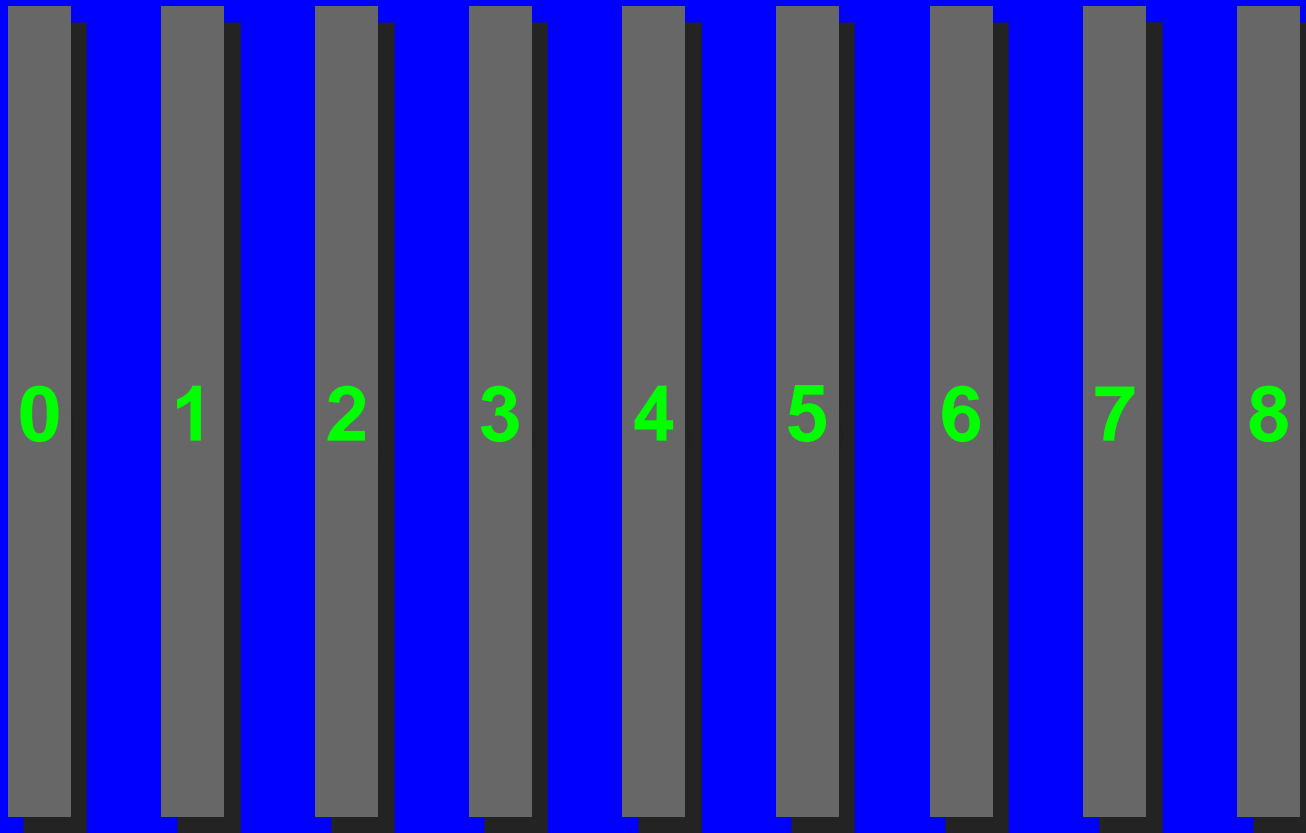
- p nodes
- physical two dimensional mesh
 - r rows, c columns
 - nodes have physical indices (i,j)
- often logically viewed as a linear array
 - indexed $0, \dots, p-1$
 - nodes are numbered in row-major order



- physical two dimensional mesh
 - r rows, c columns
 - nodes have physical indices (i,j)



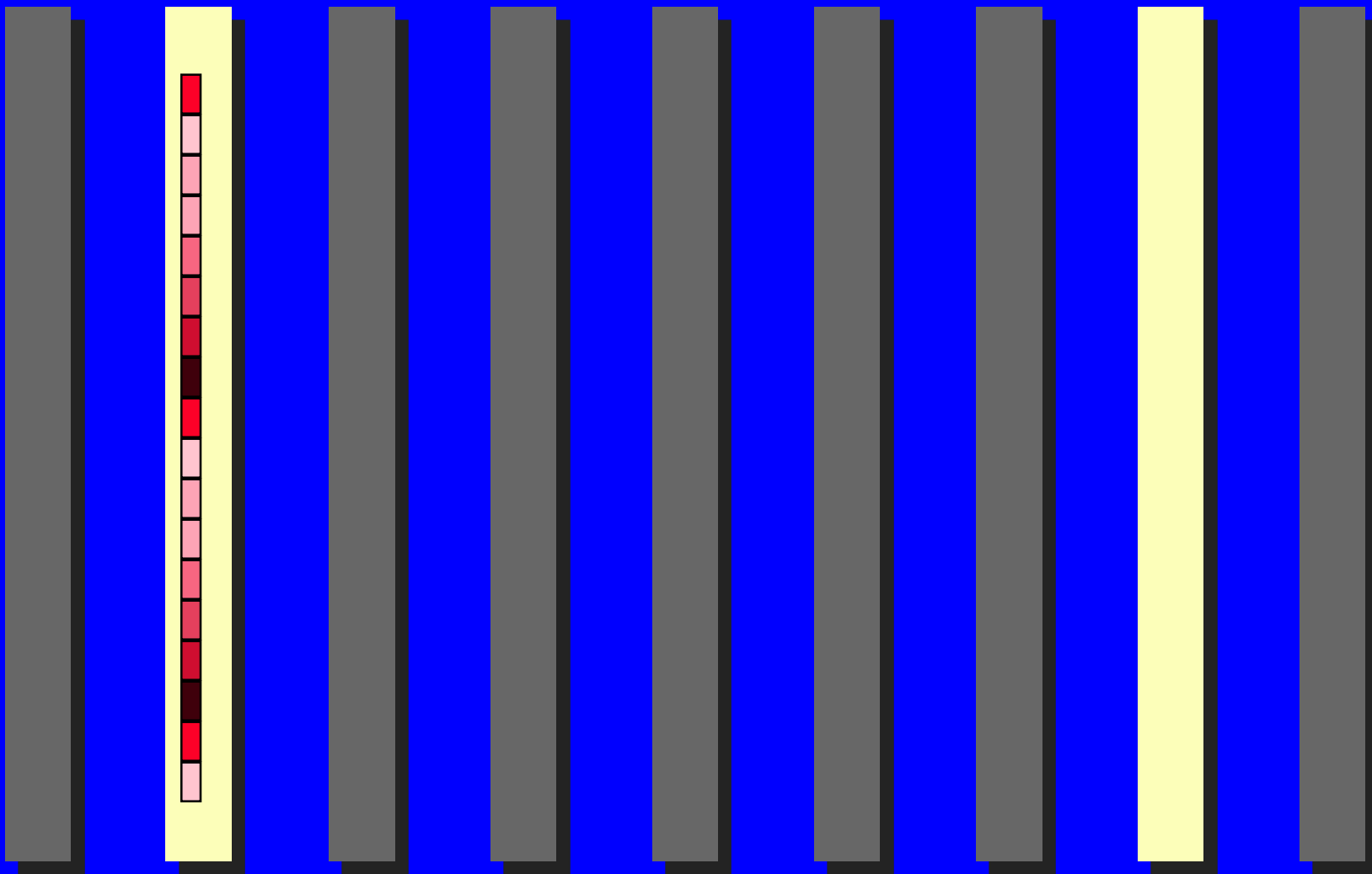
- often logically viewed as a linear array
 - indexed $0, \dots, p-1$
 - nodes are numbered in row-major order

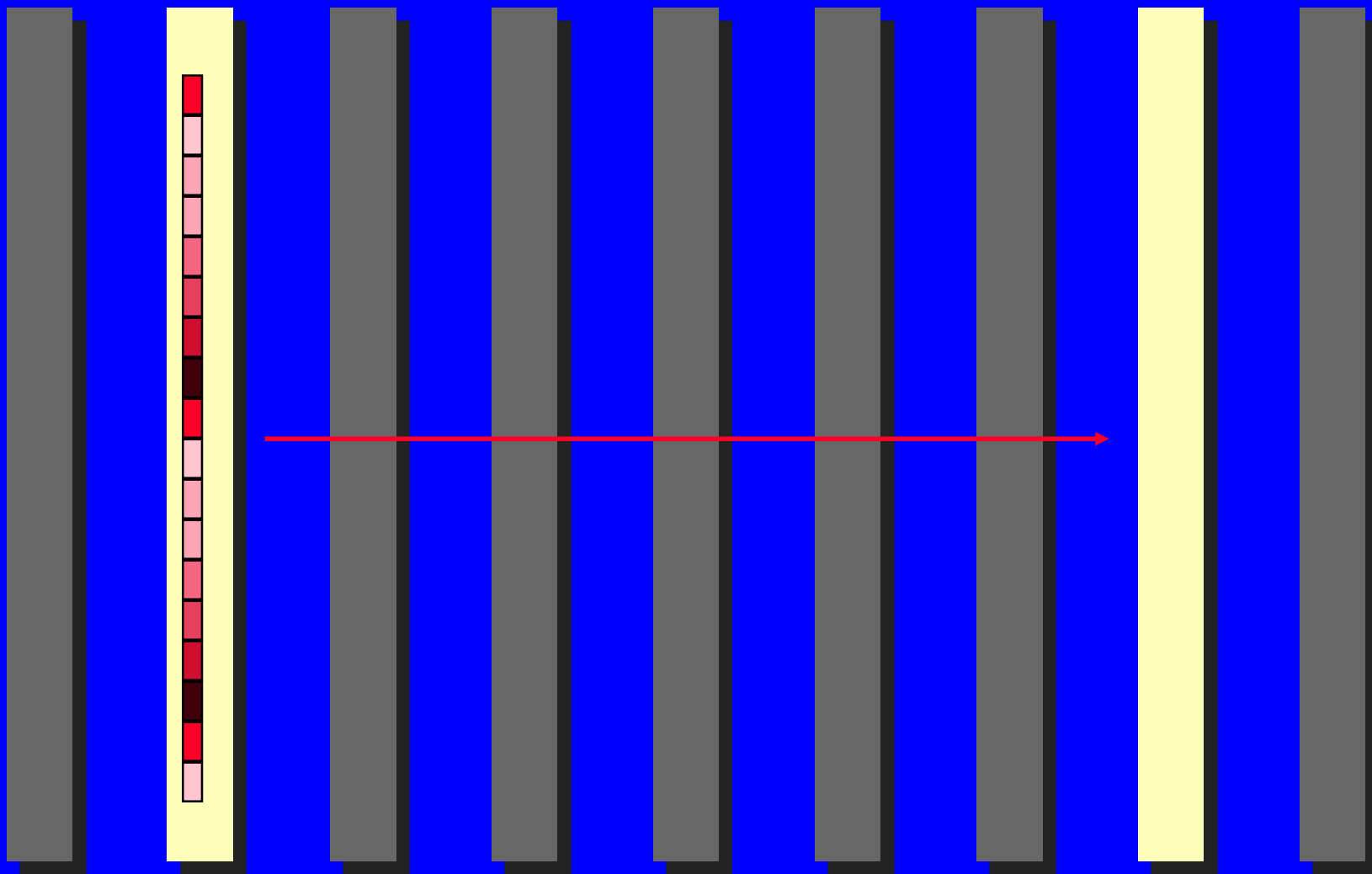


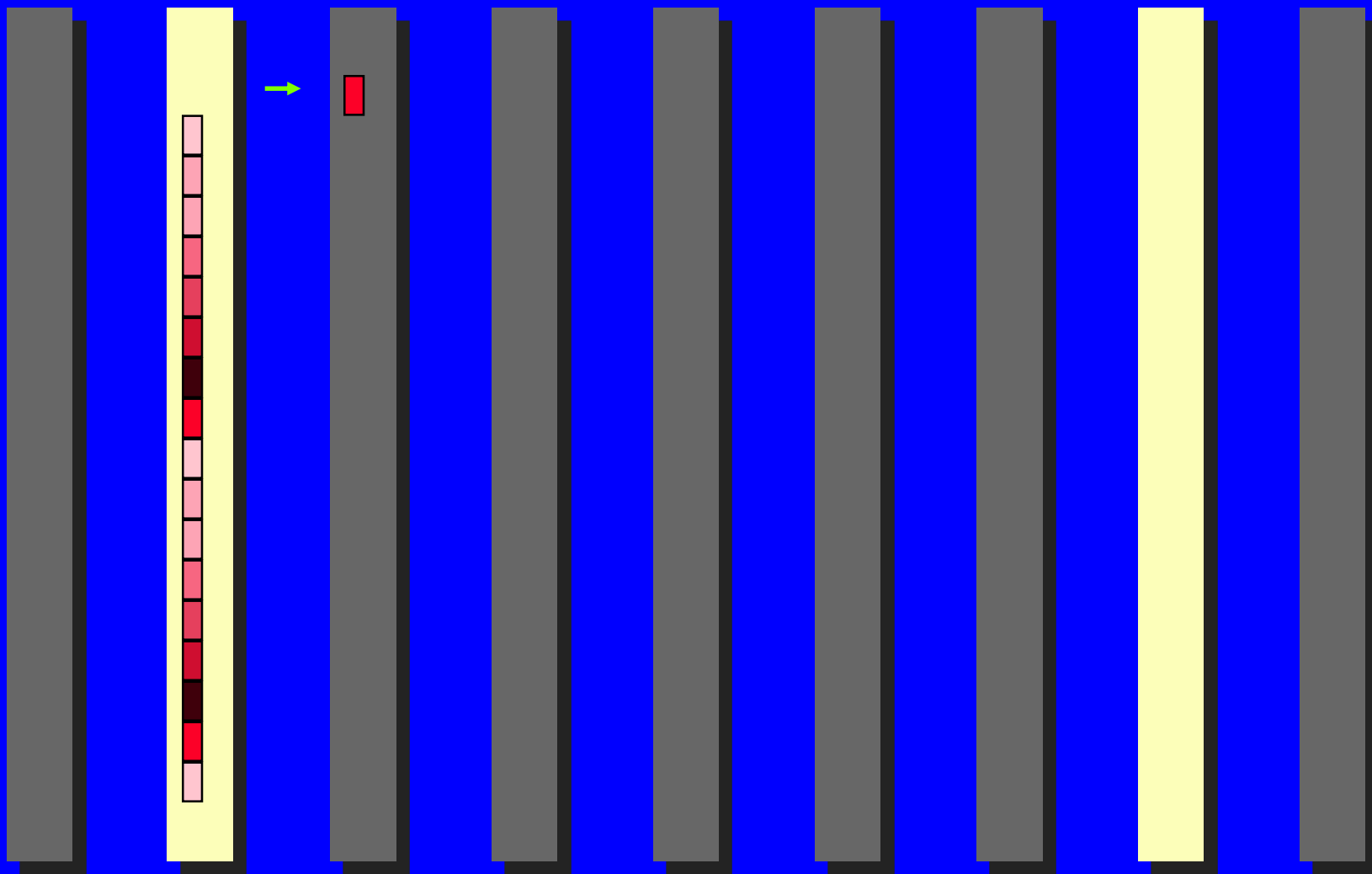
- often logically viewed as a linear array
 - indexed $0, \dots, p-1$

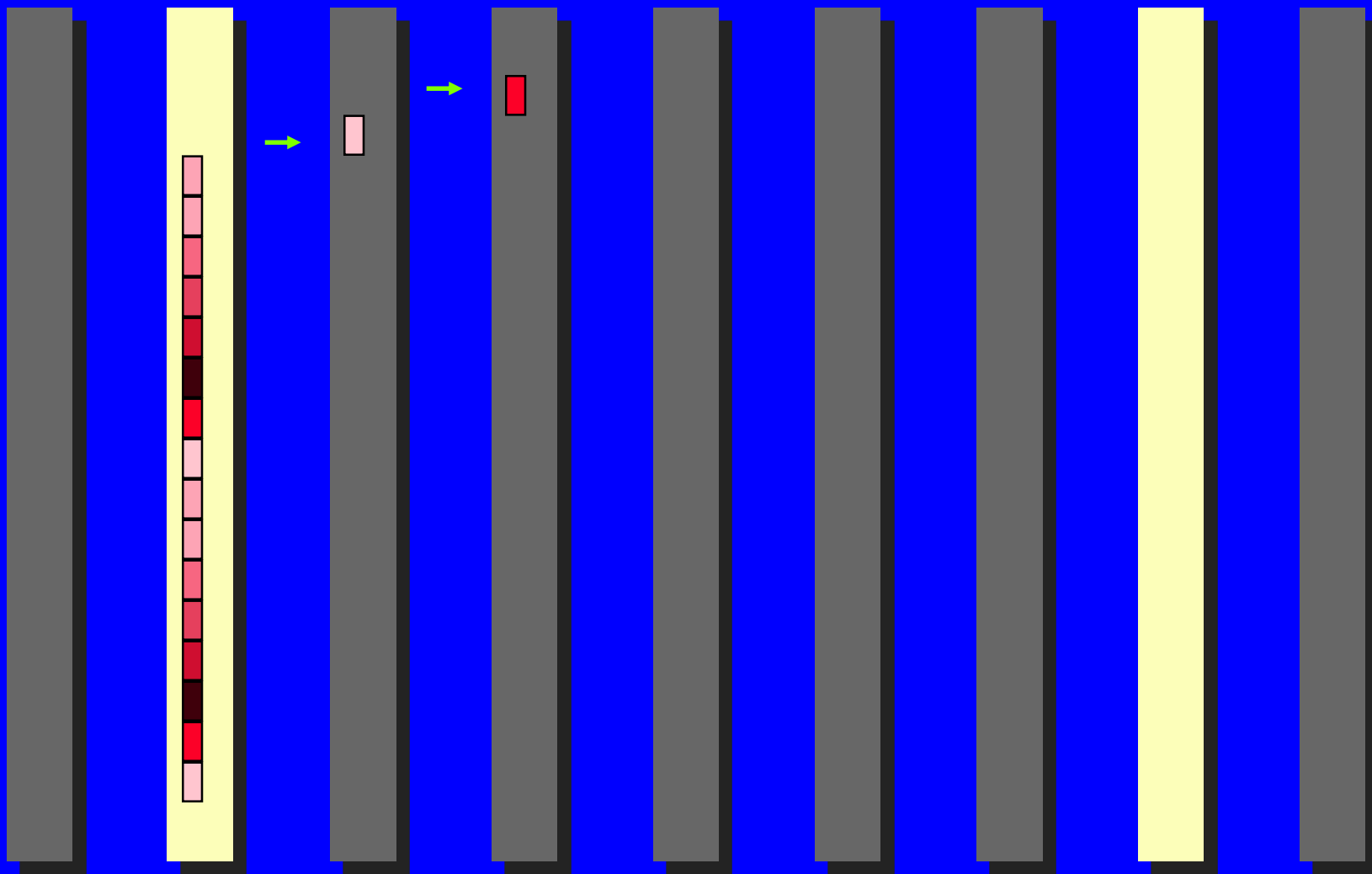
The Cost of Communication

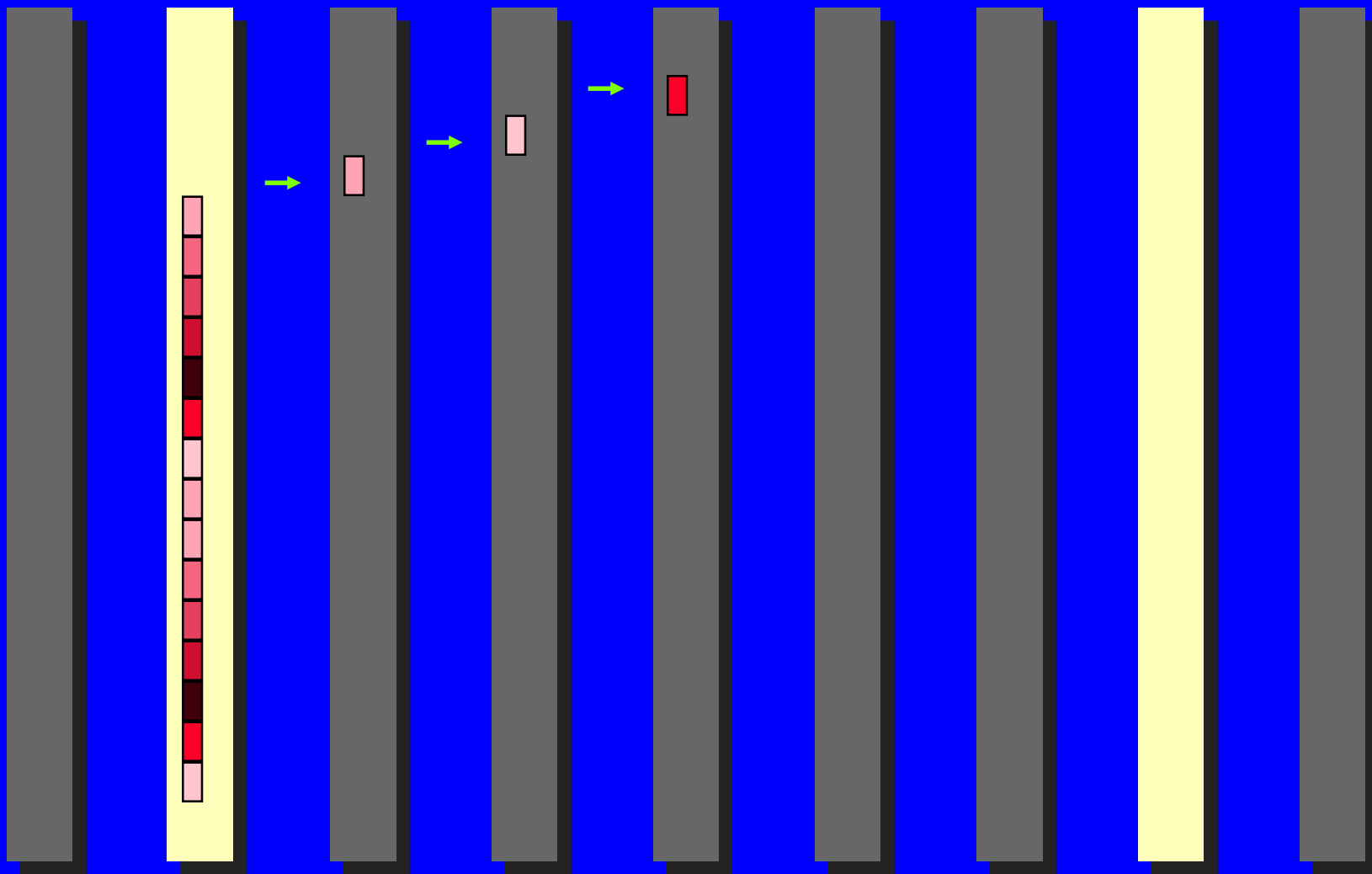
- send a message of length n over d links
- packetize the message
- Example: $d=6$

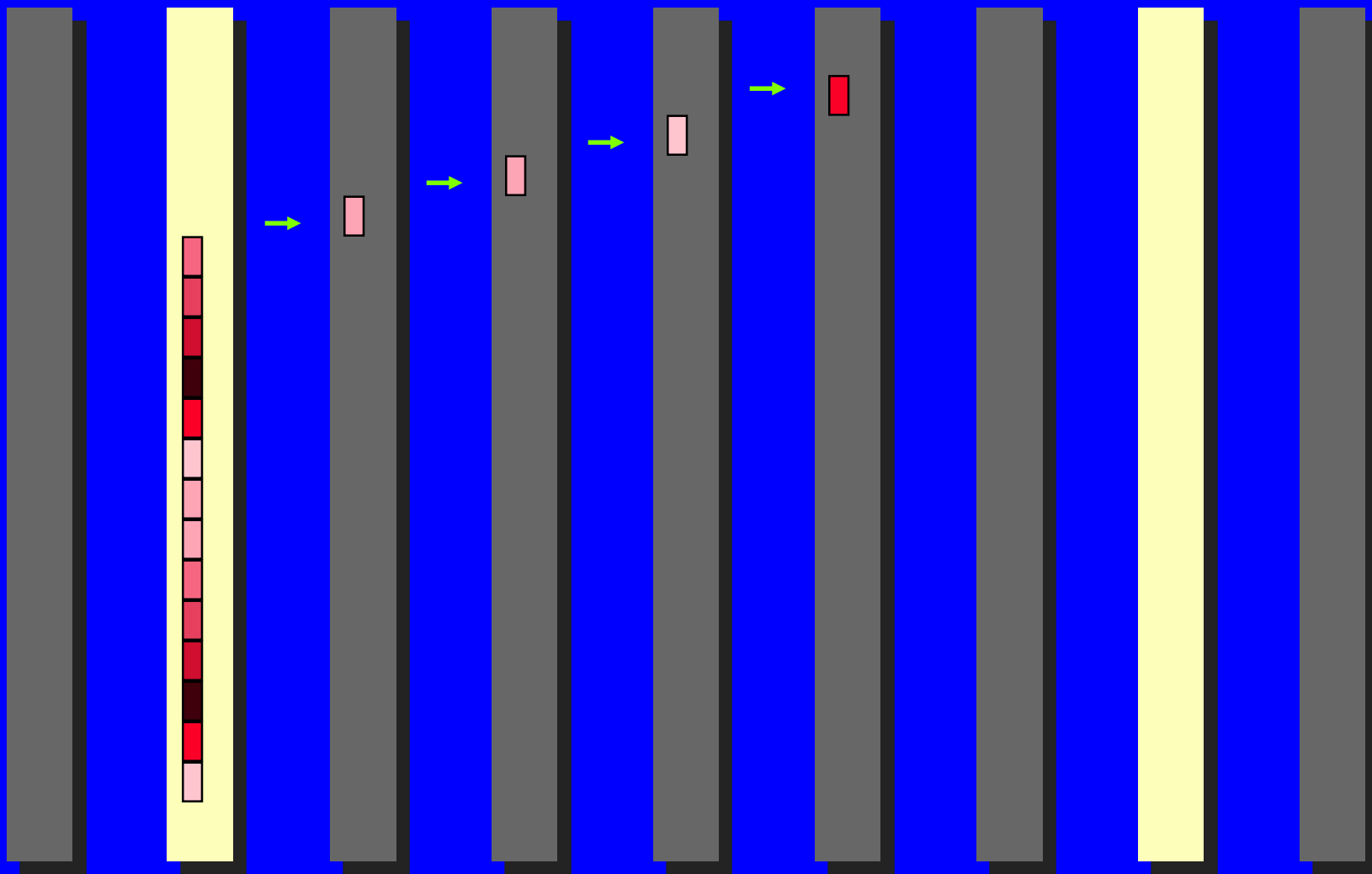


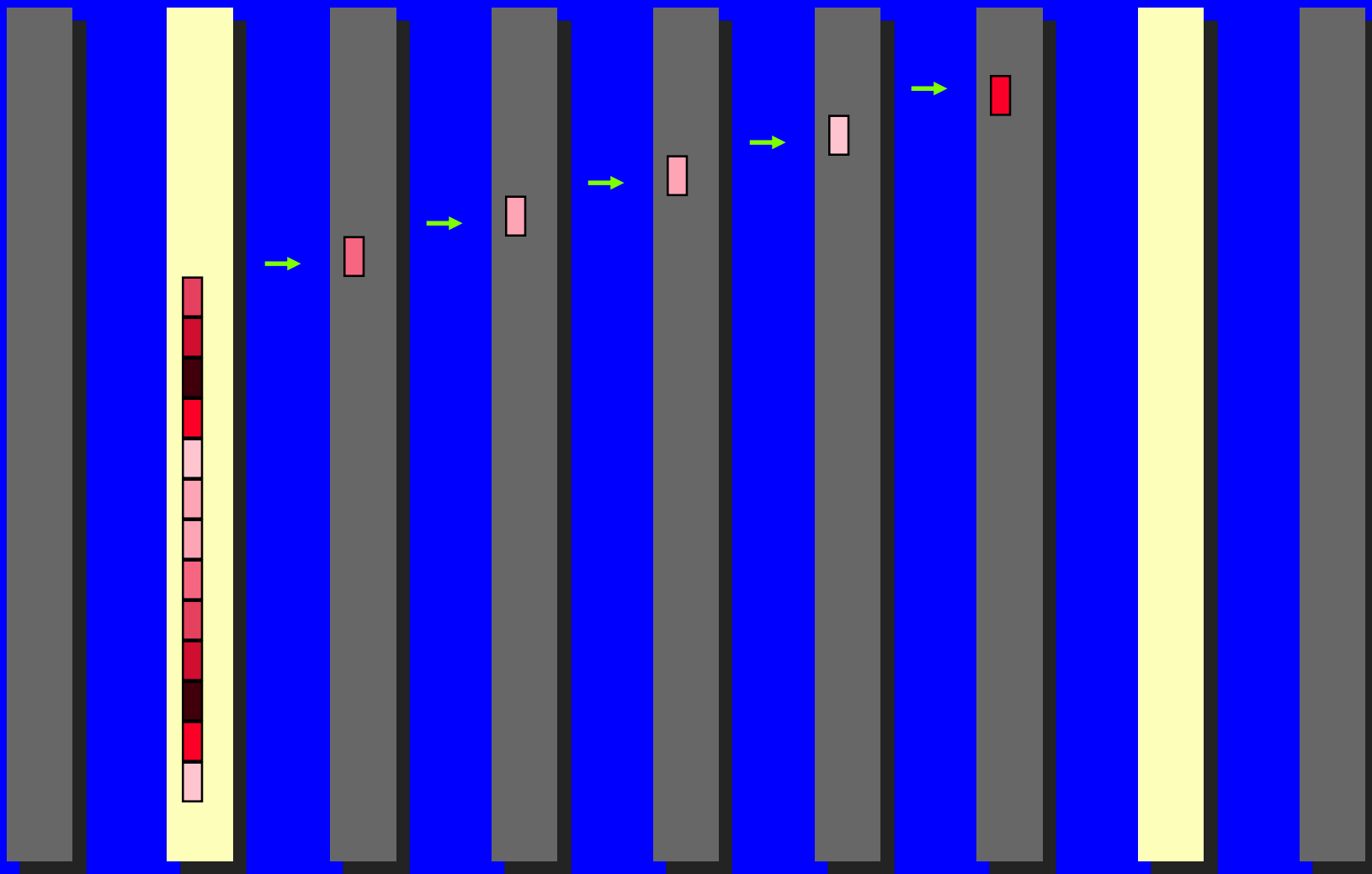


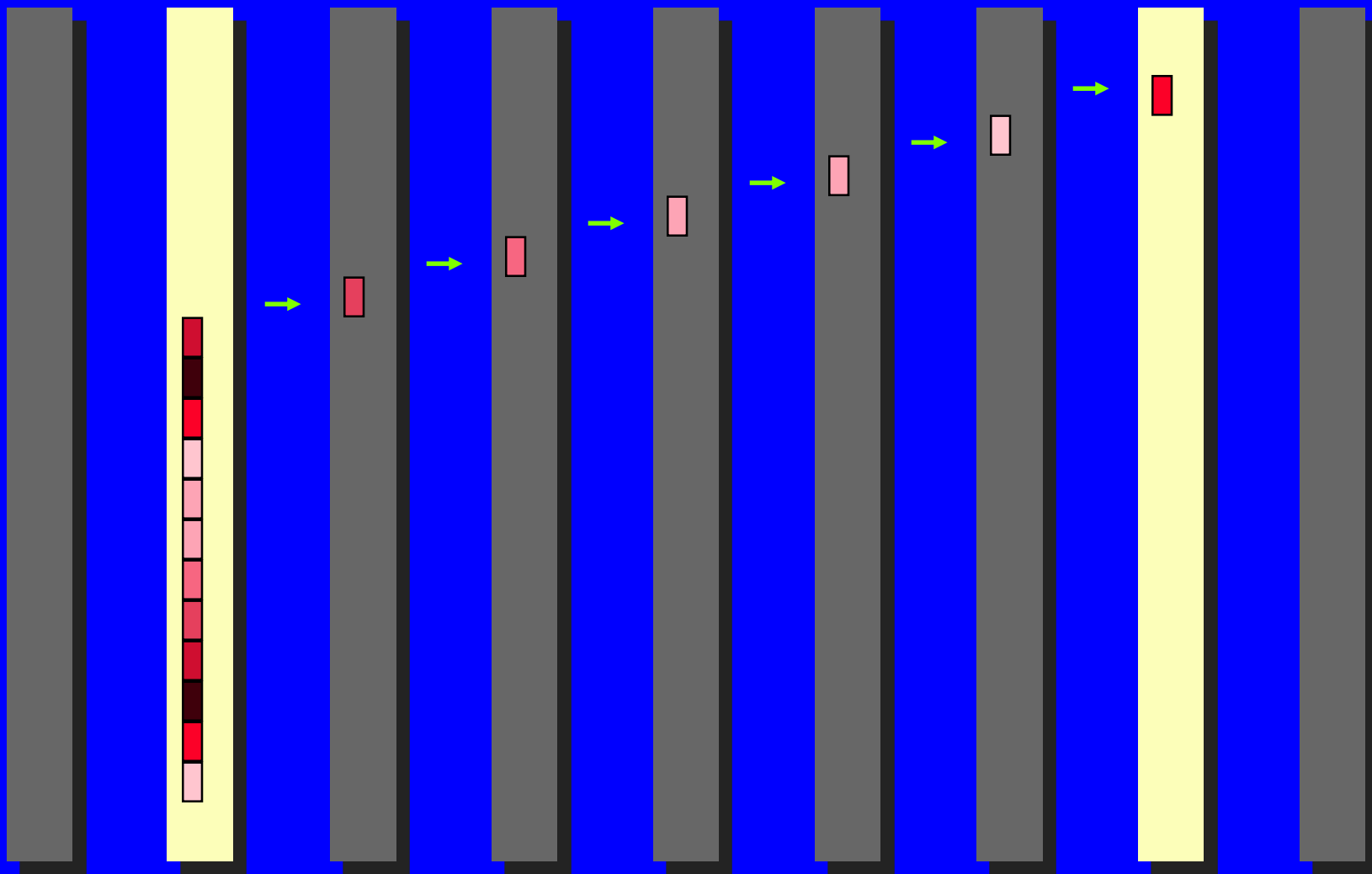


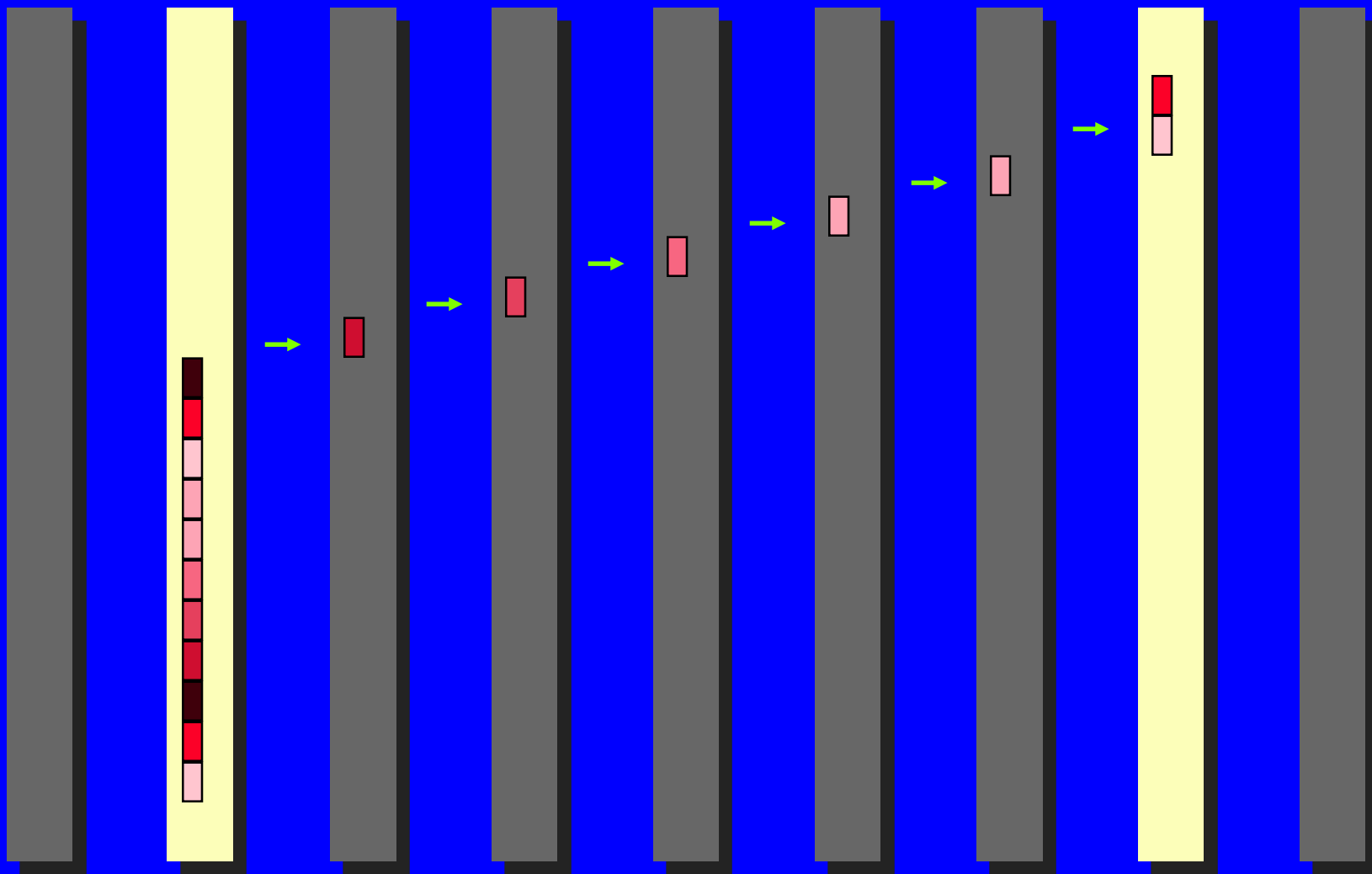


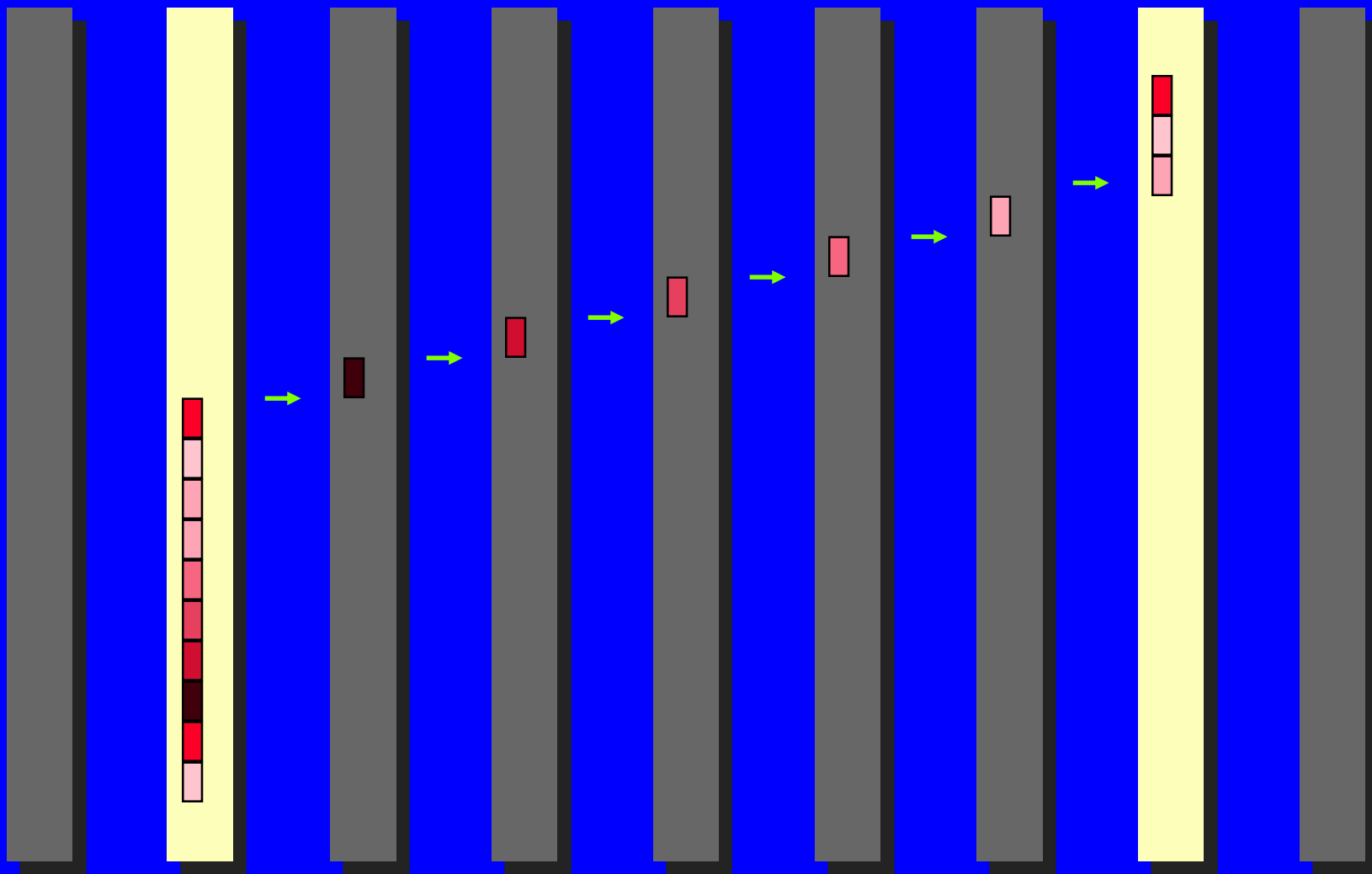


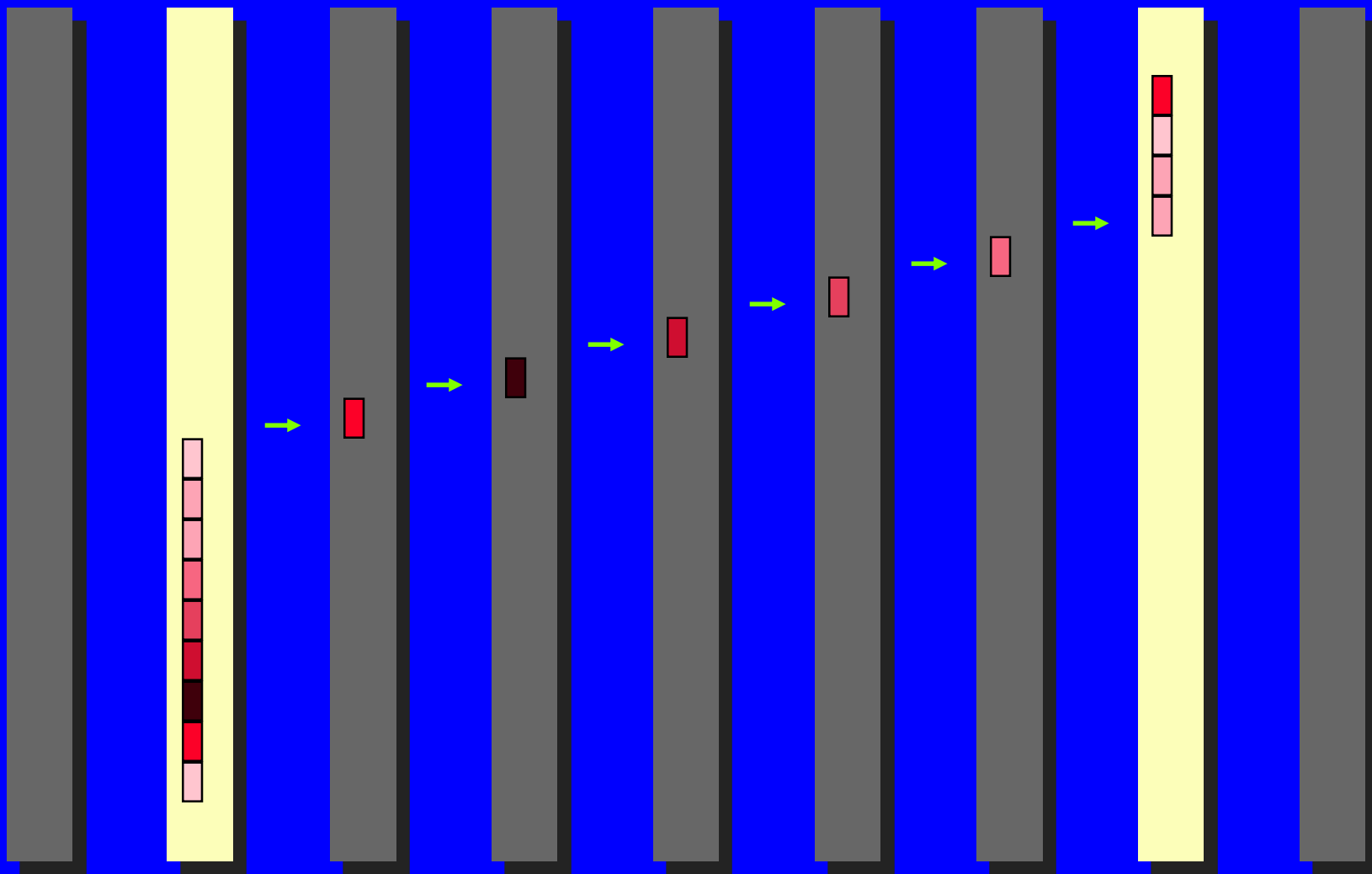


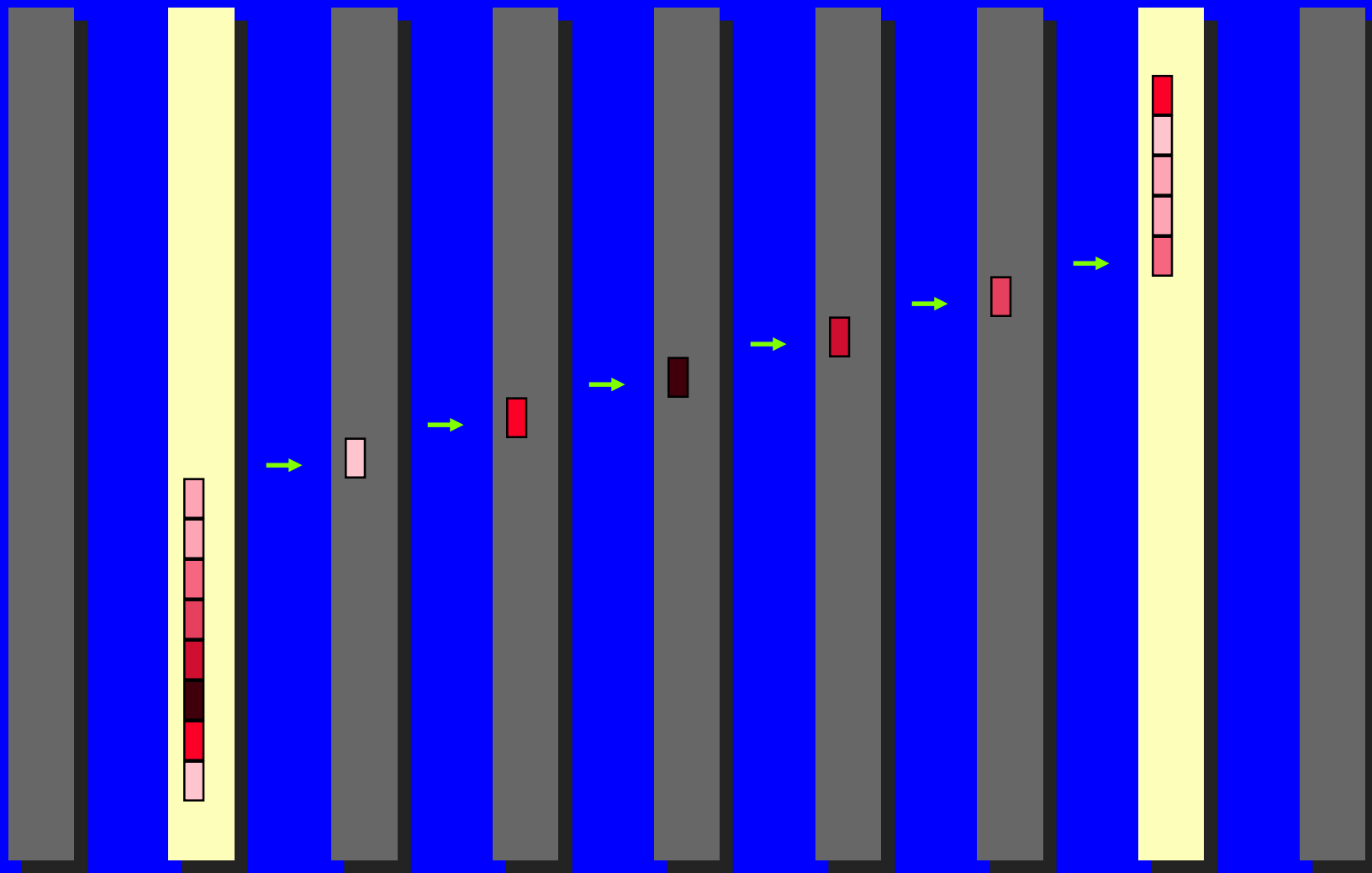


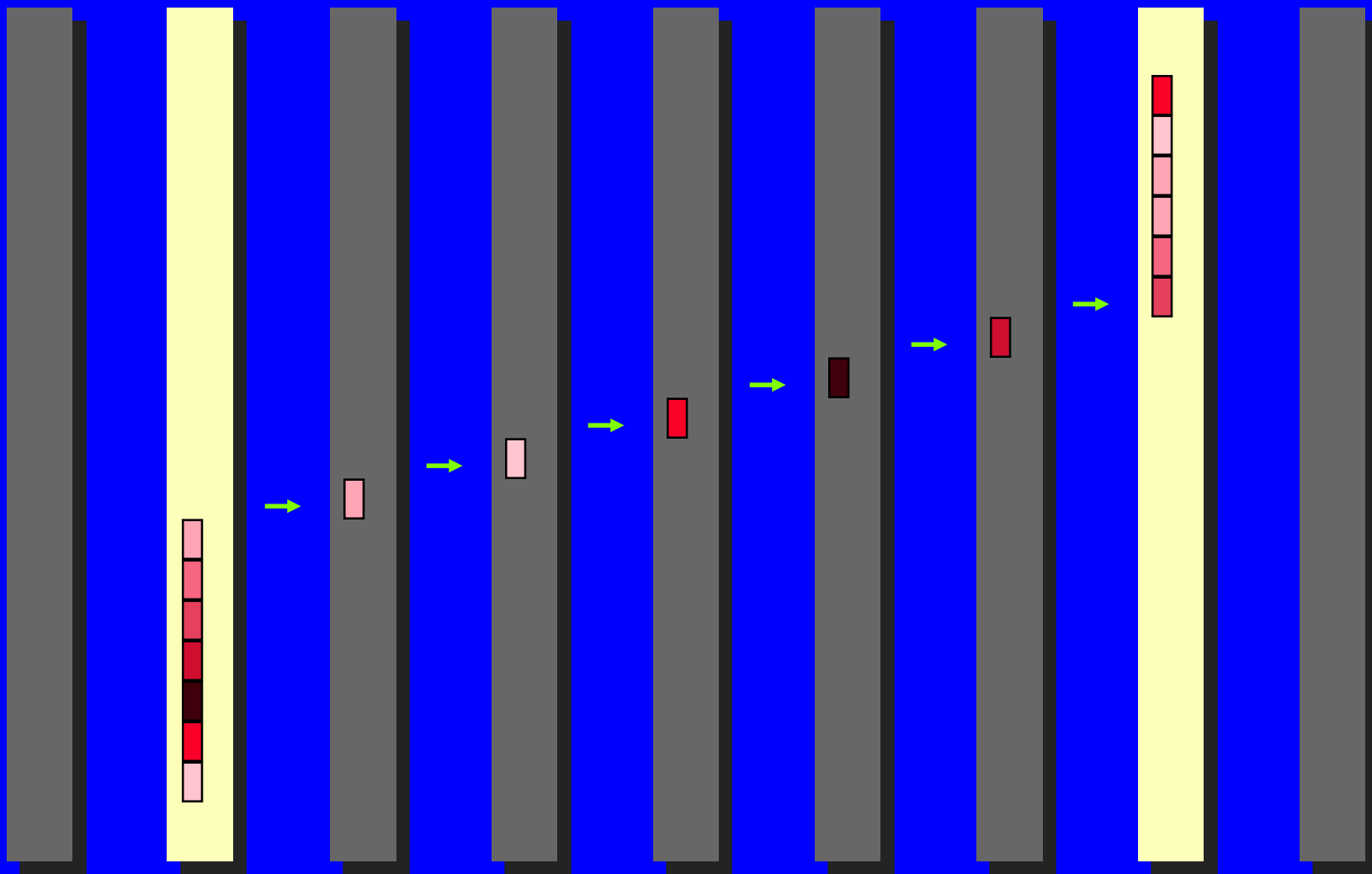


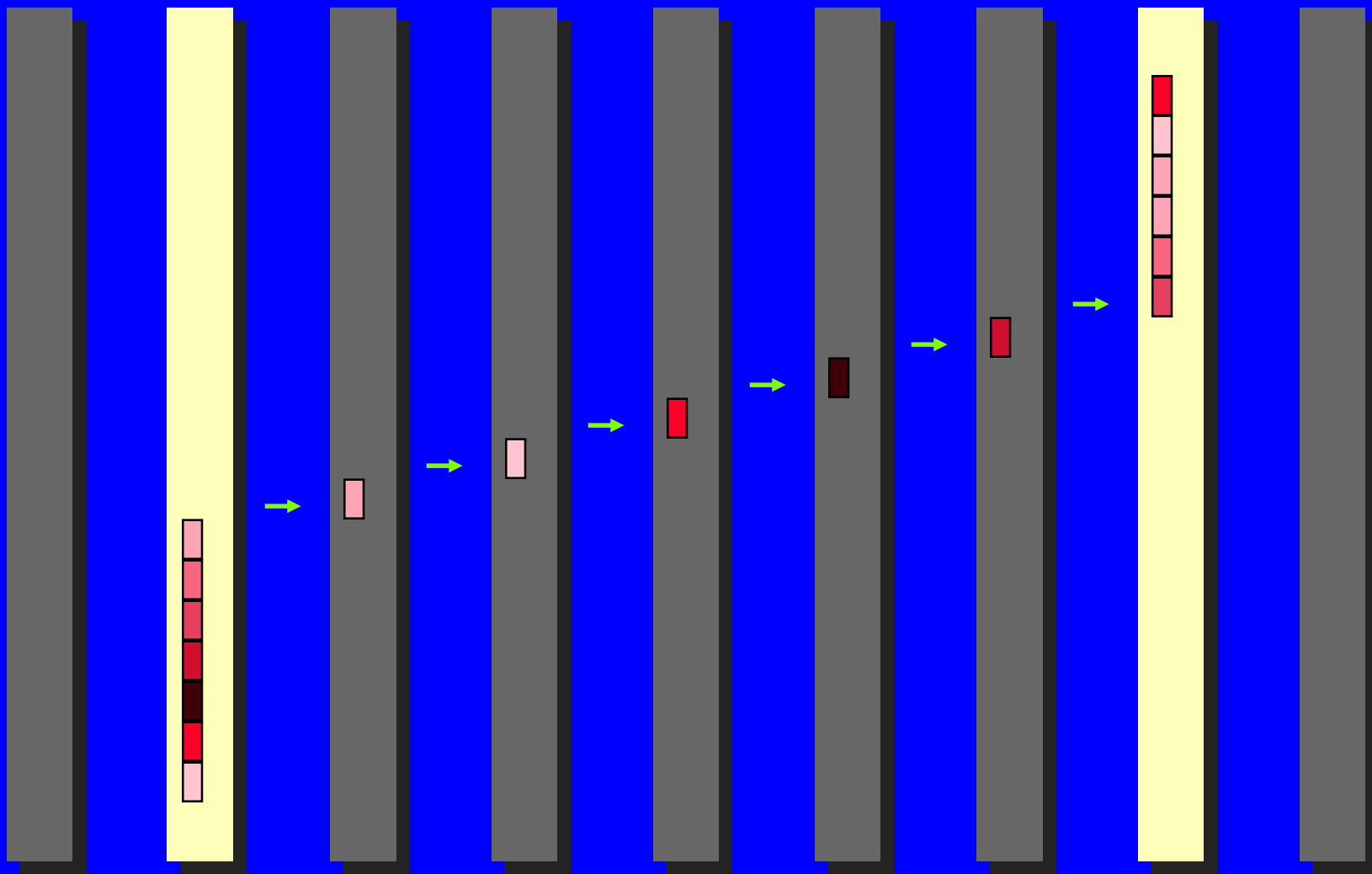


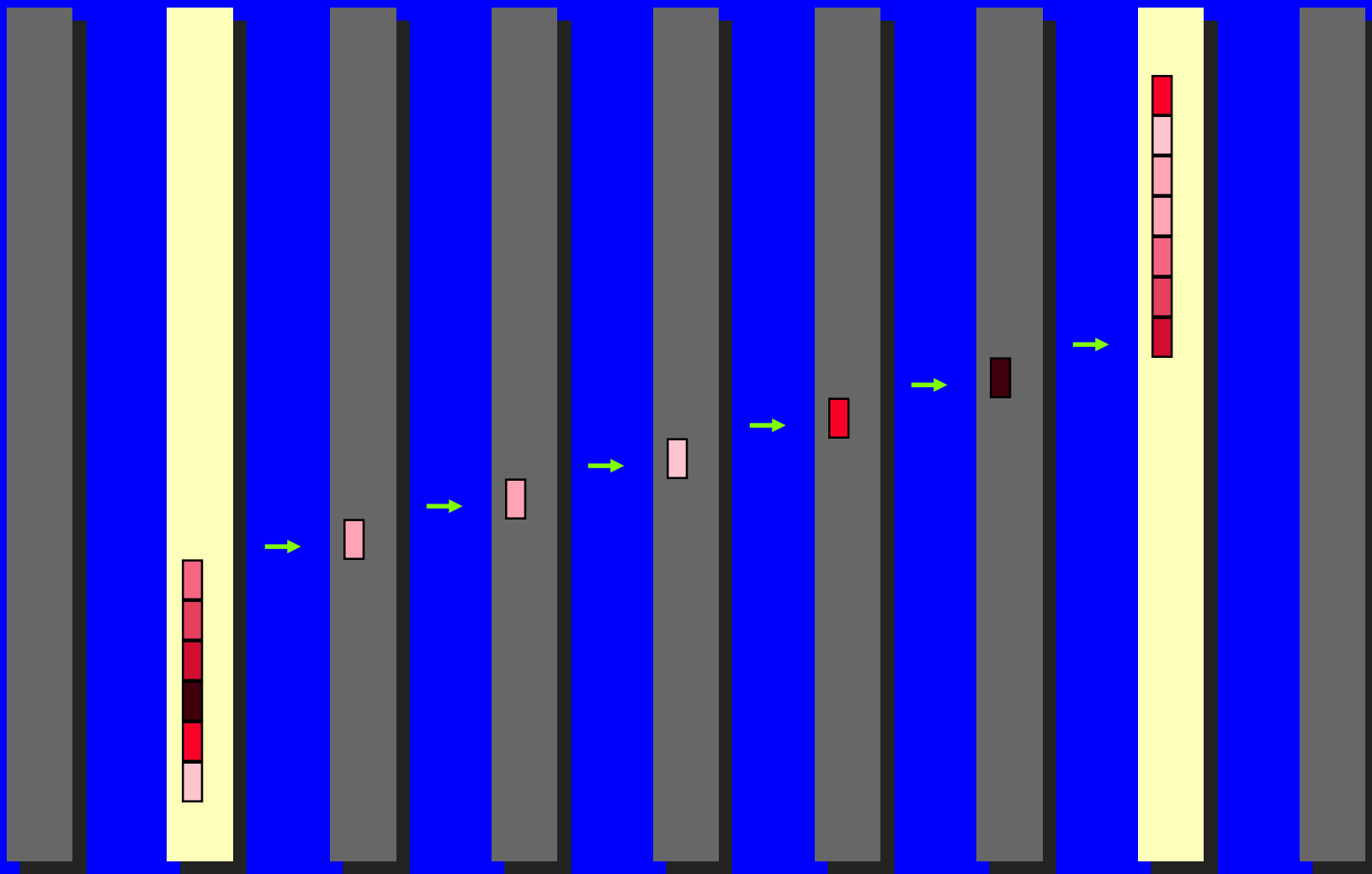


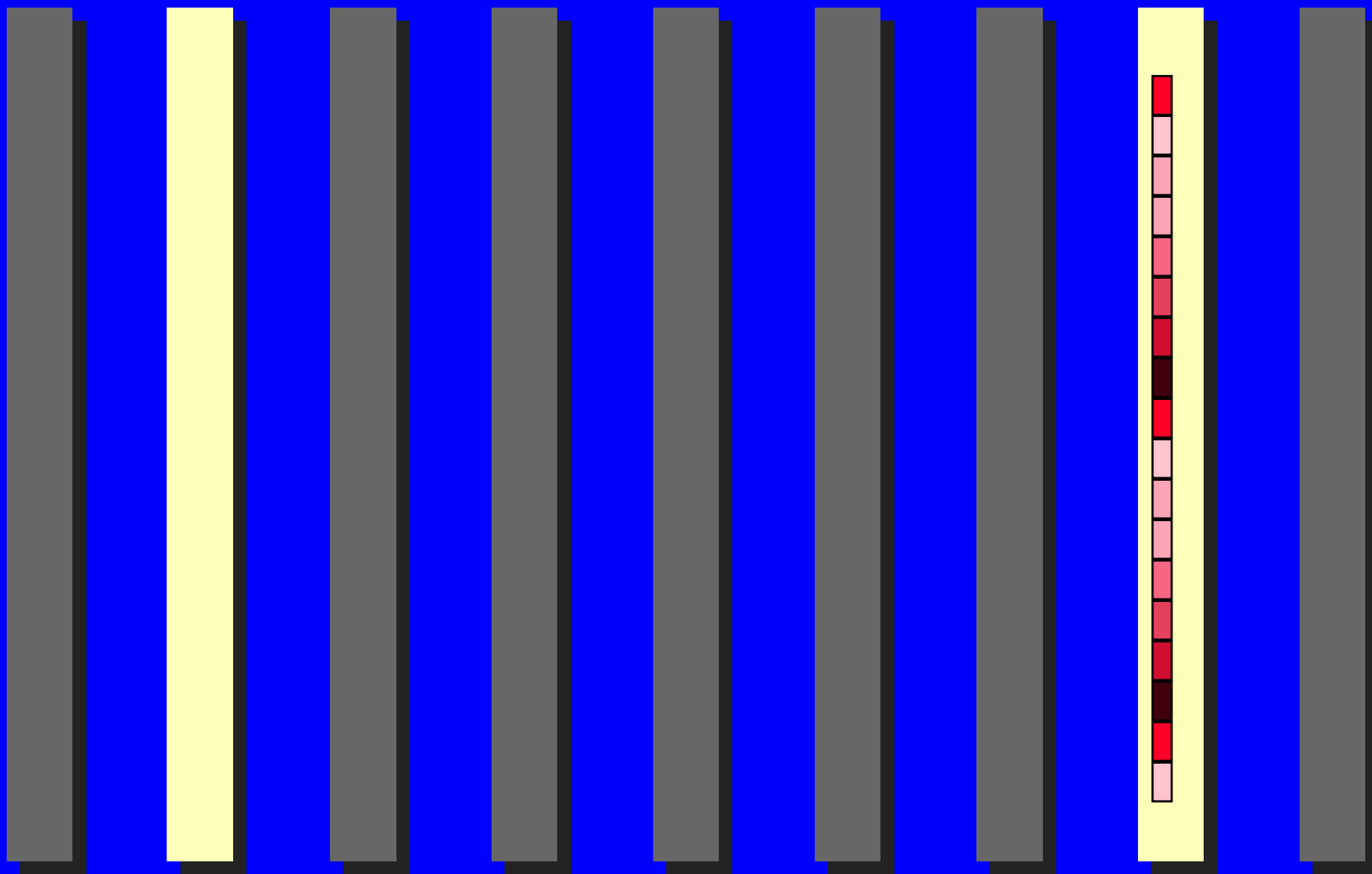












The Cost of Communication

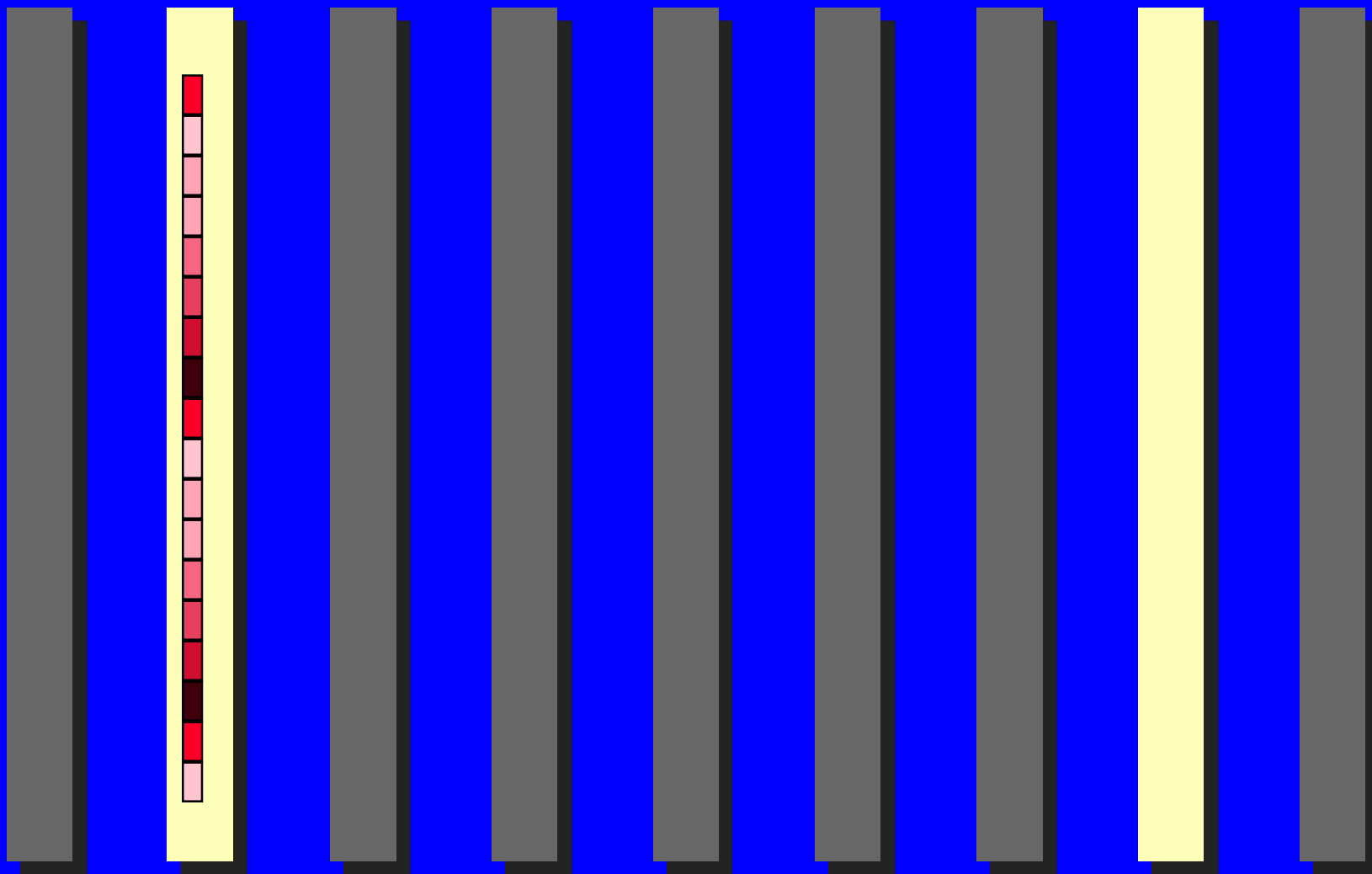
- send a message of length n over d links
- k packets
- Cost:

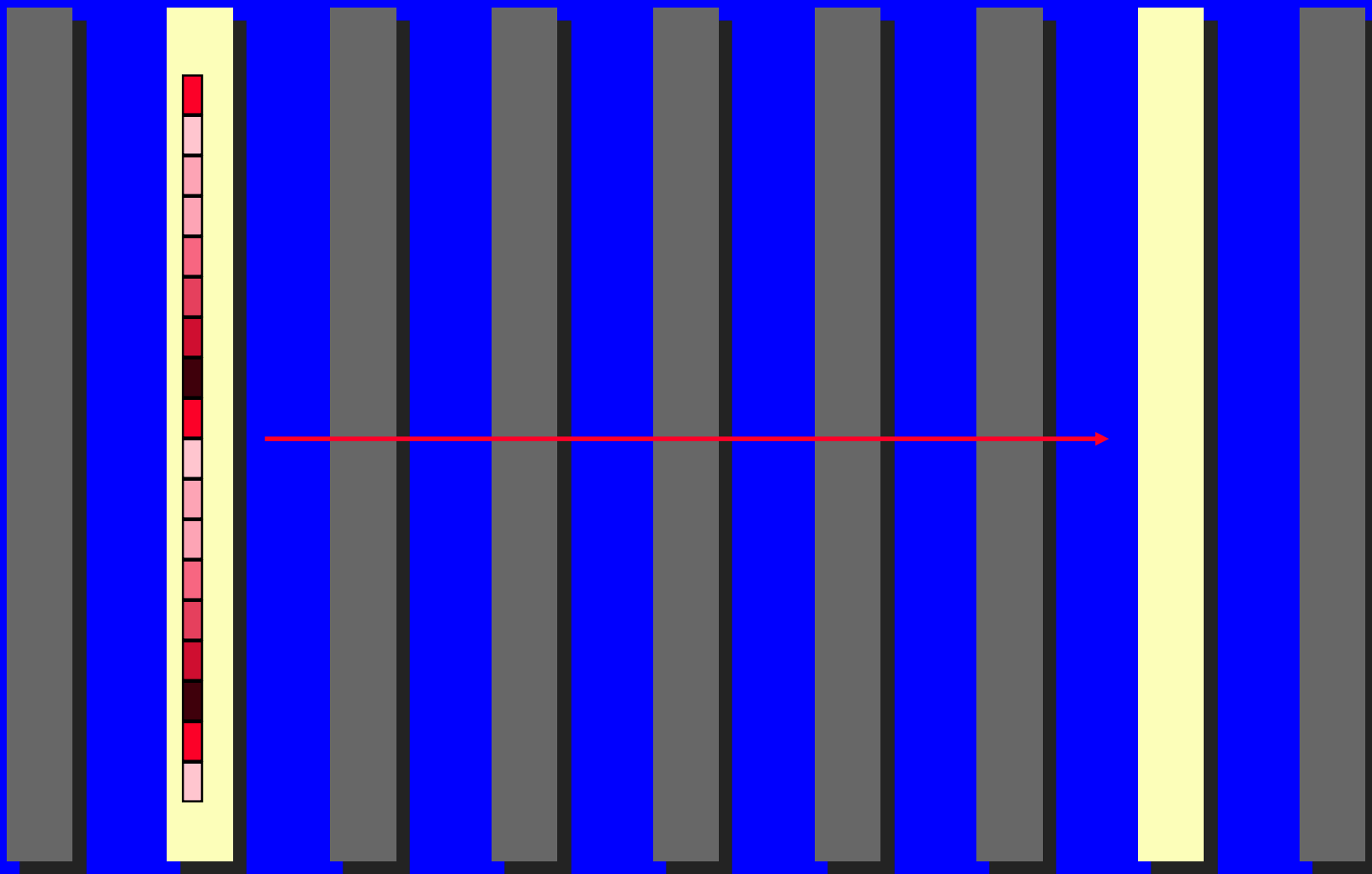
The Cost of Communication

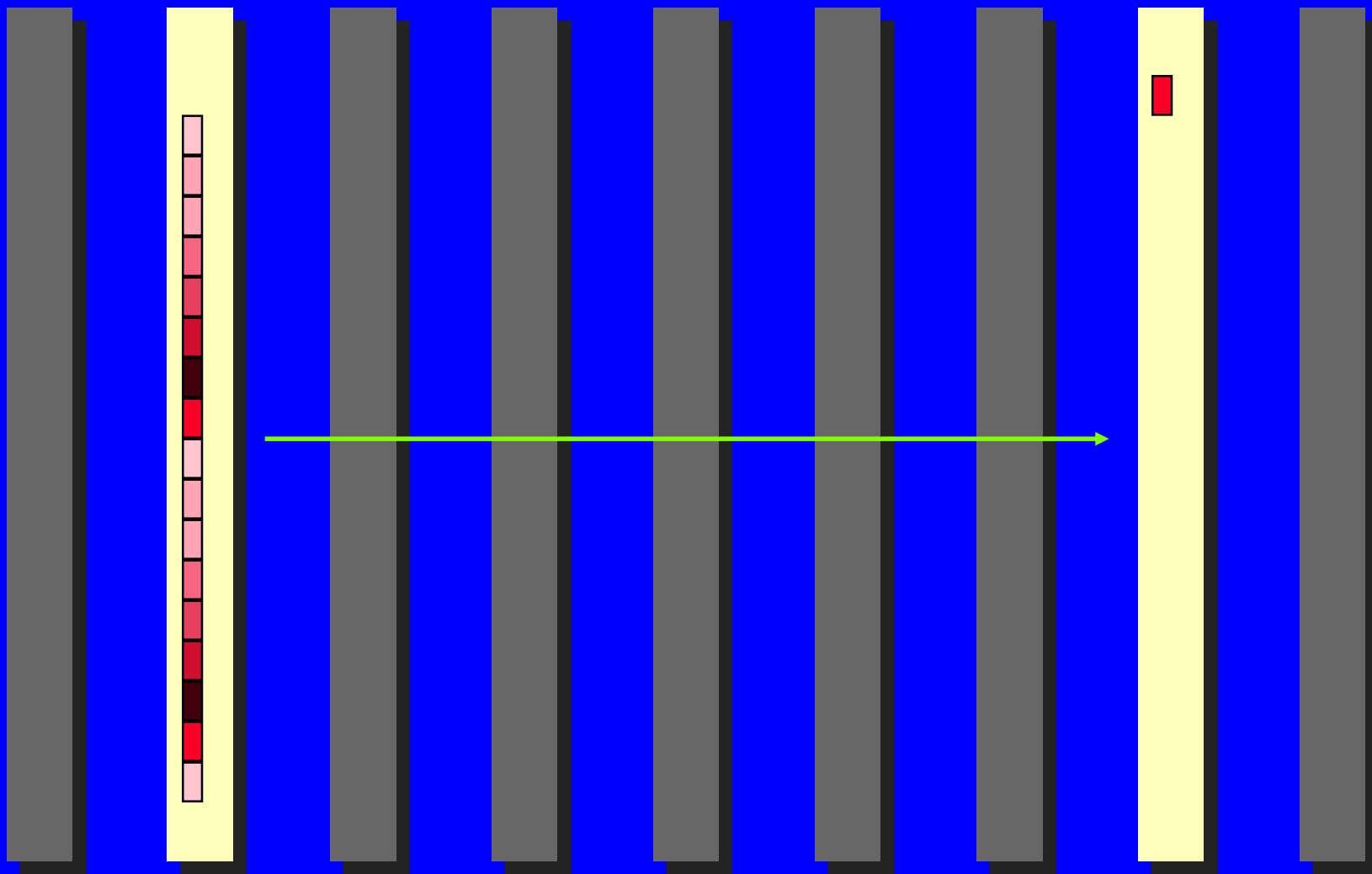
- send a message of length n over d links
- k packets
- Cost:

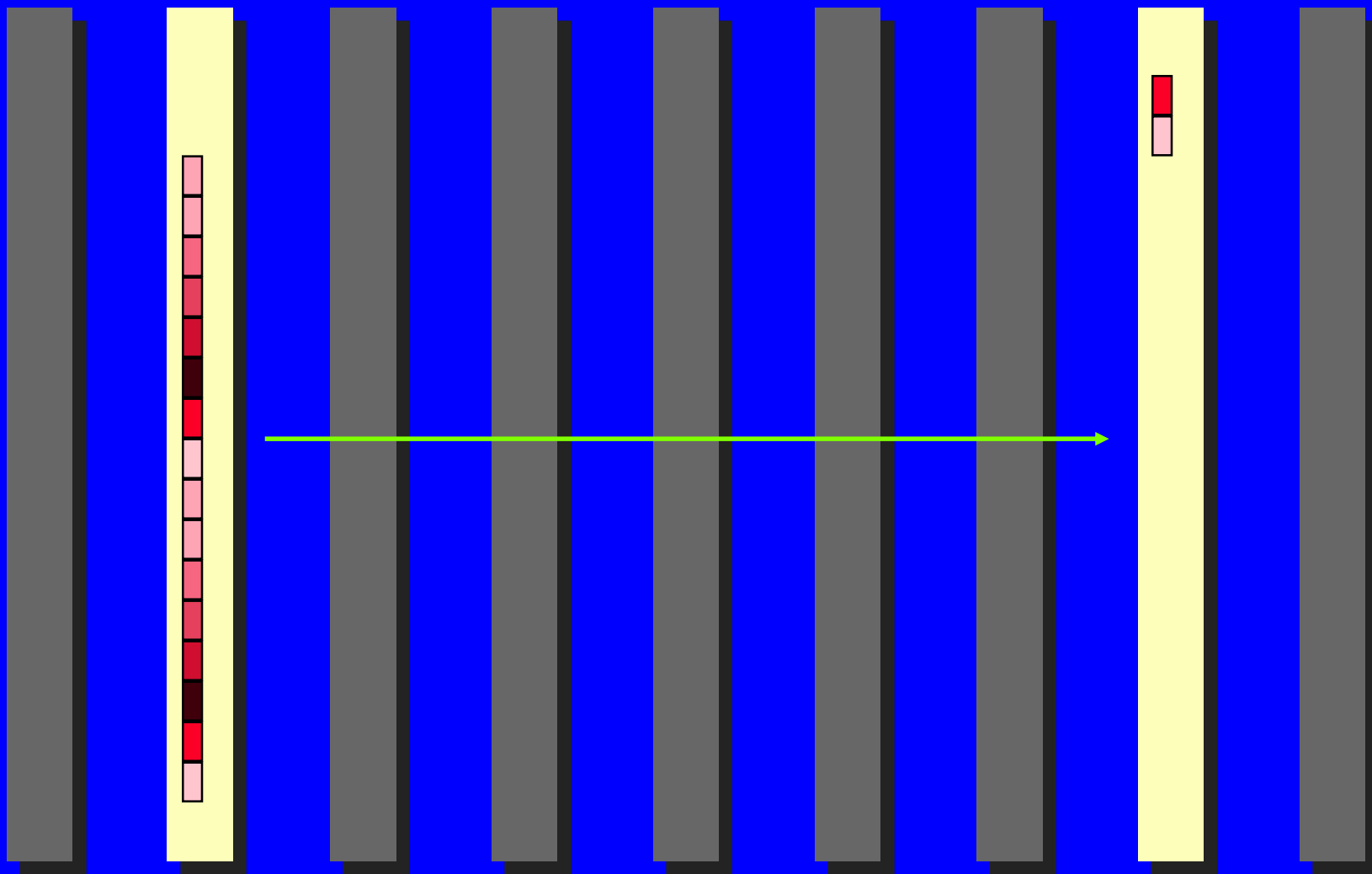
$$\begin{aligned} & \alpha + d \left(\alpha_{net} + \frac{n}{k} \beta \right) + (k-1) \left(\alpha_{net} + \frac{n}{k} \beta \right) \\ & = \\ & \alpha + n\beta + (d+k-1)\alpha_{net} + \frac{d-1}{k} n\beta \\ & \approx \\ & \alpha + n\beta \end{aligned}$$

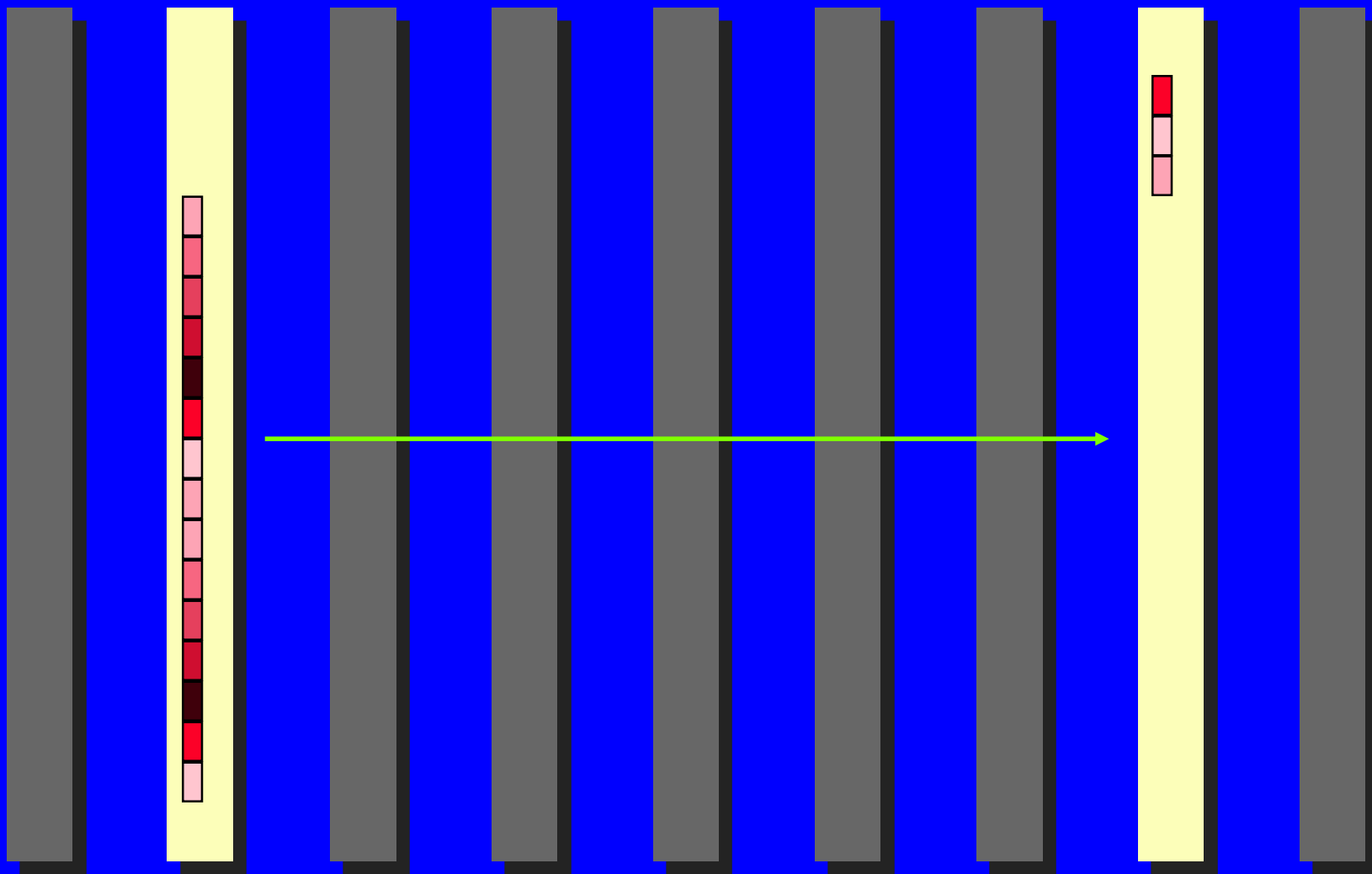
- Example revisited ...

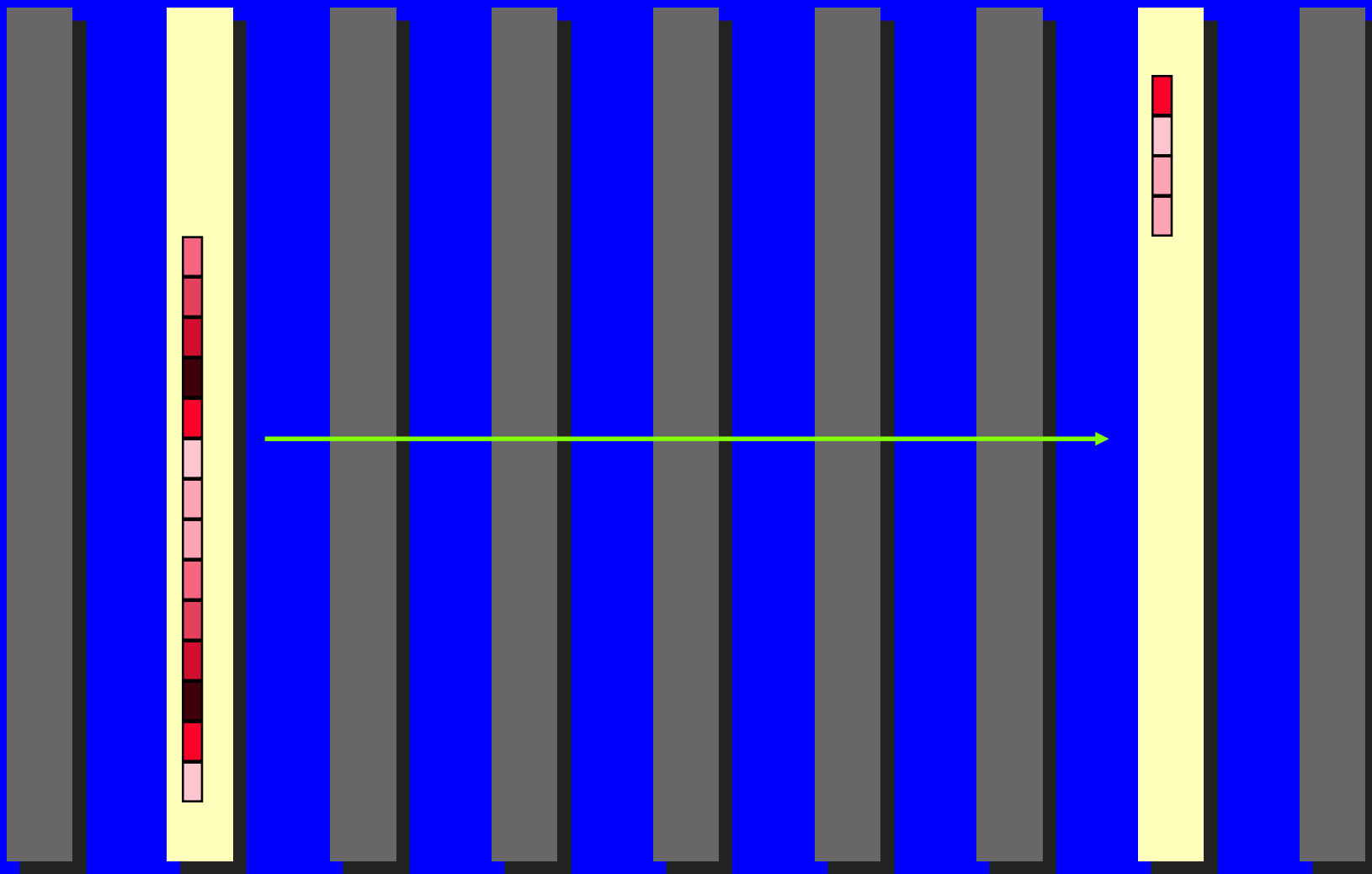


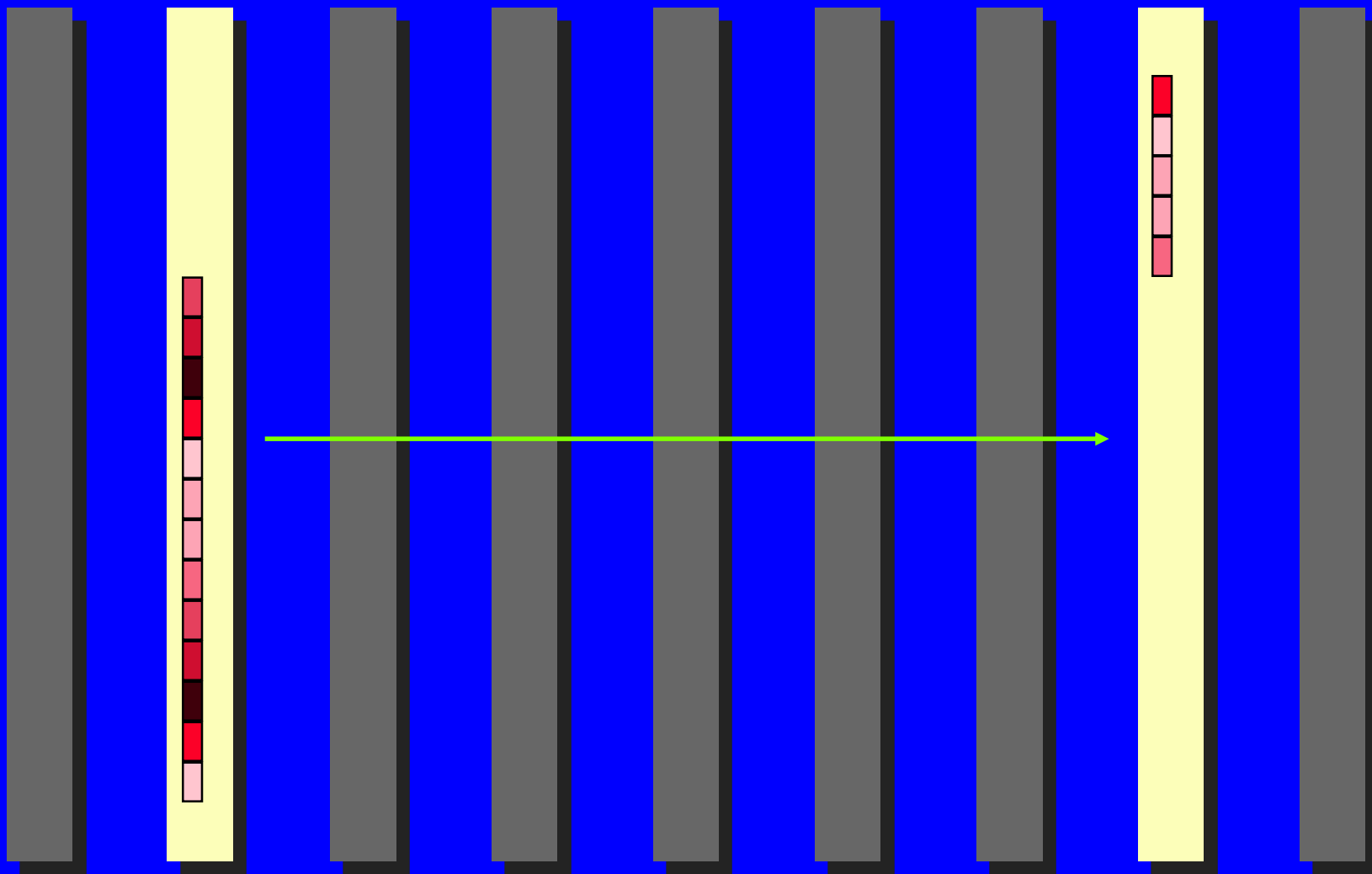


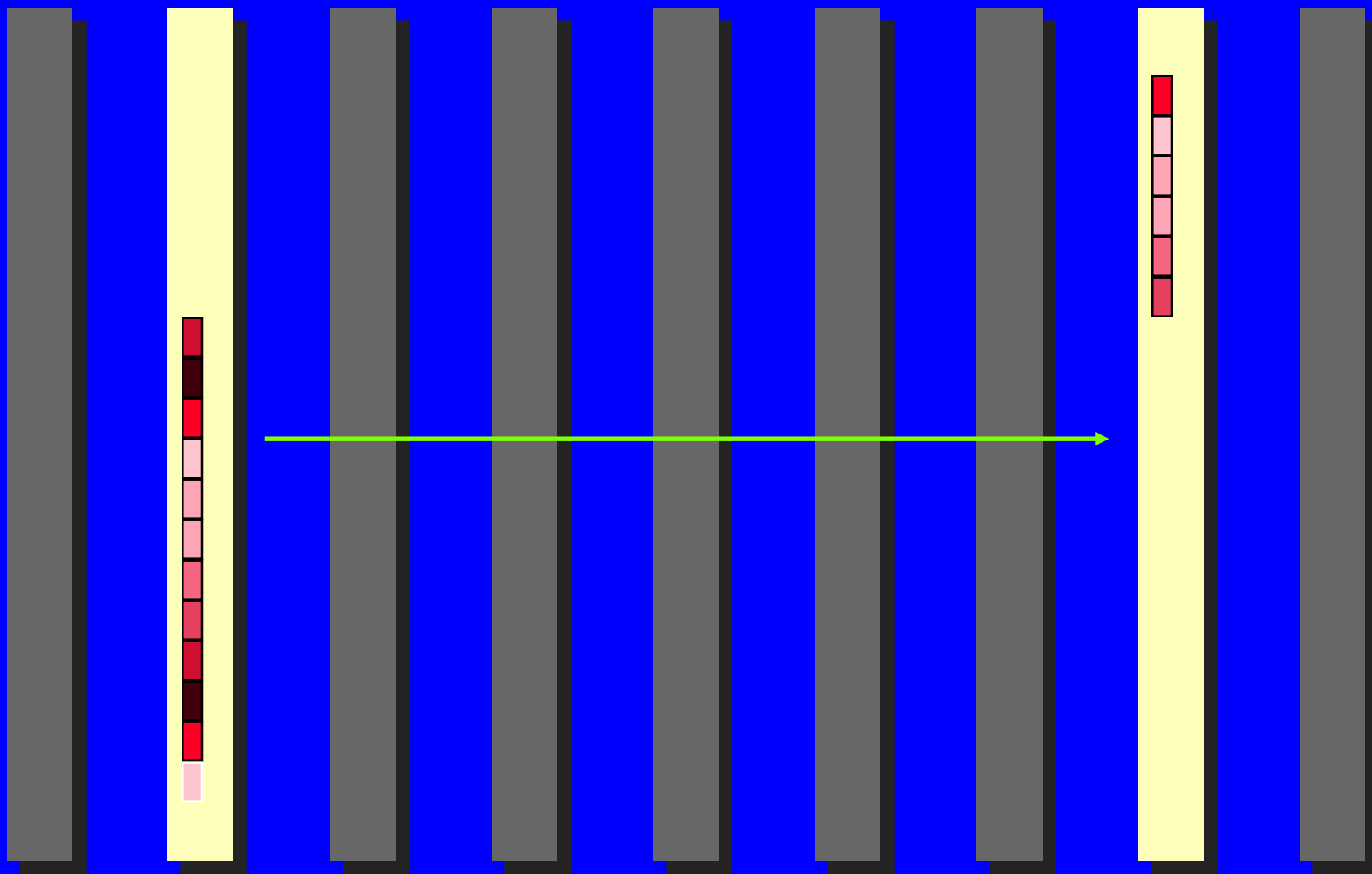


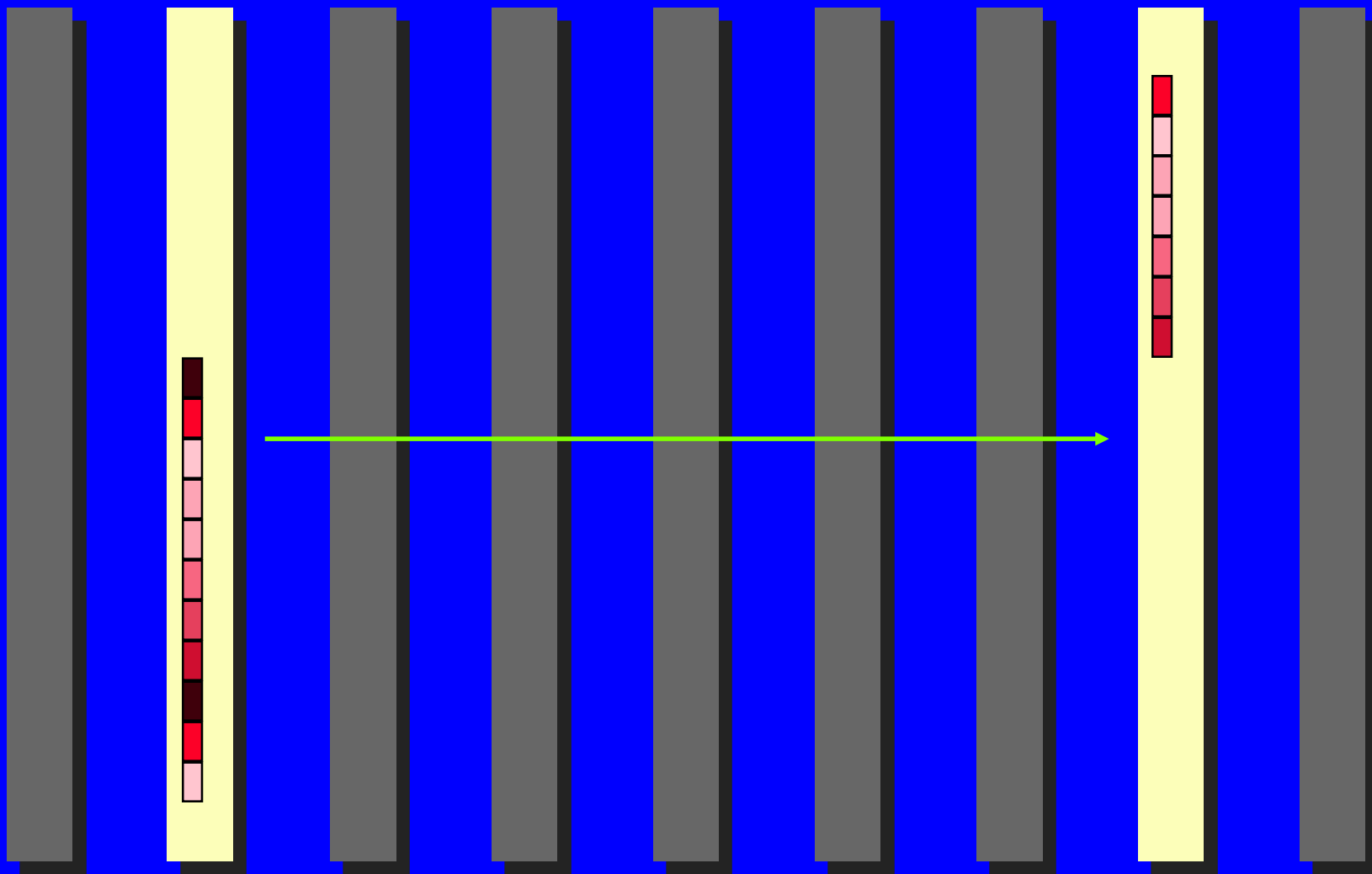


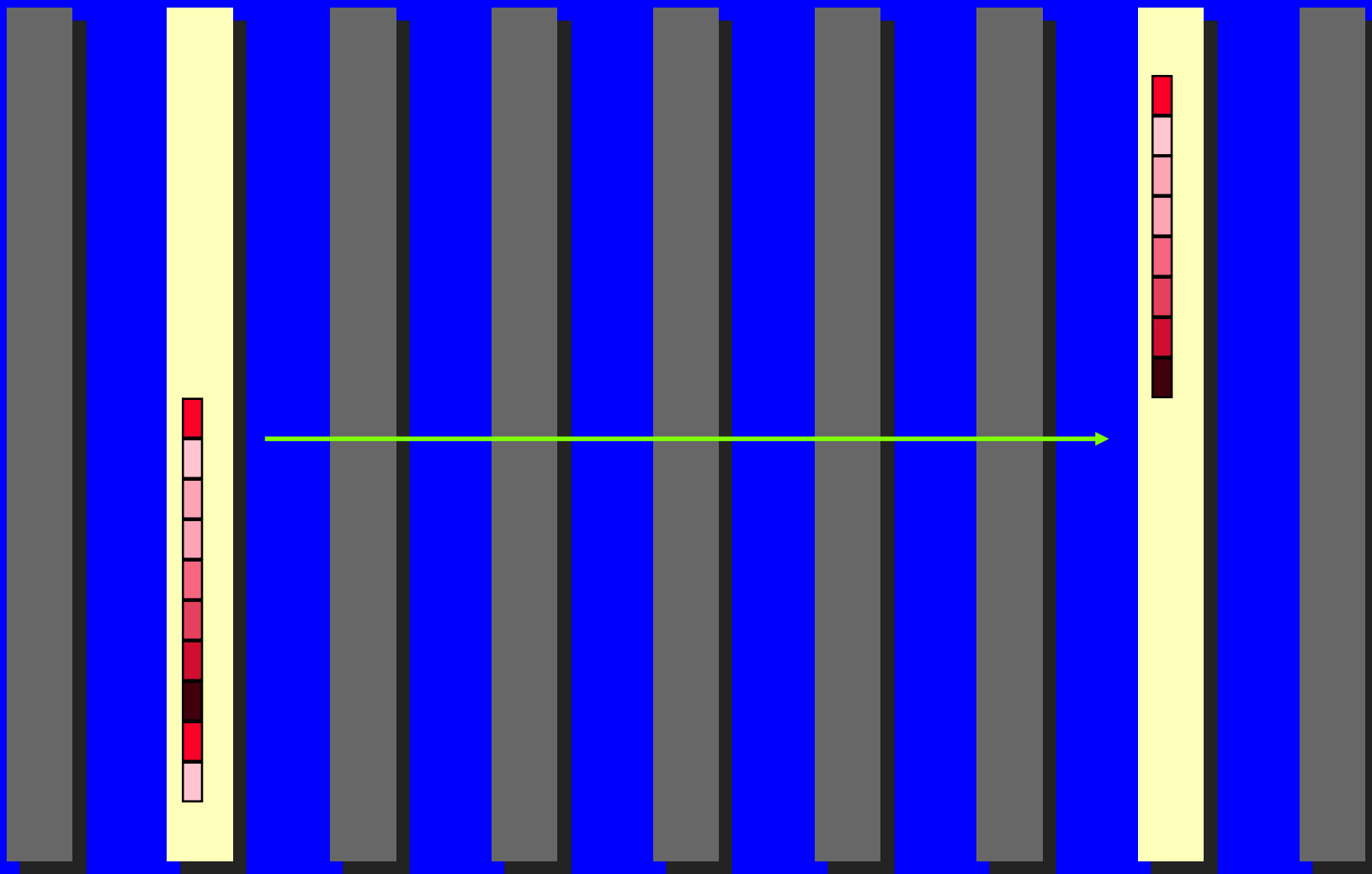


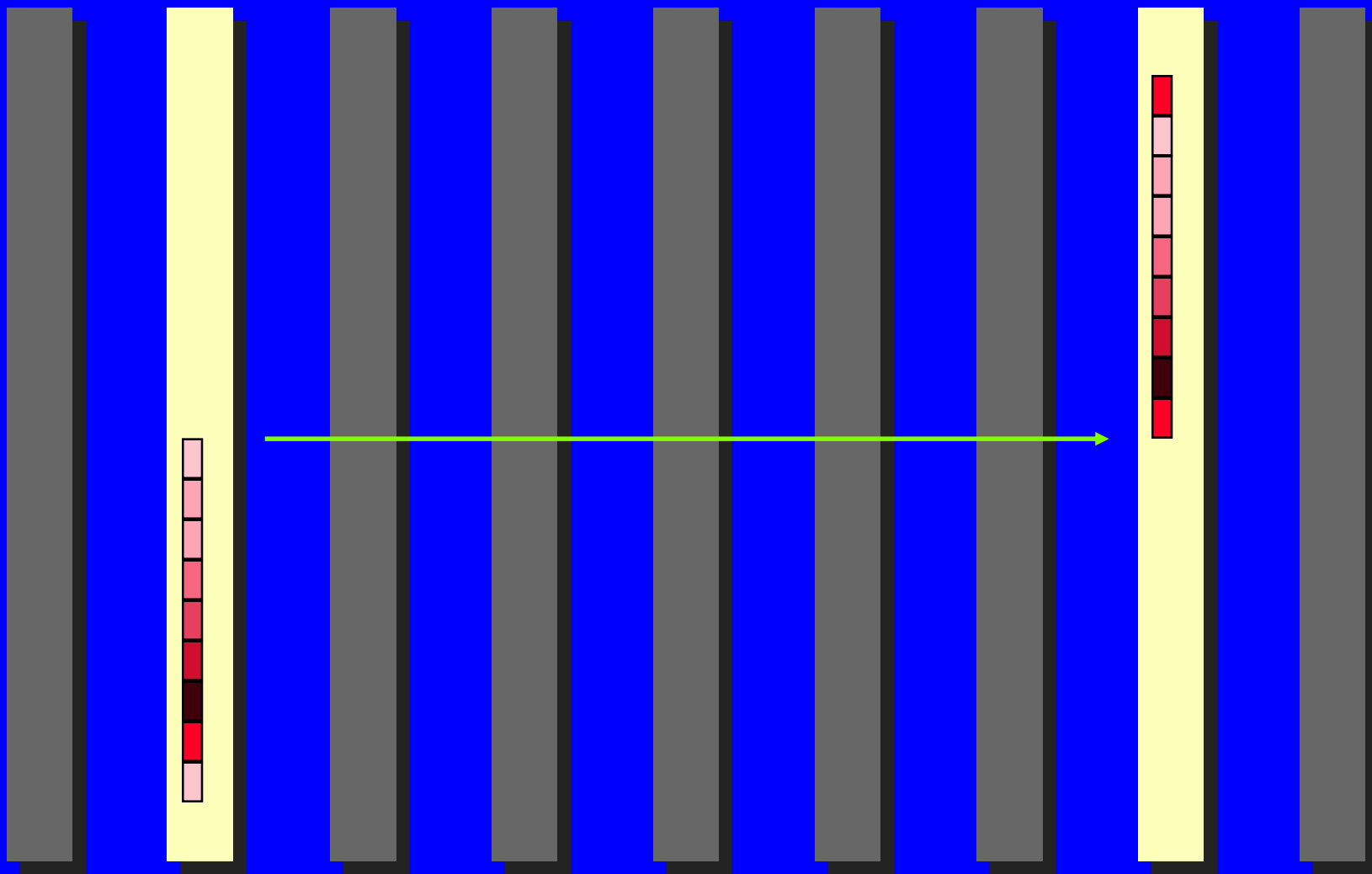


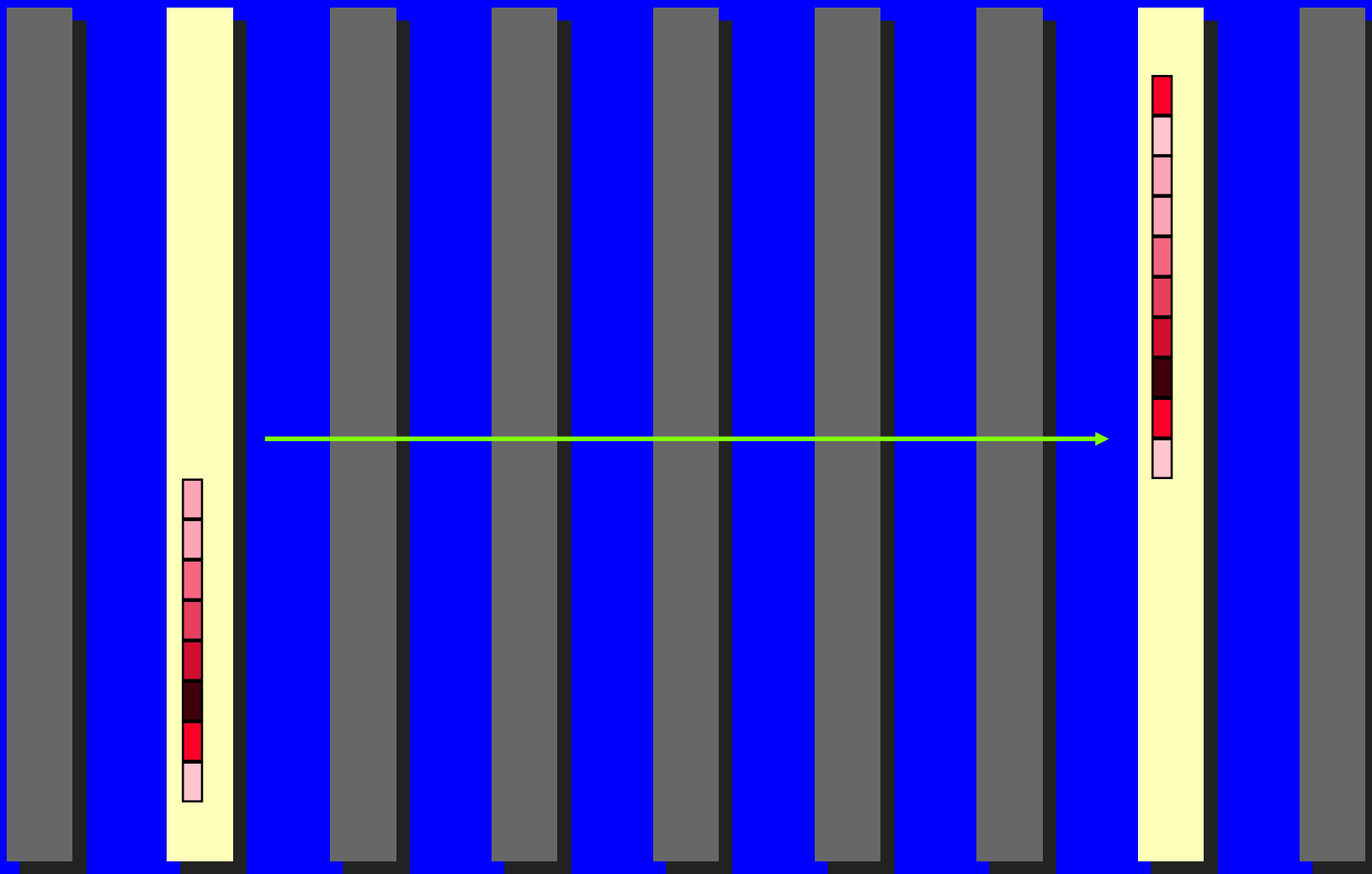


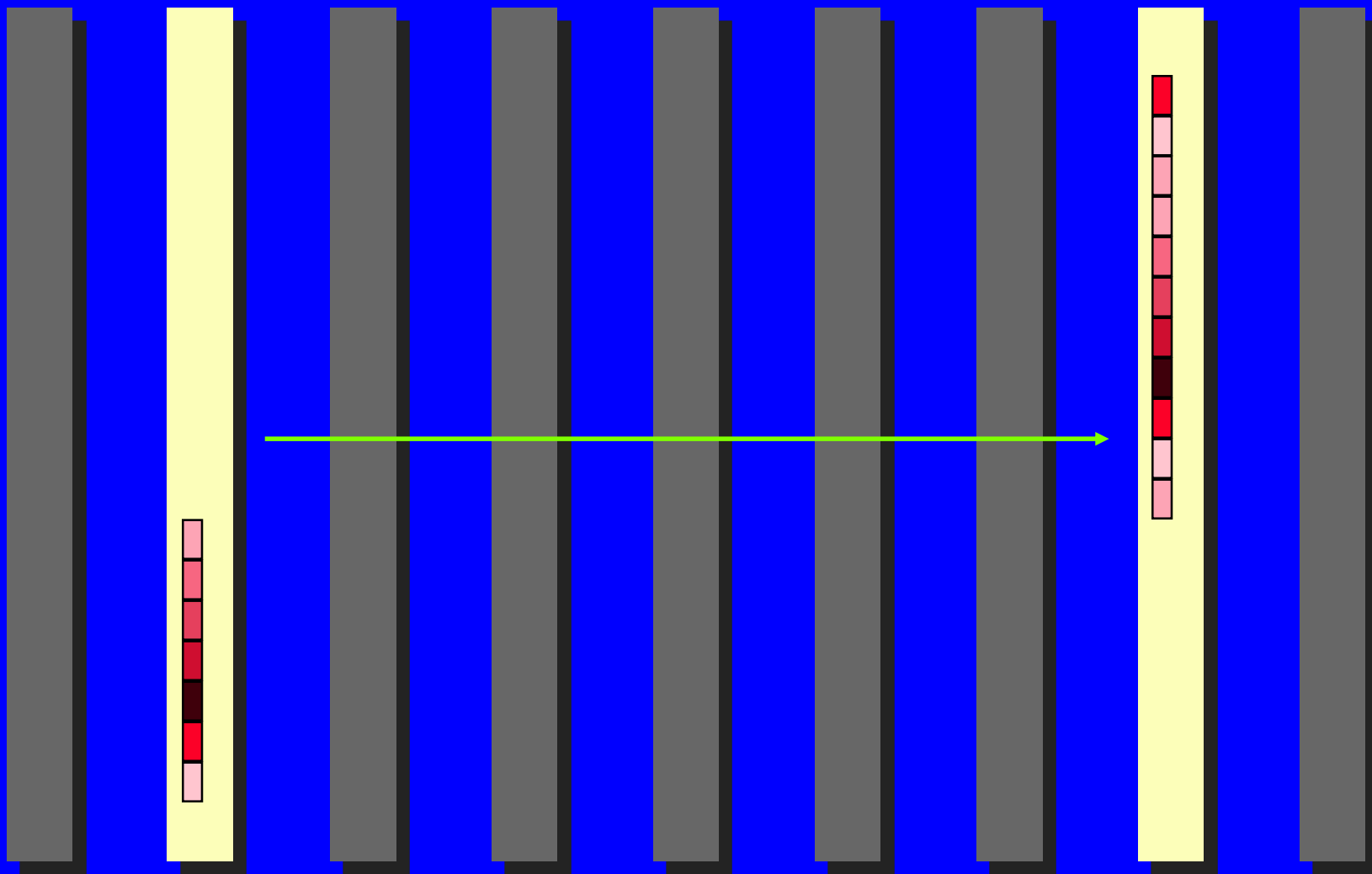


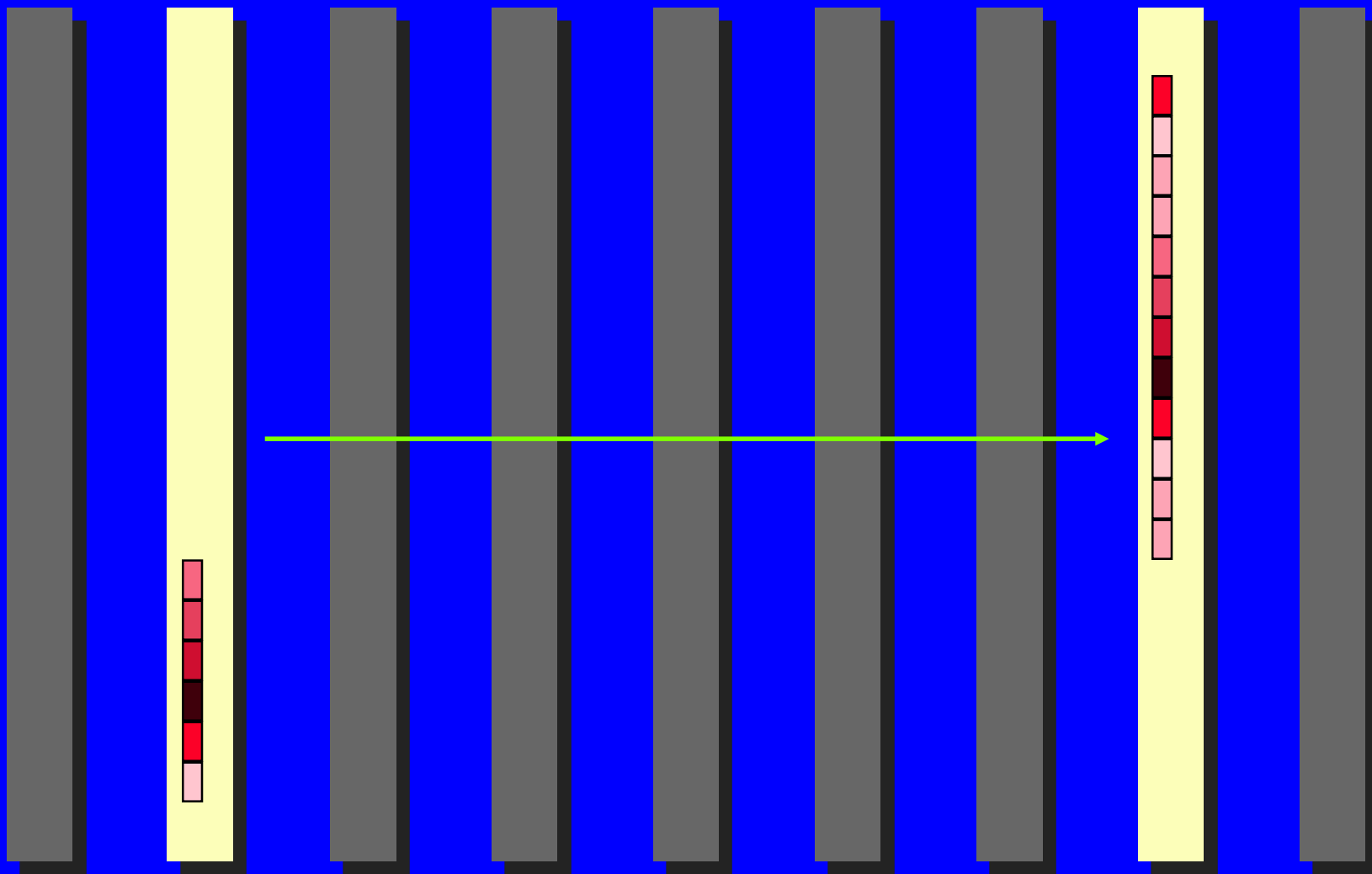


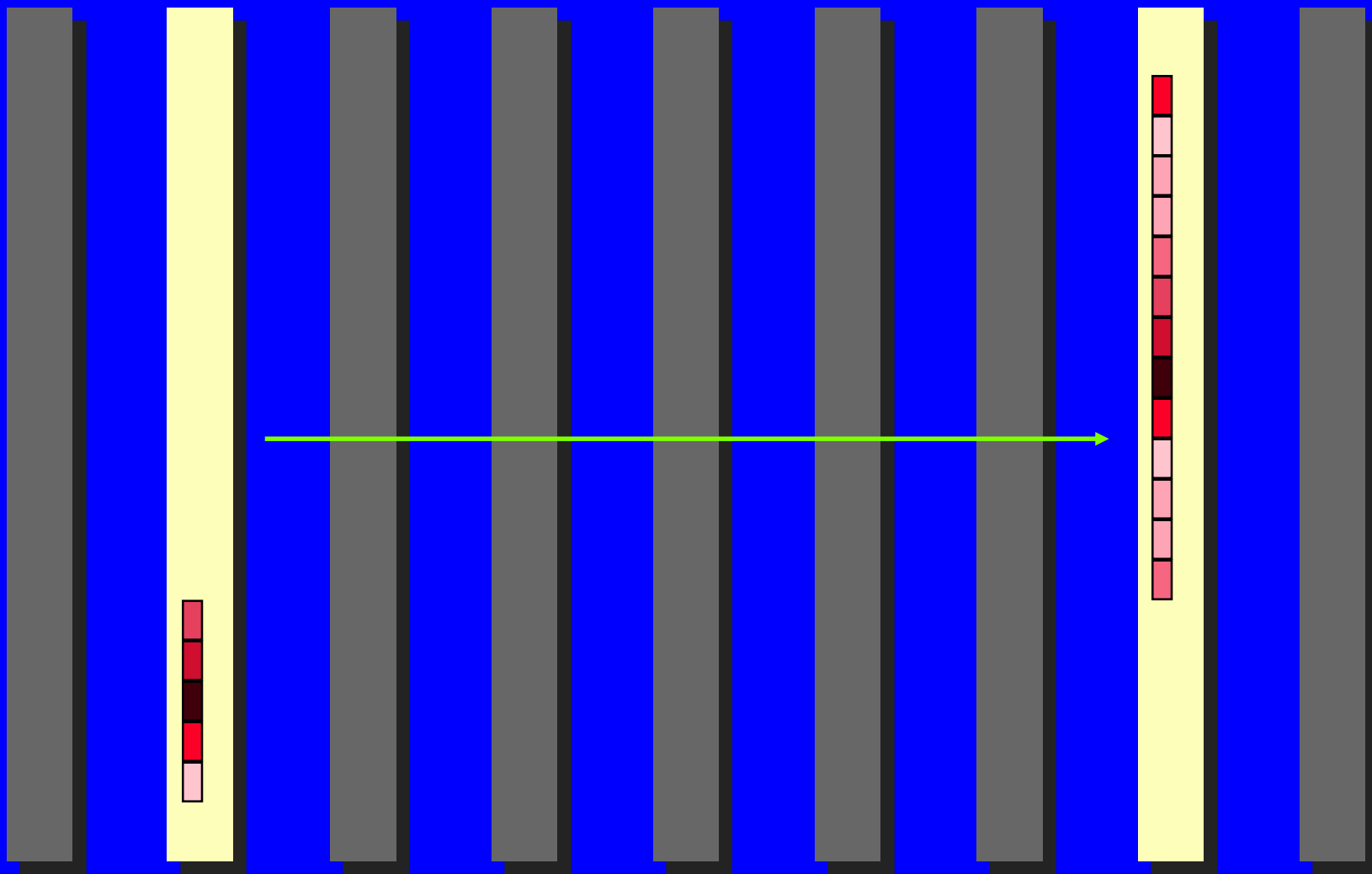


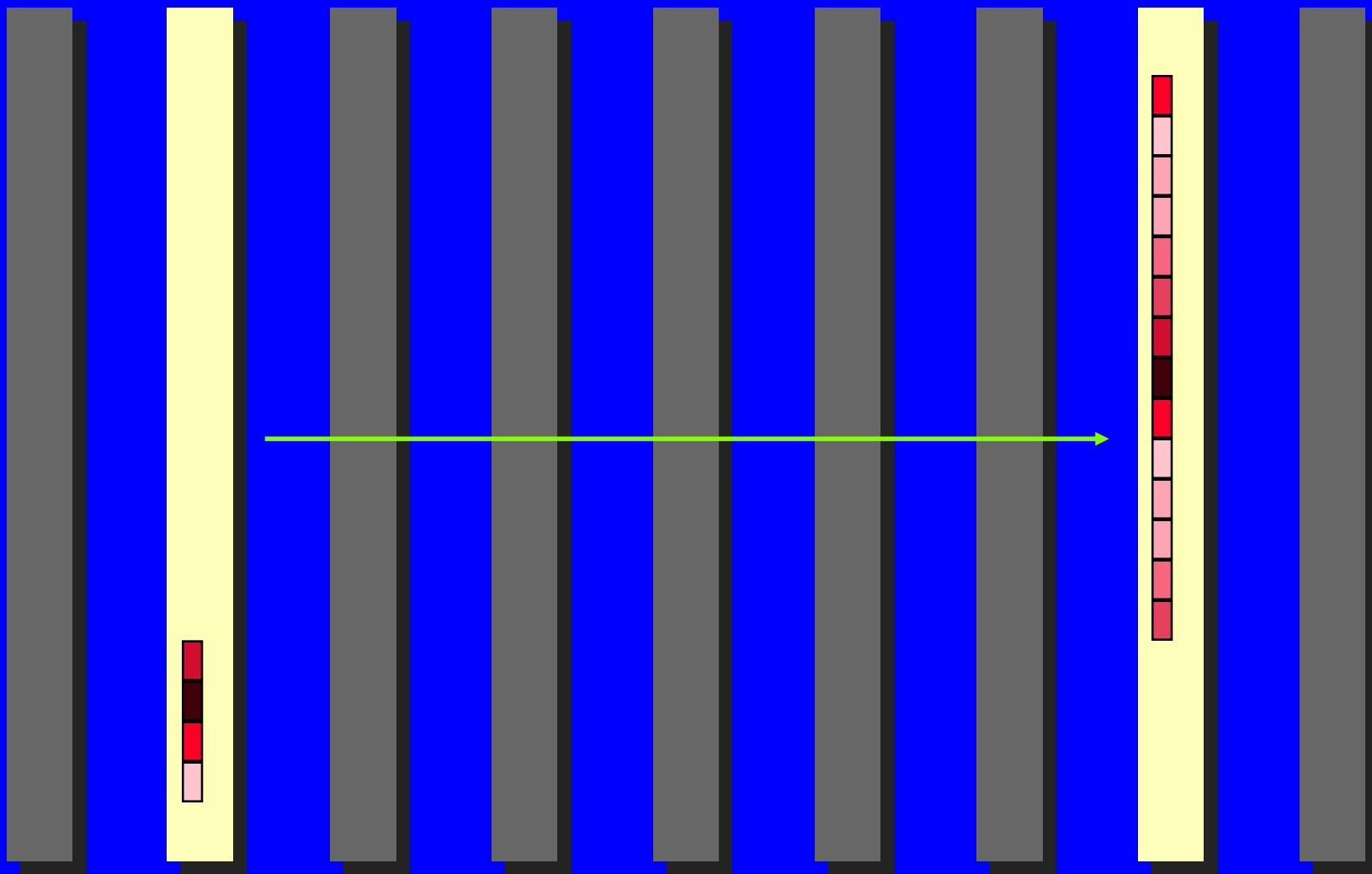


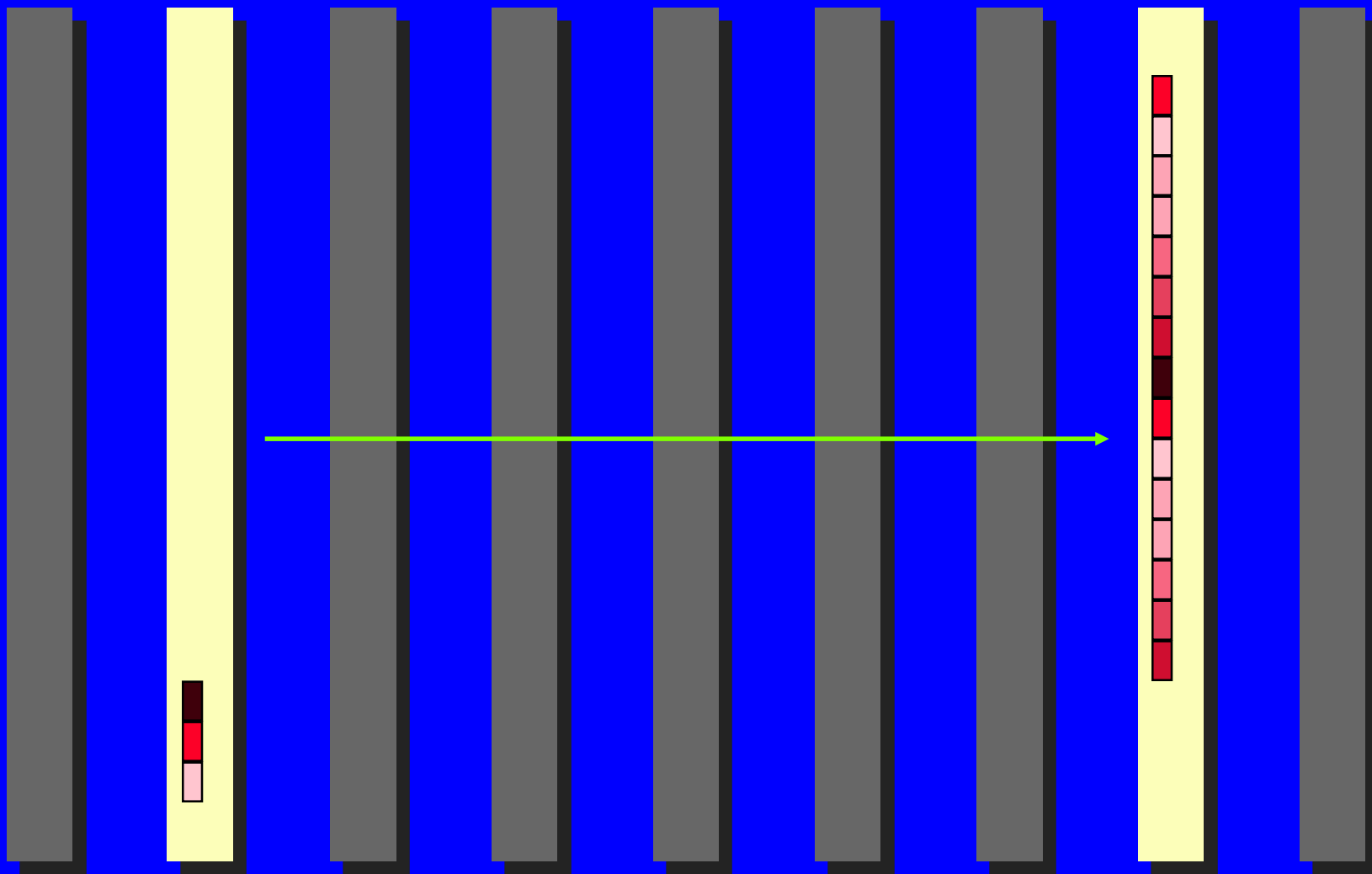


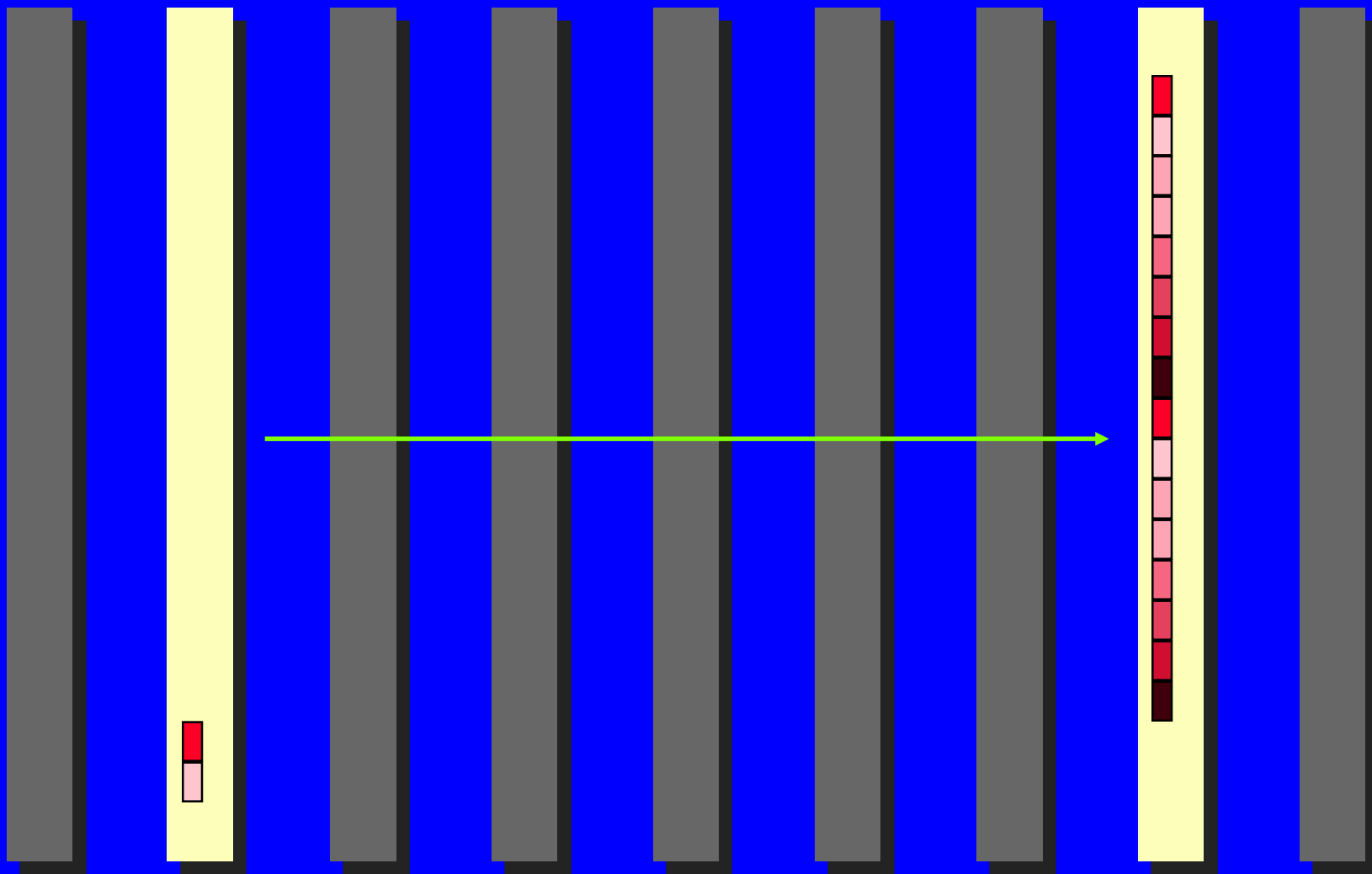


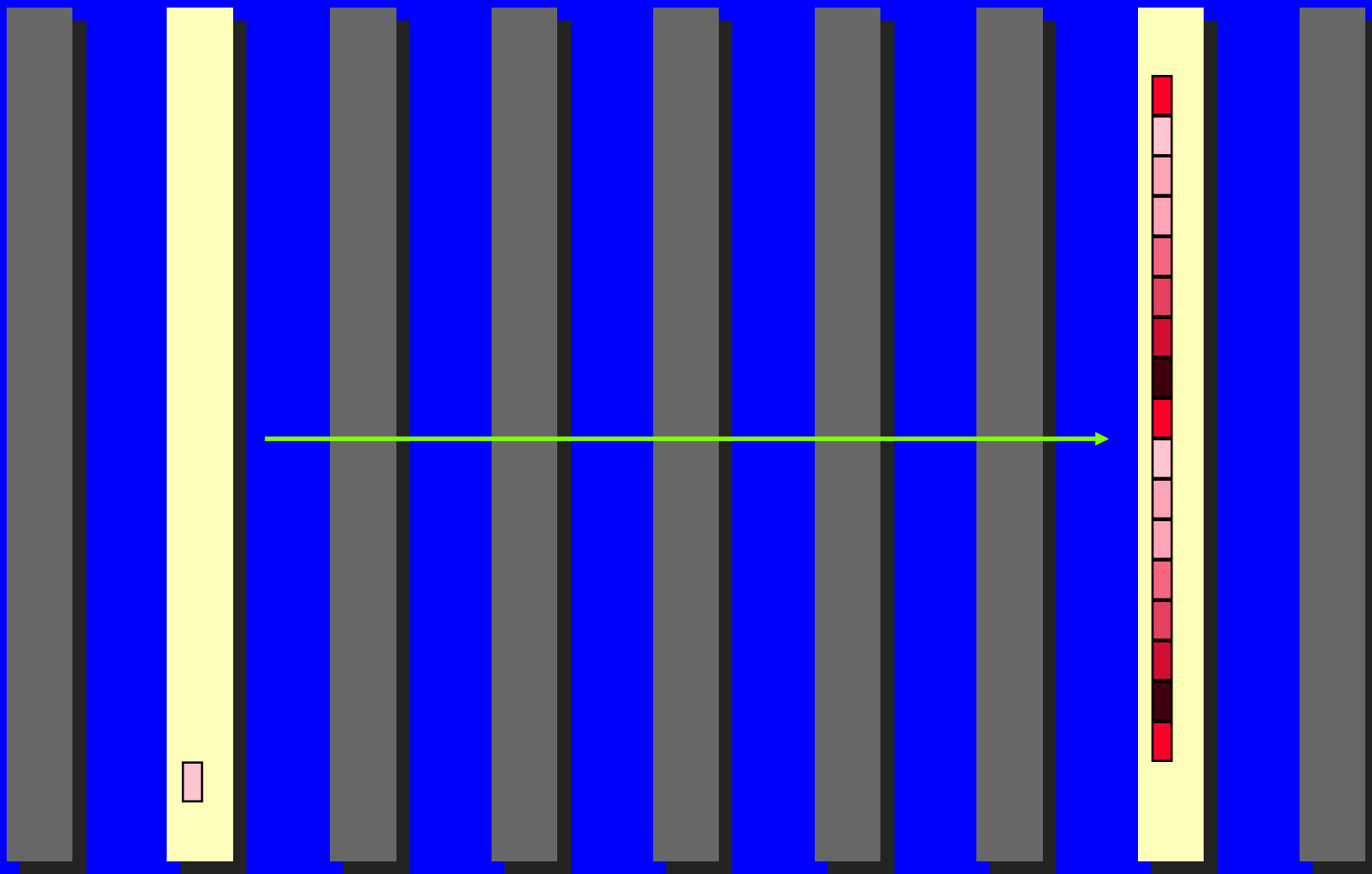


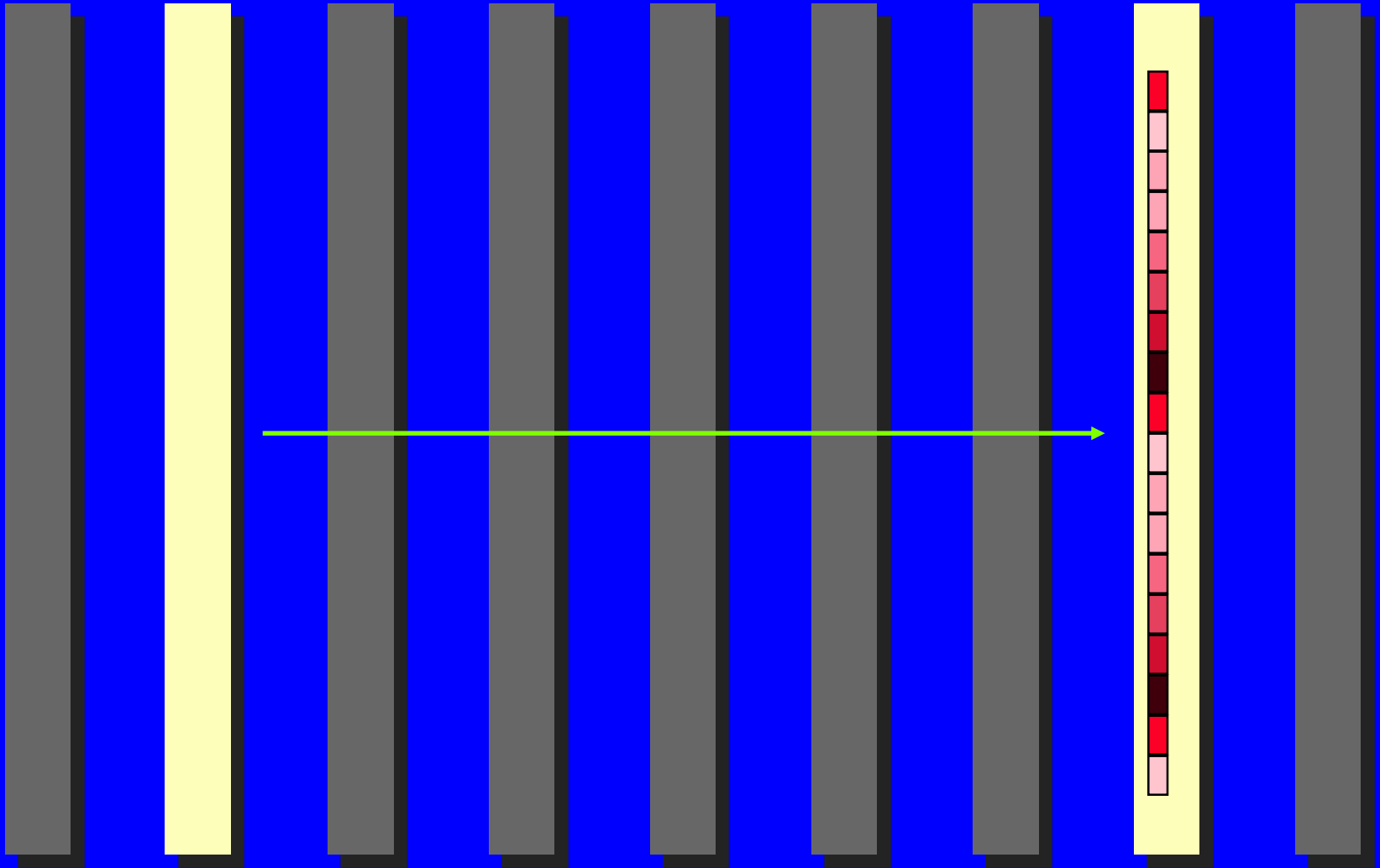


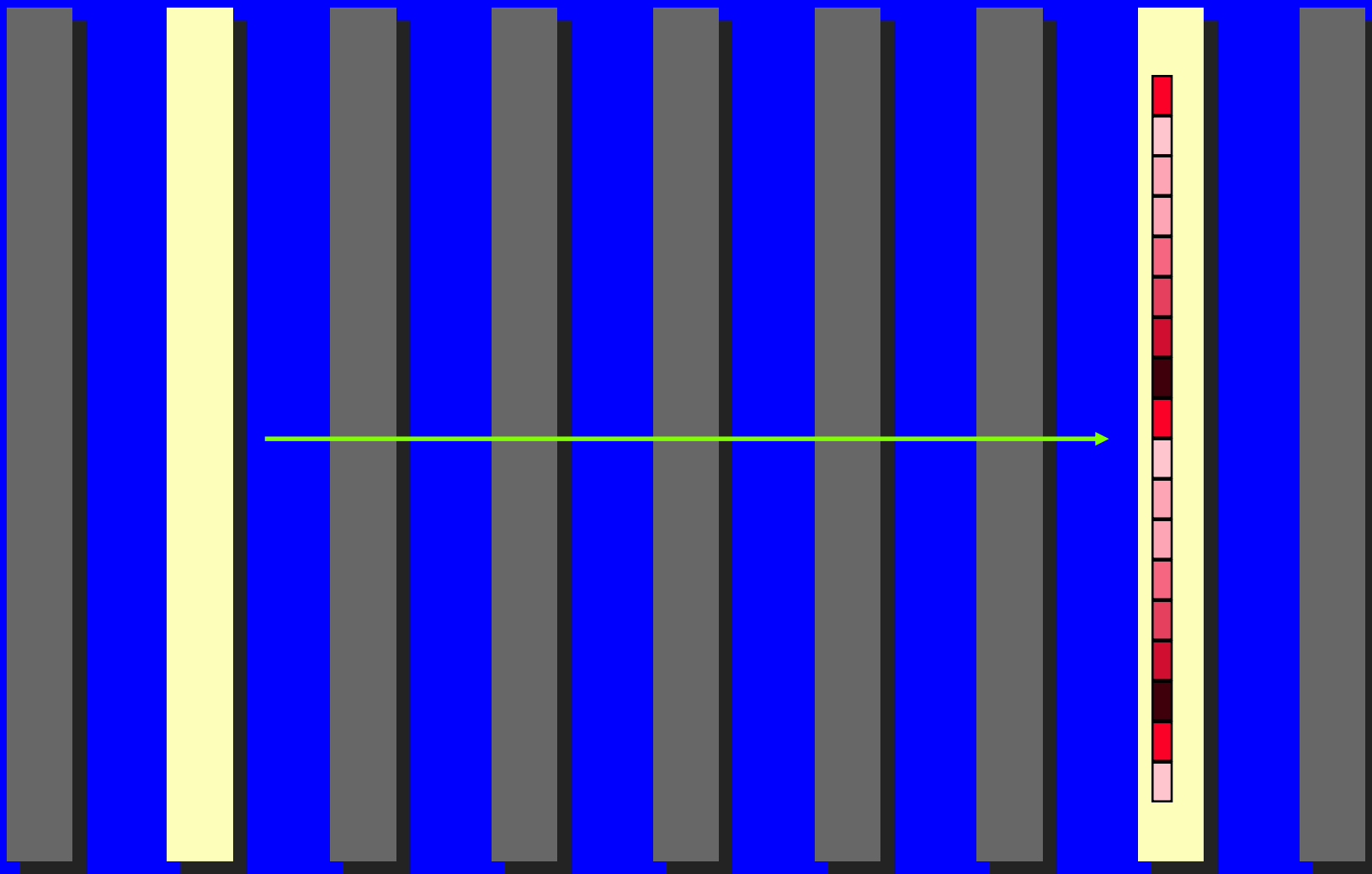












Model of Parallel Computation

- a node can send directly to any other node
- a node can simultaneously receive and send
- cost of communication
 - sending a message of length n between any two nodes
 - if a message encounters a link that simultaneously accomodates M messages, the cost becomes

Model of Parallel Computation

- a node can send directly to any other node
- a node can simultaneously receive and send
- cost of communication
 - sending a message of length n between any two nodes

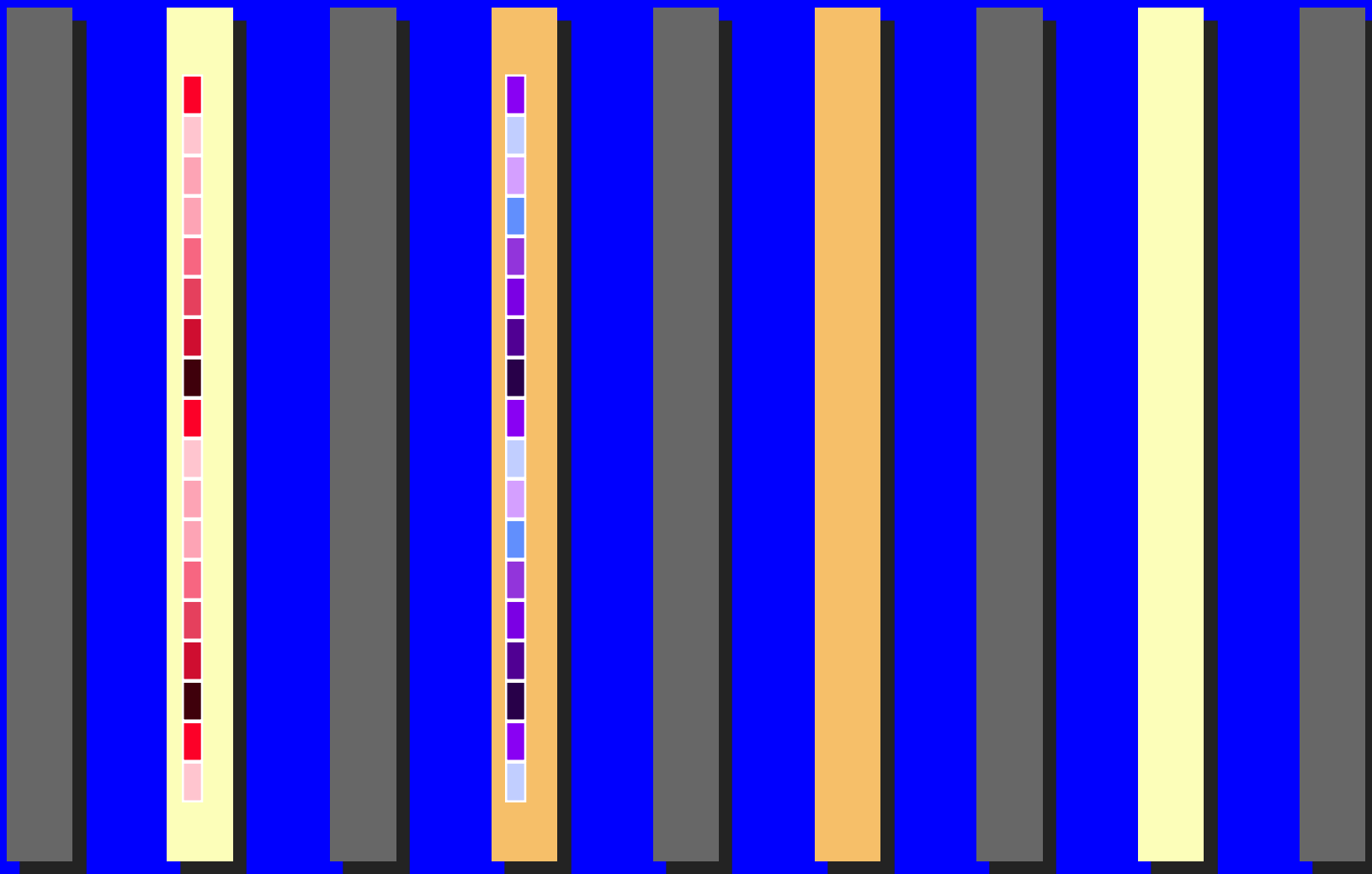
$$\alpha + n\beta$$

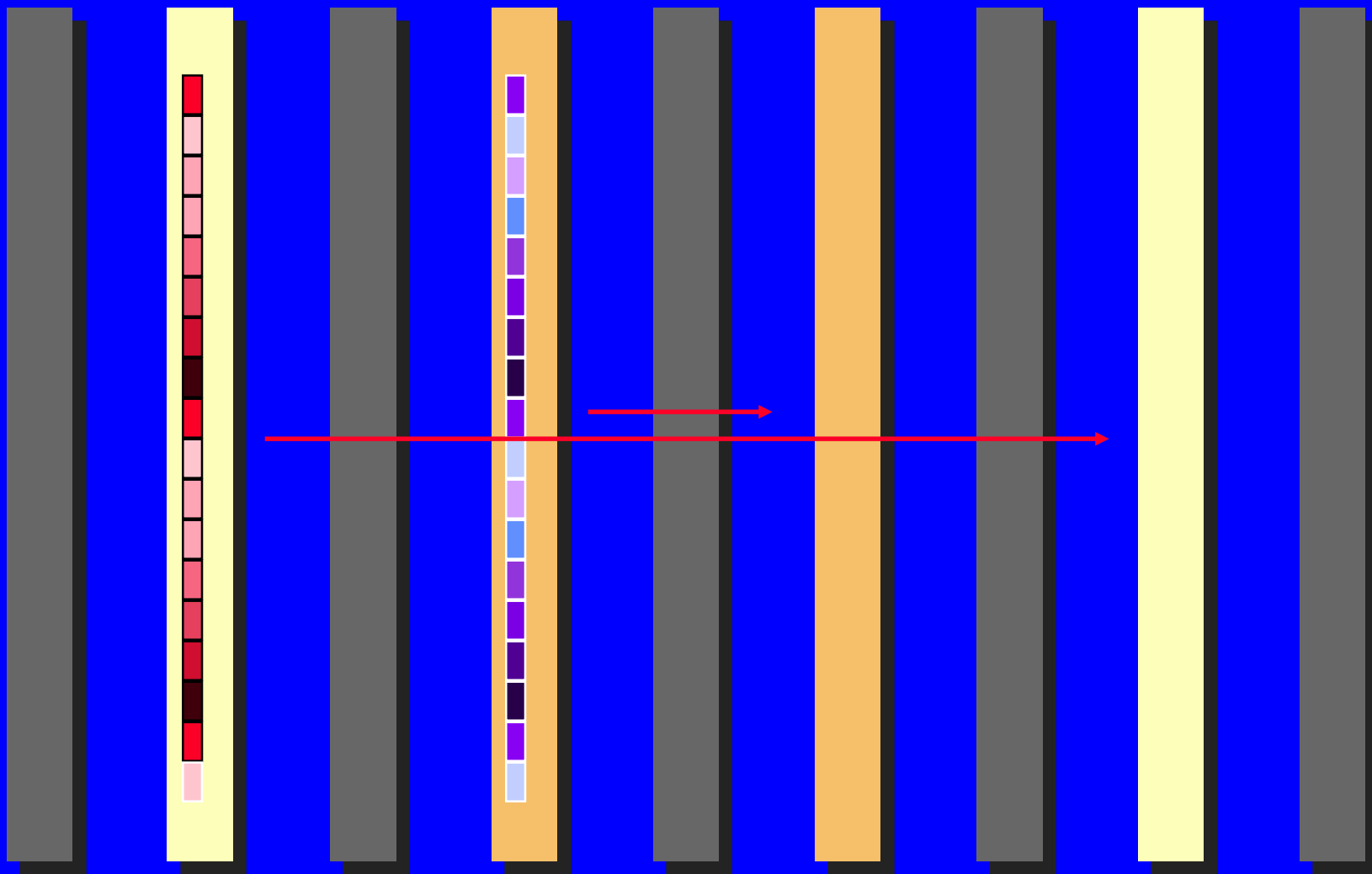
- if a message encounters a link that simultaneously accomodates M messages, the cost becomes

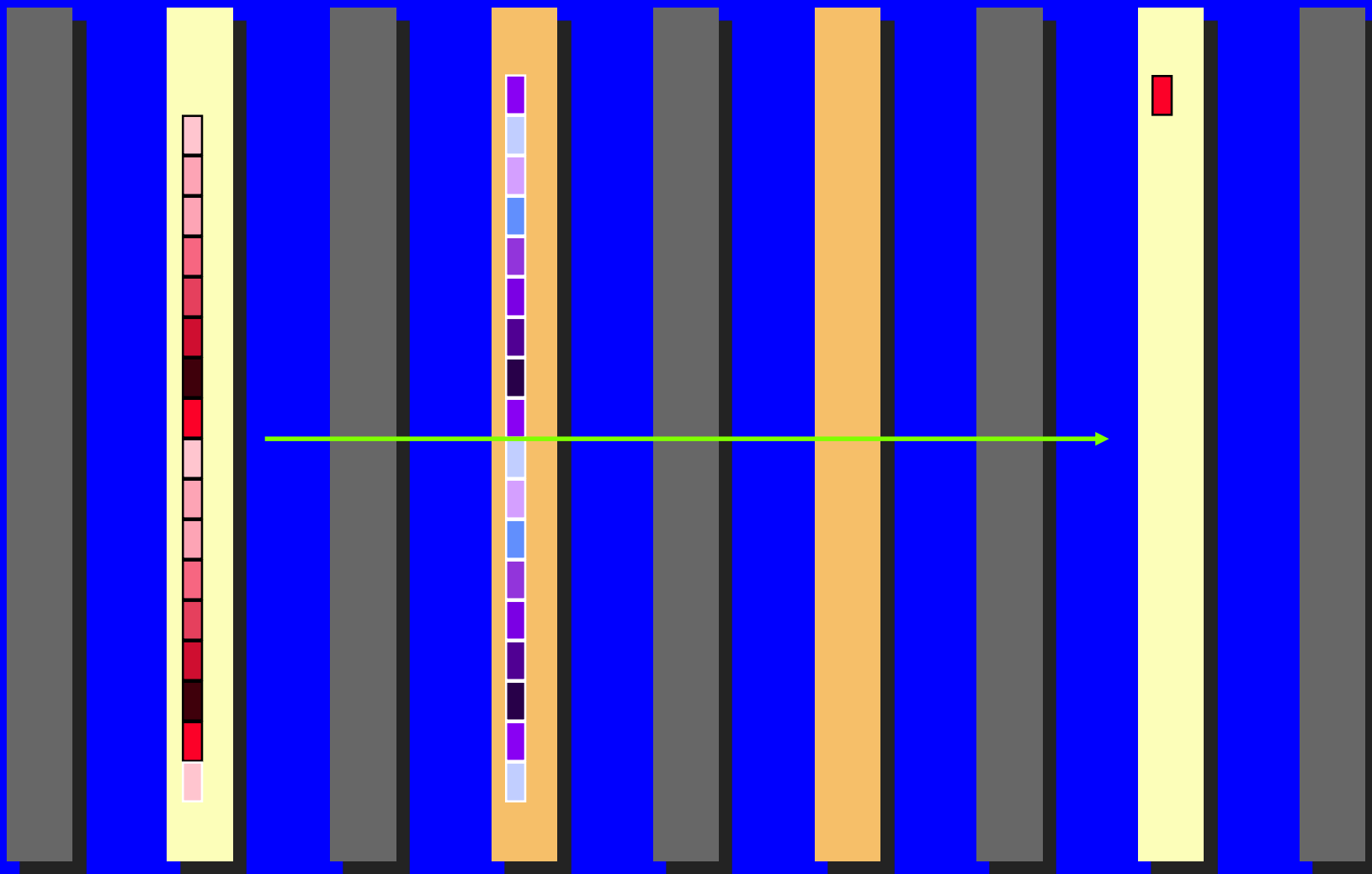
$$\alpha + Mn\beta$$

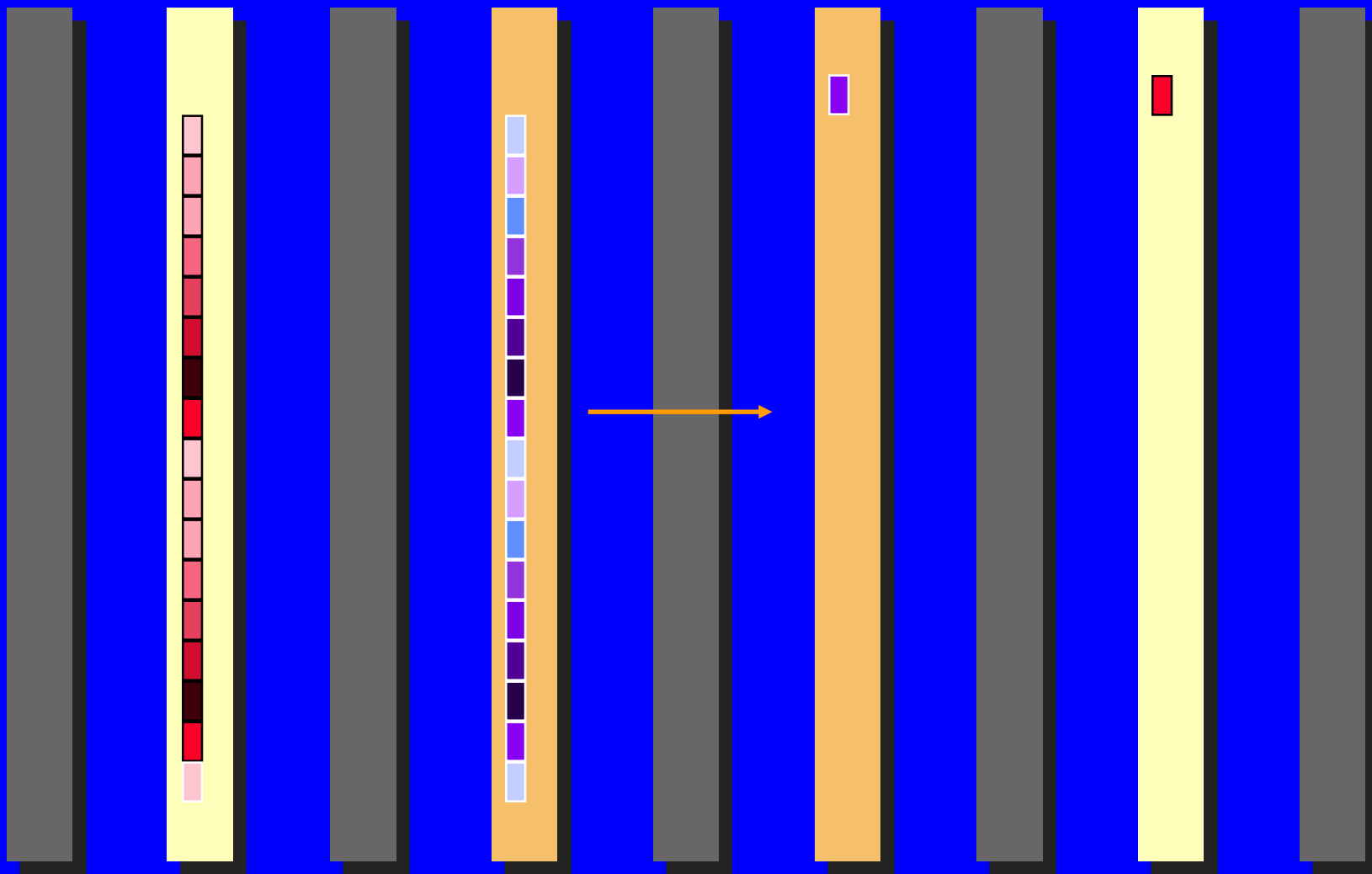
Interfering messages

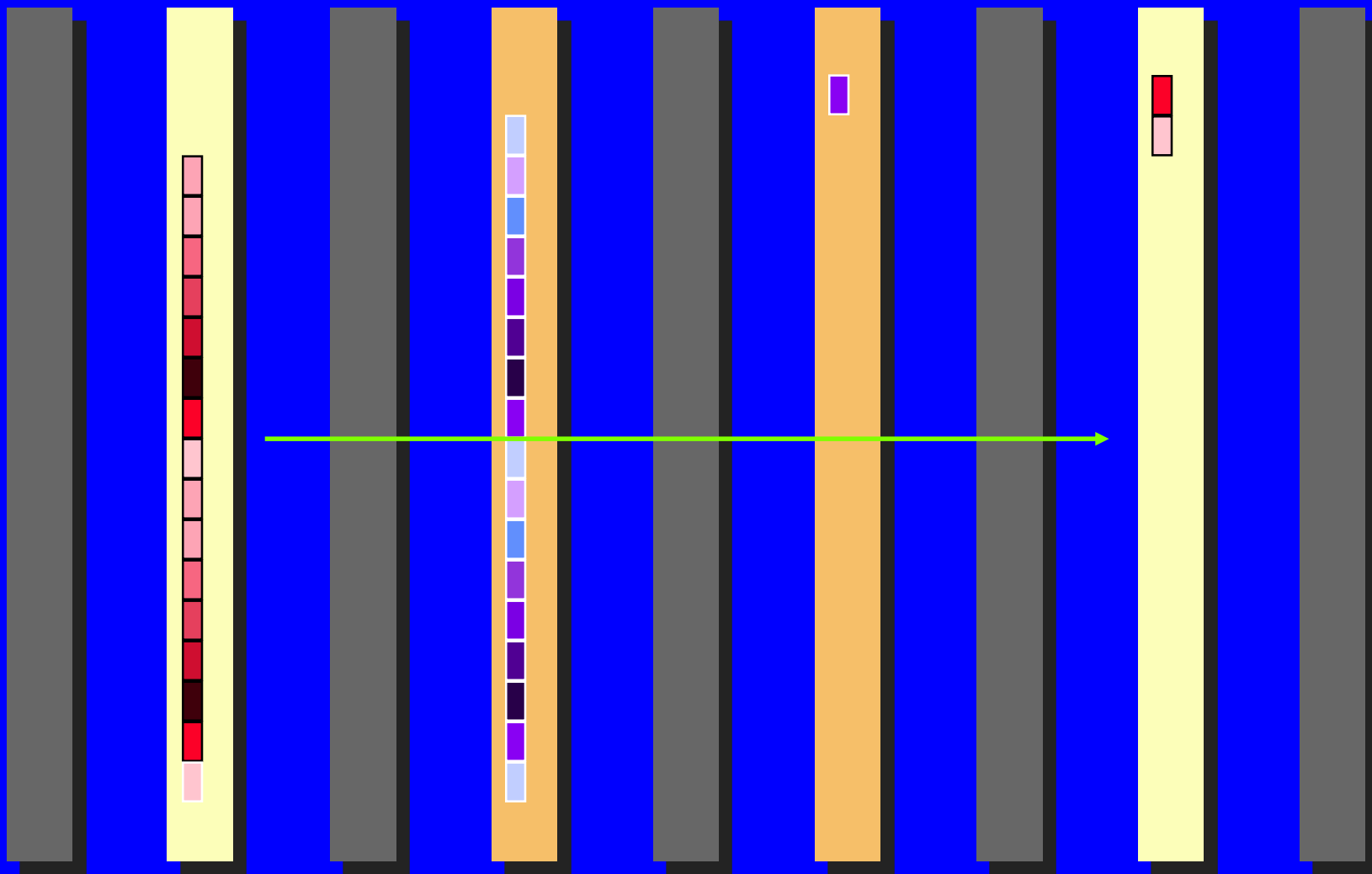
- Example: two messages of length n which share at least one link

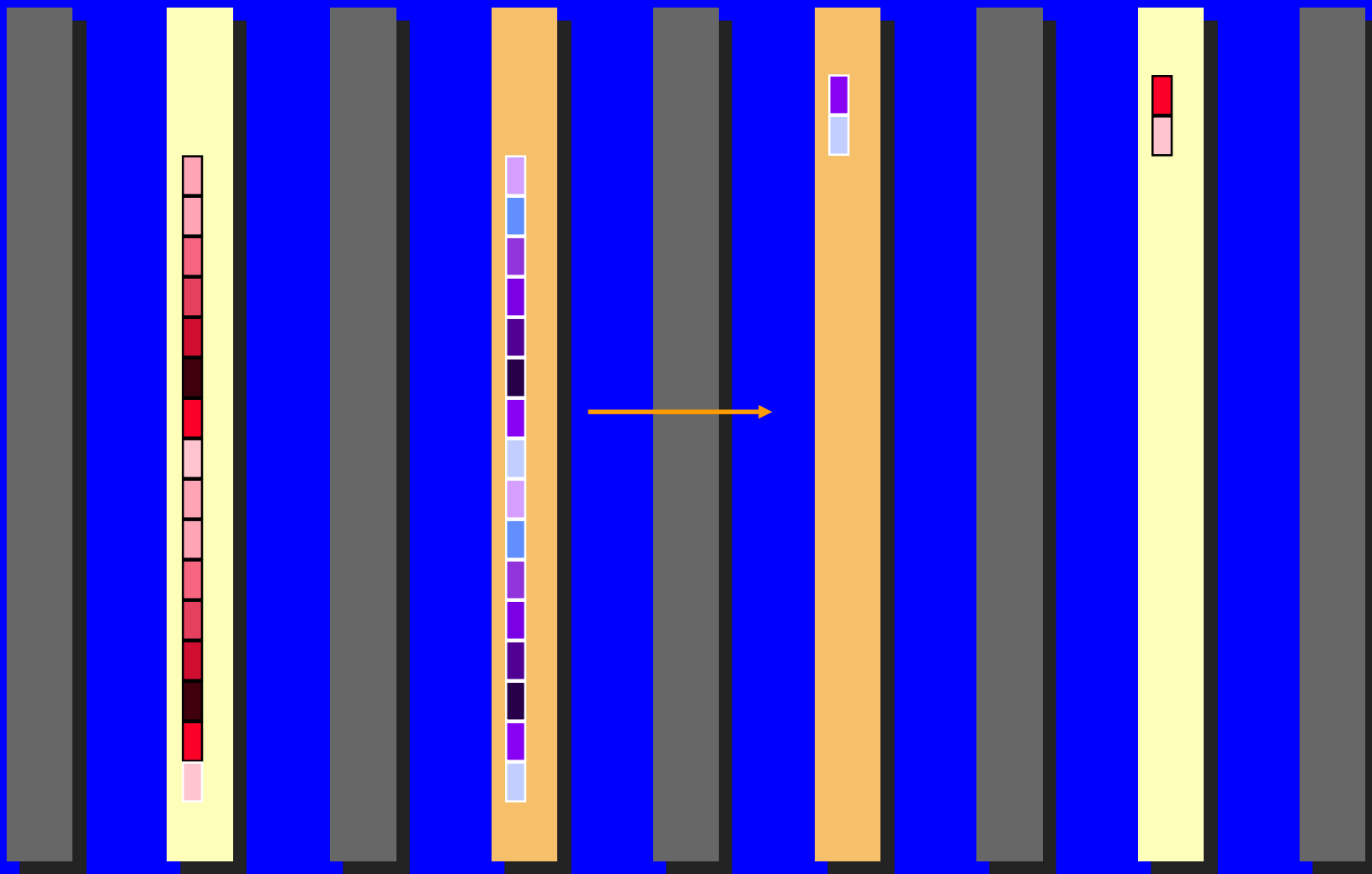




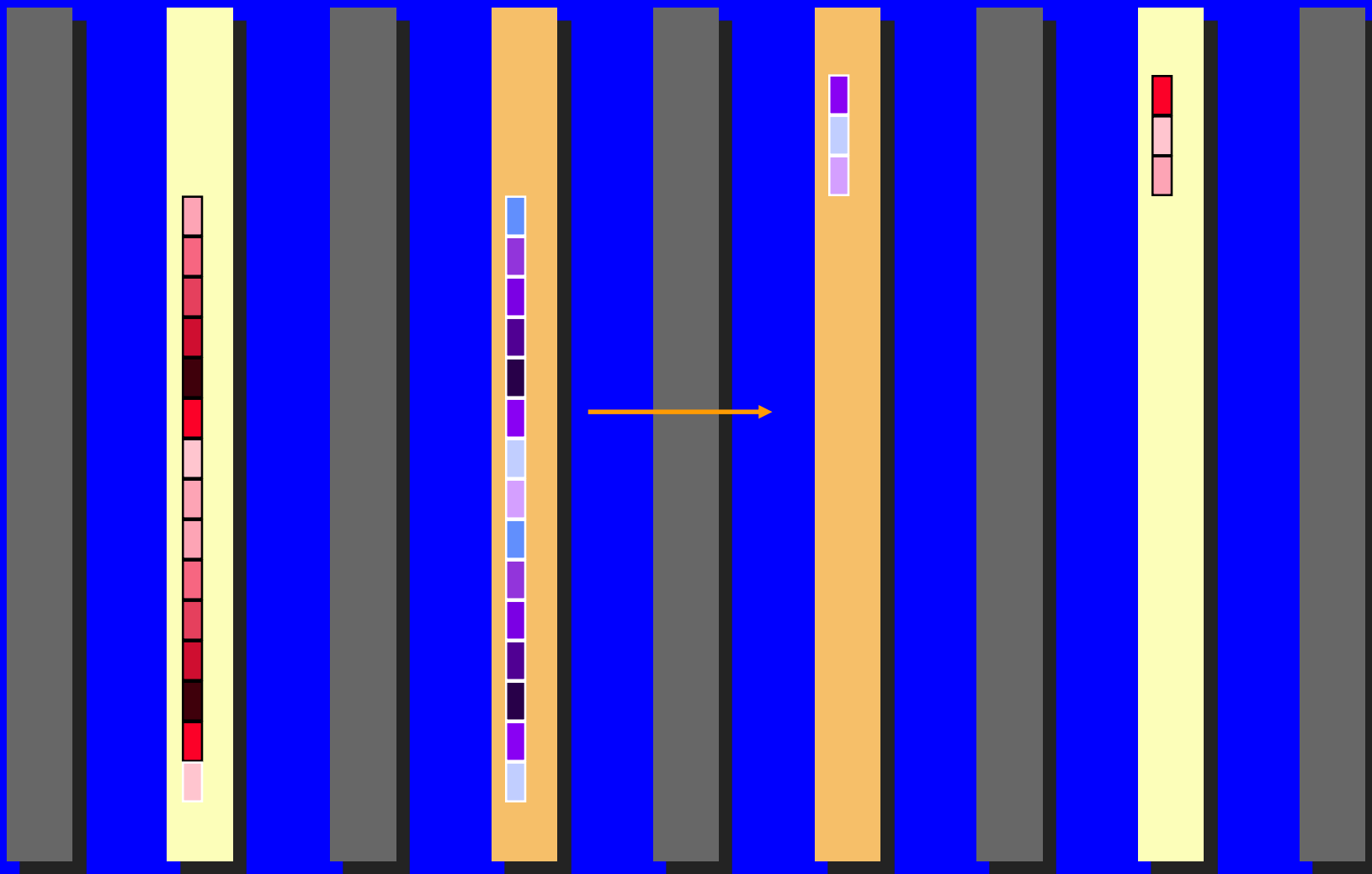








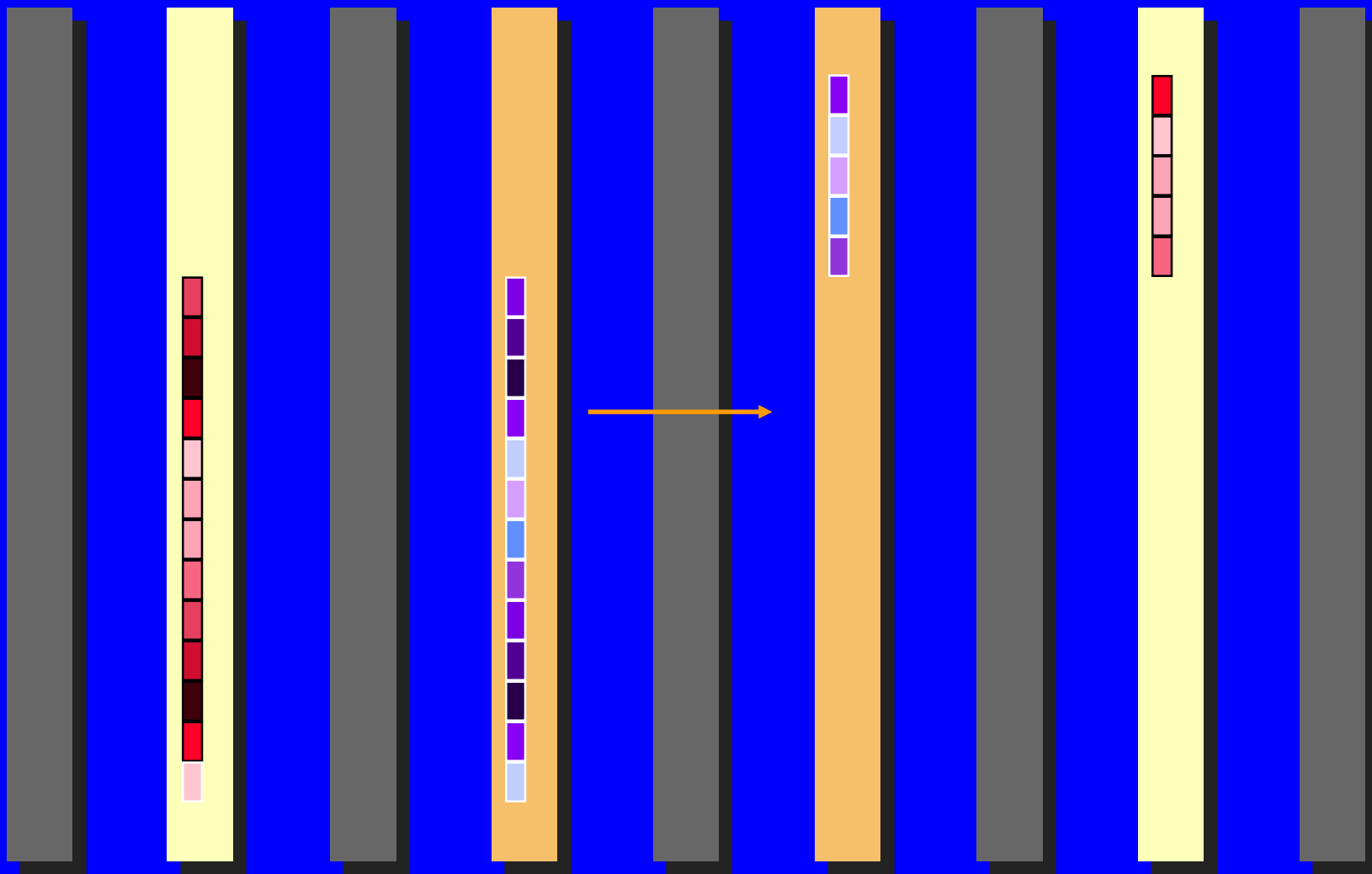


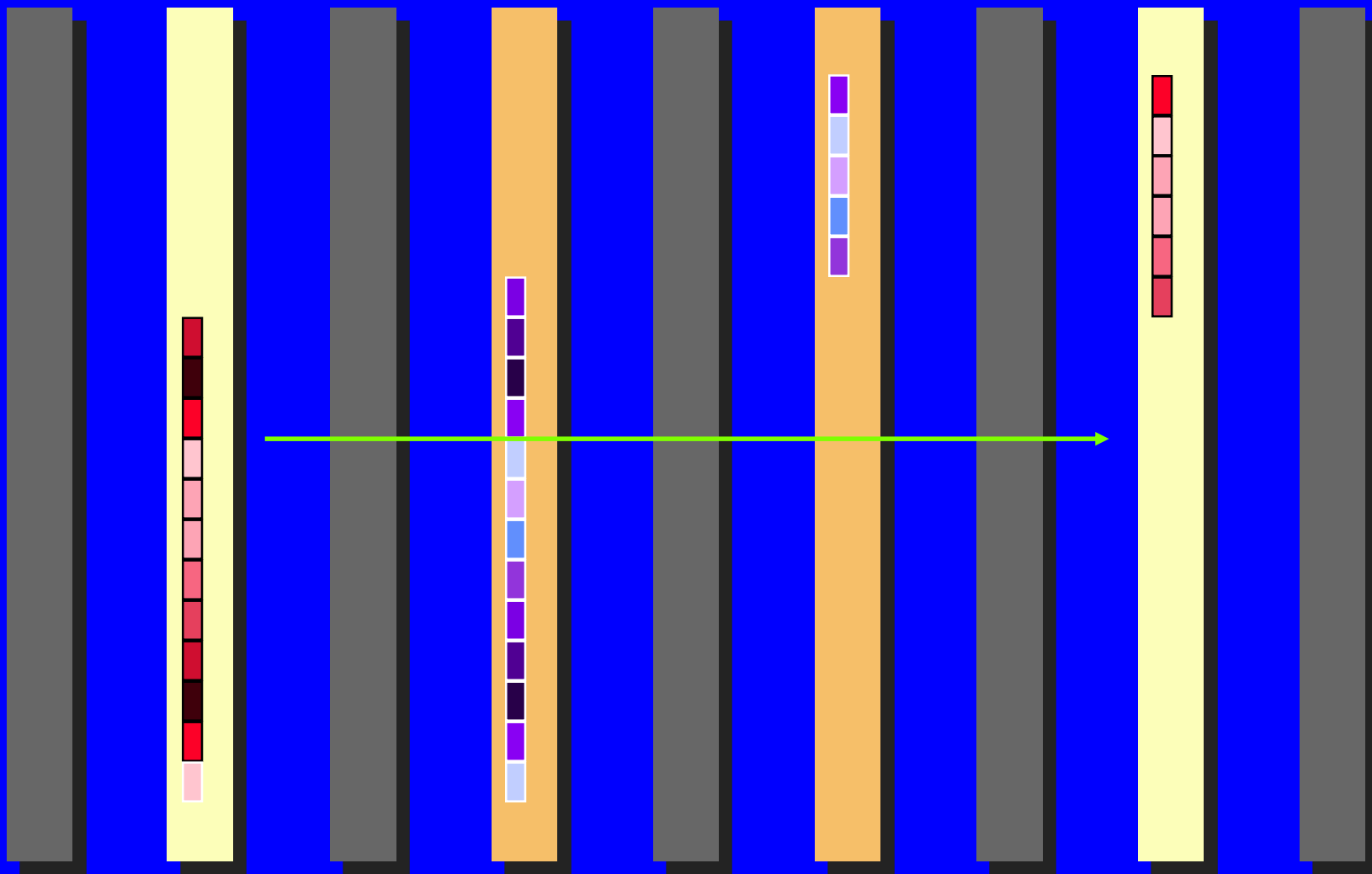


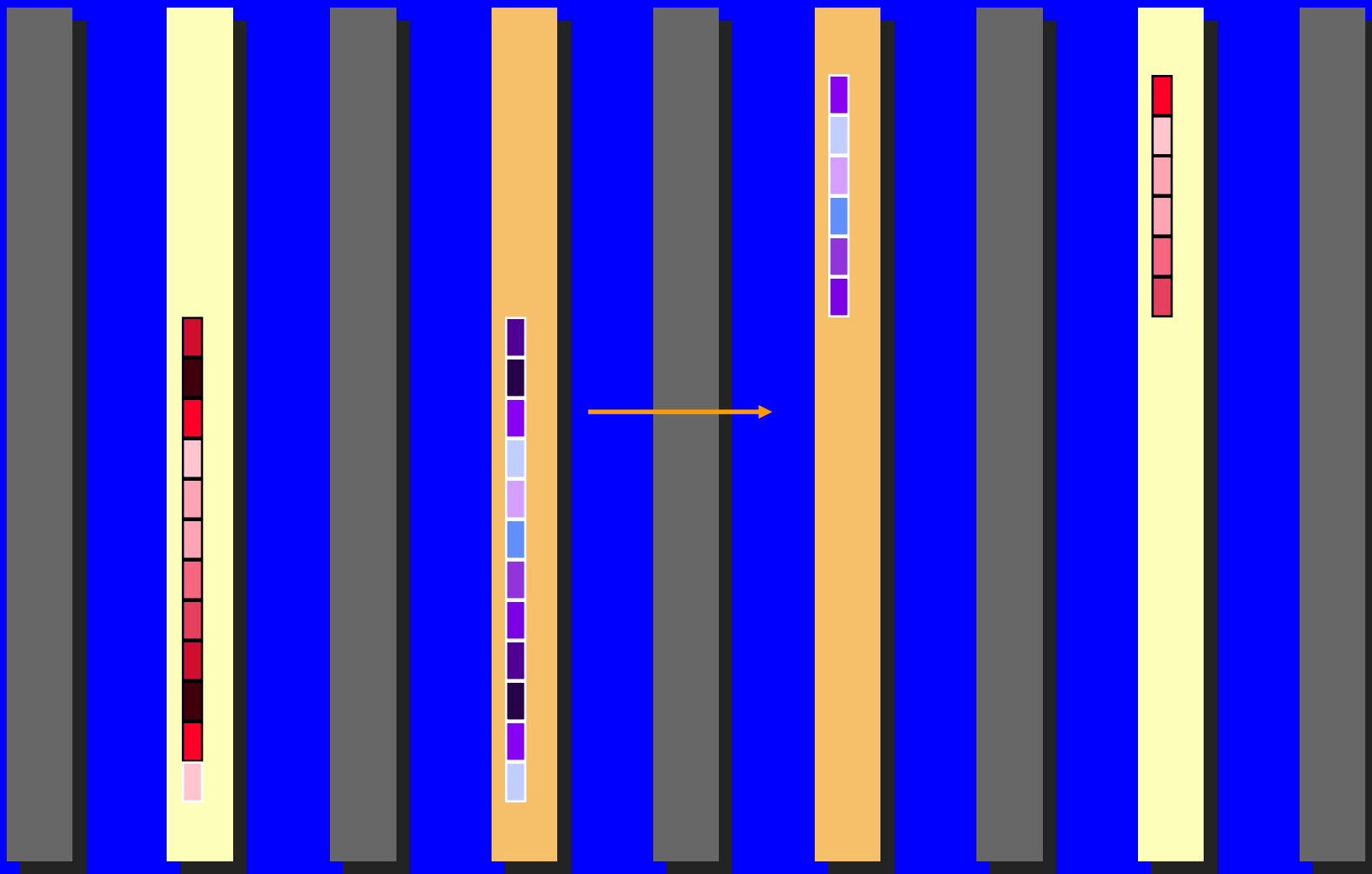


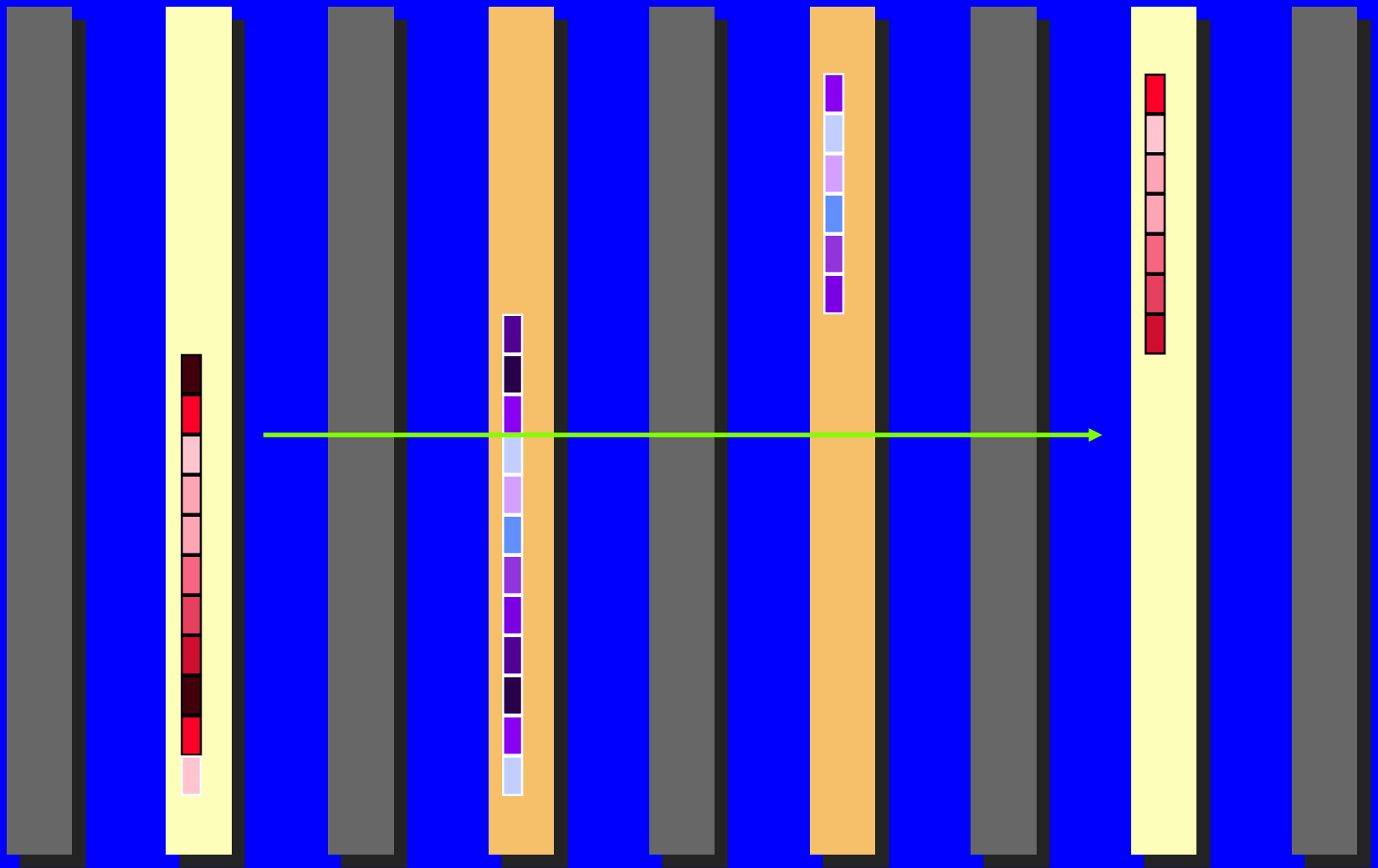


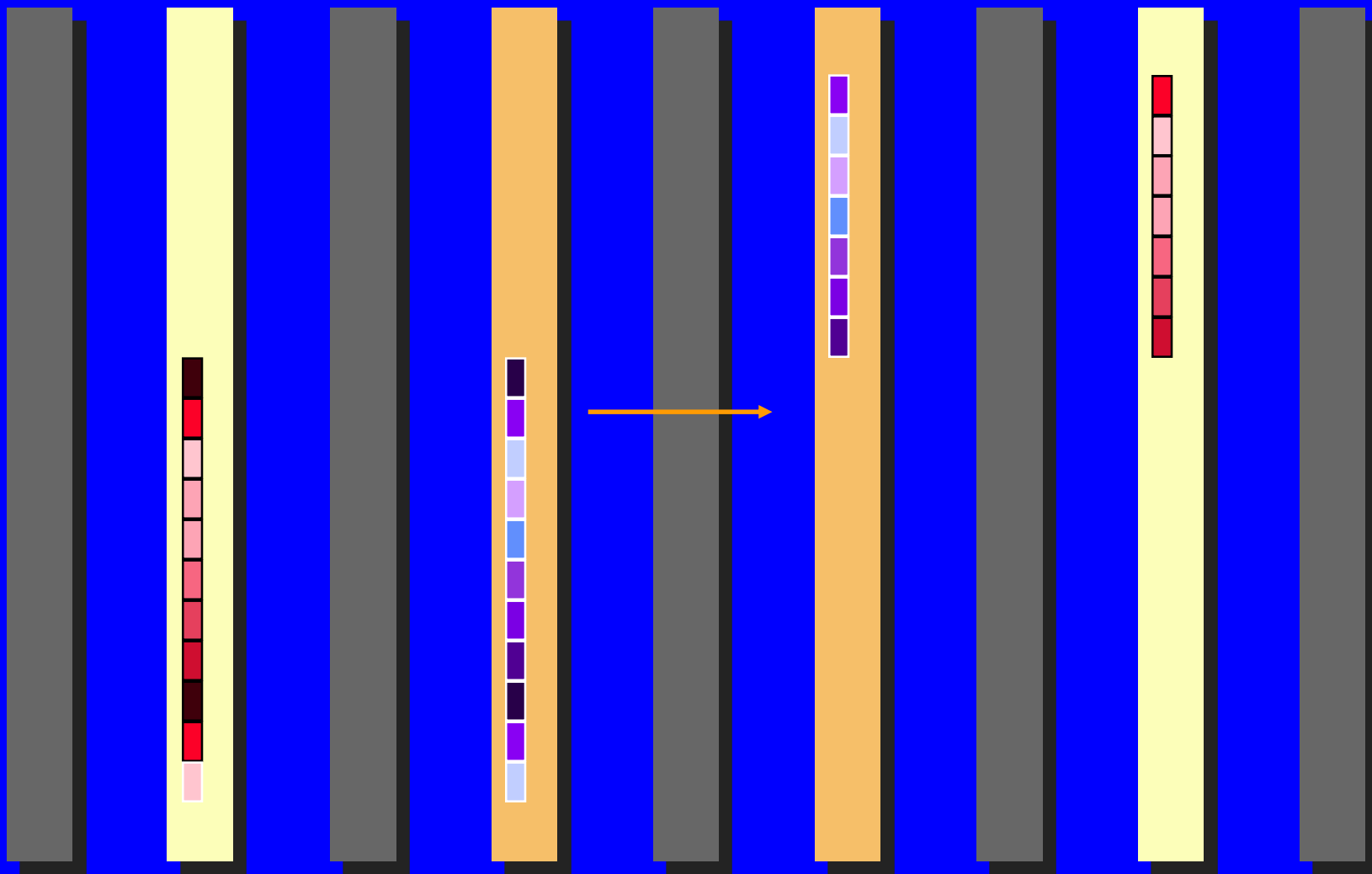


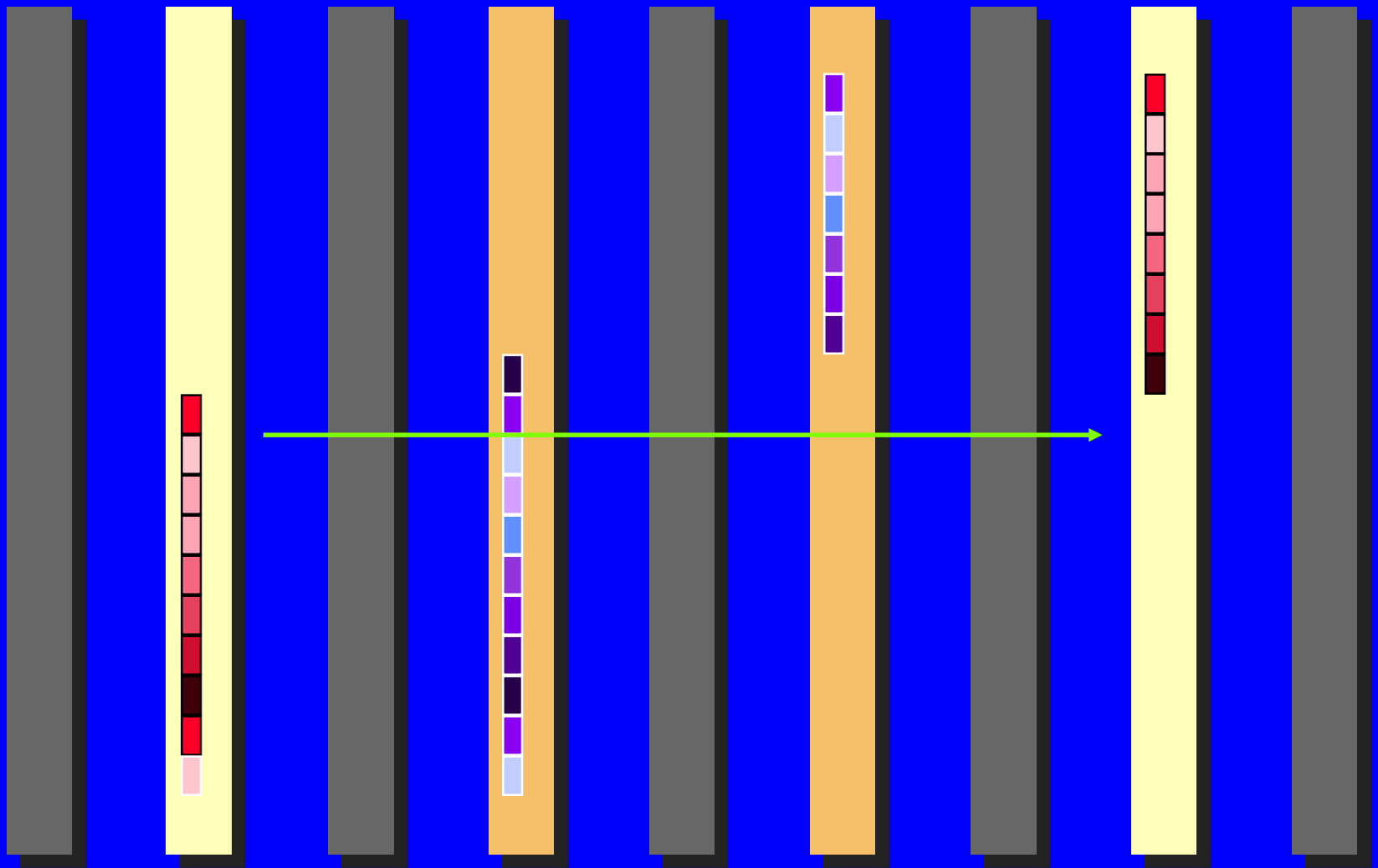


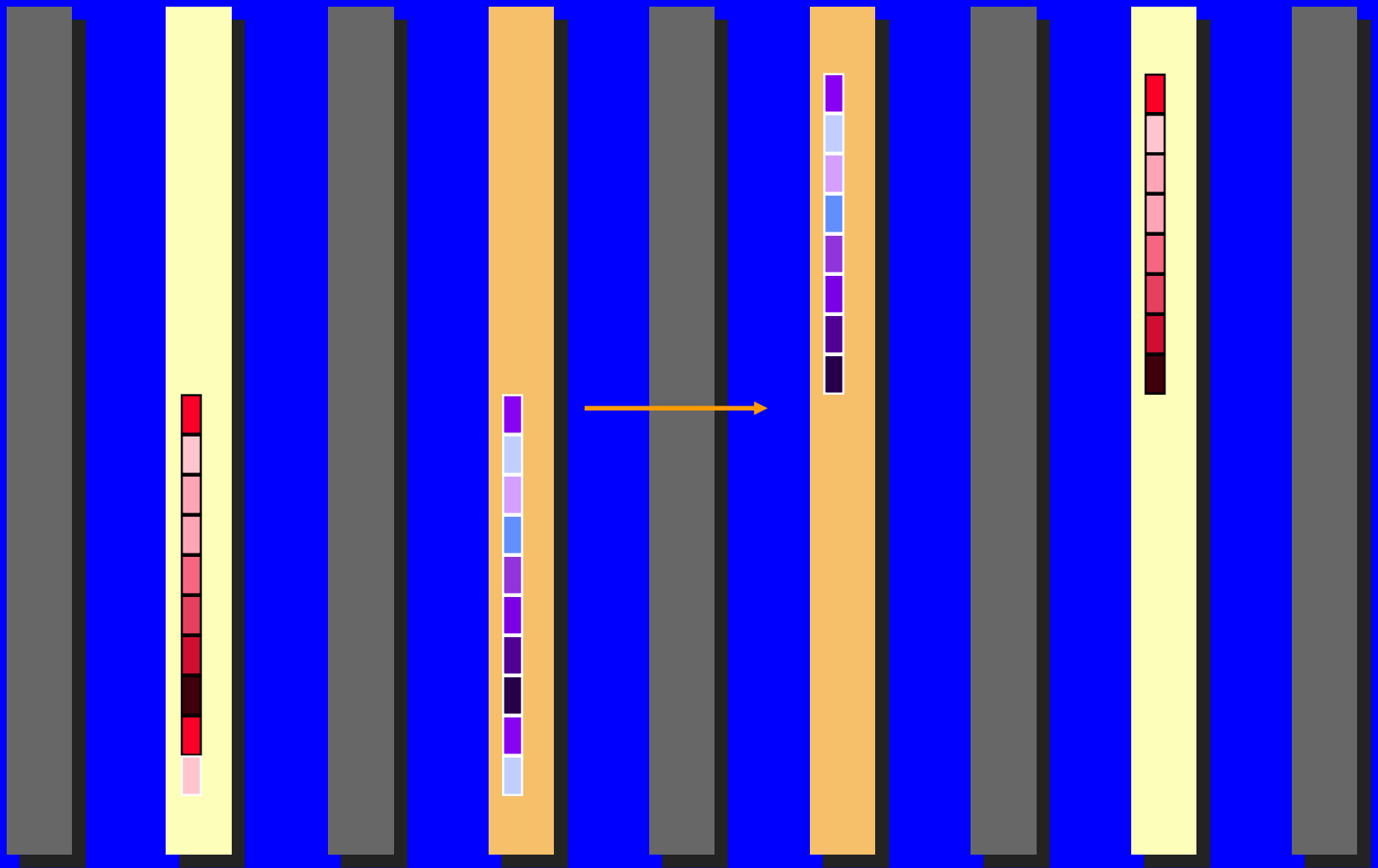


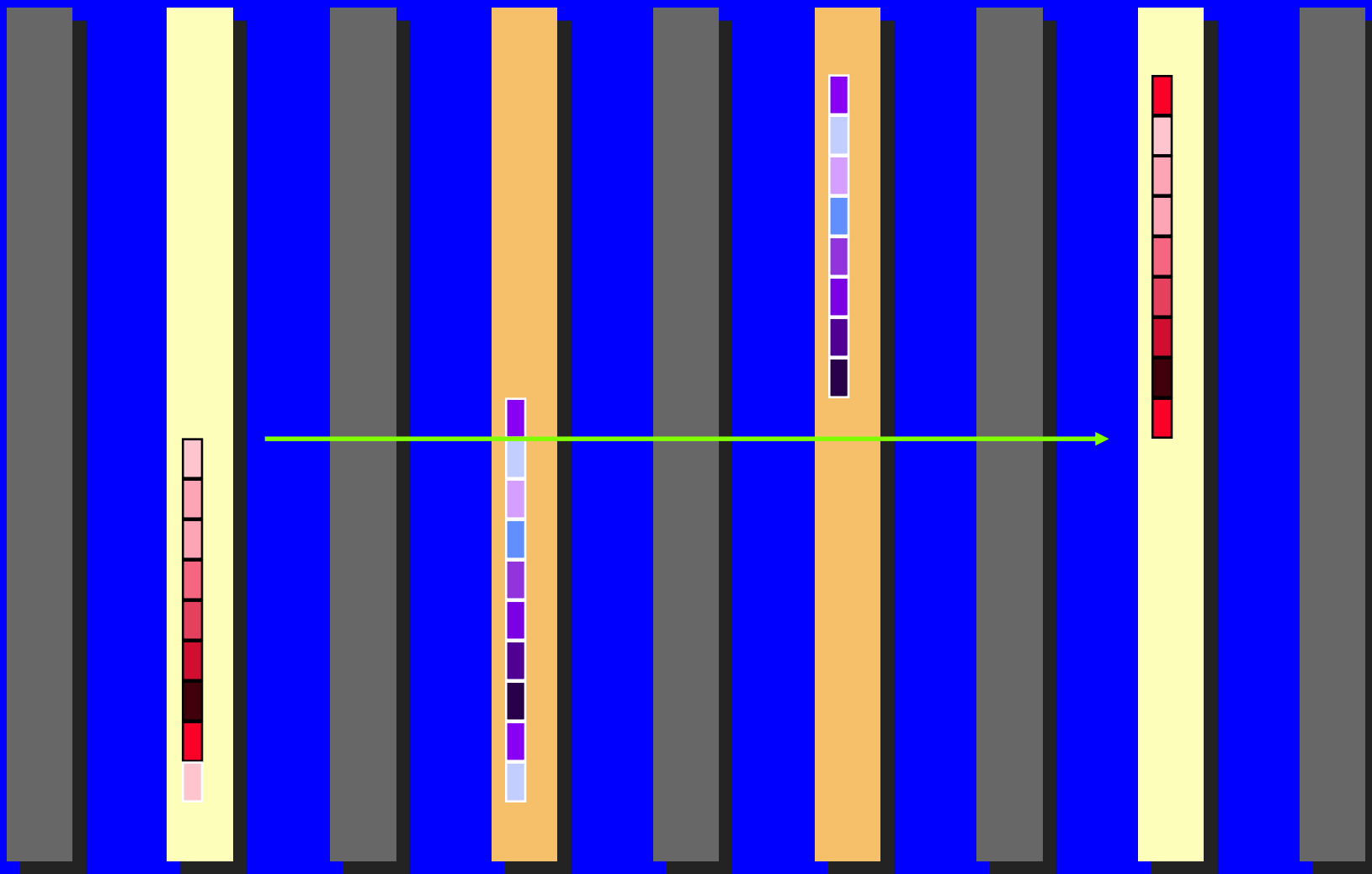


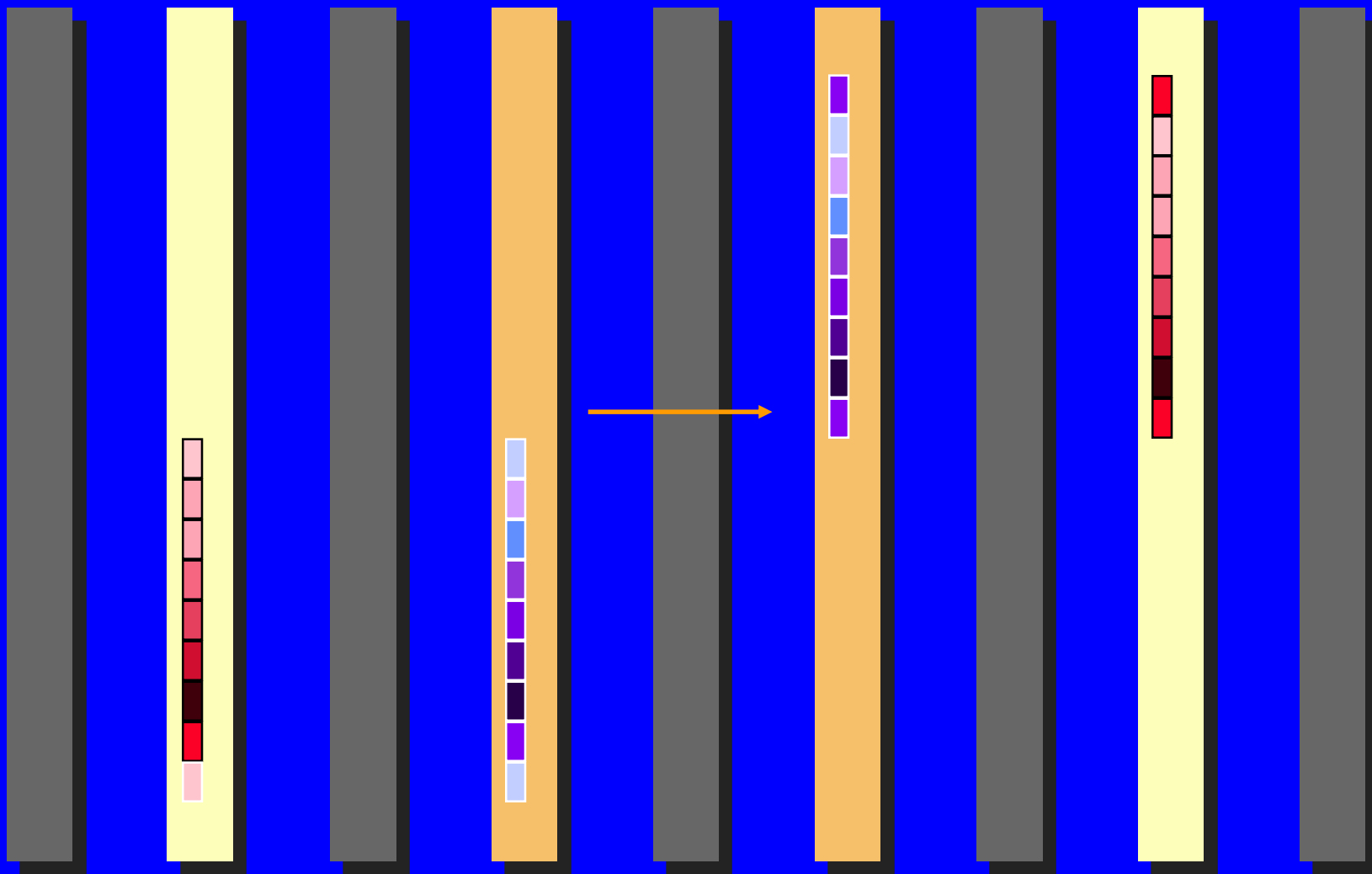


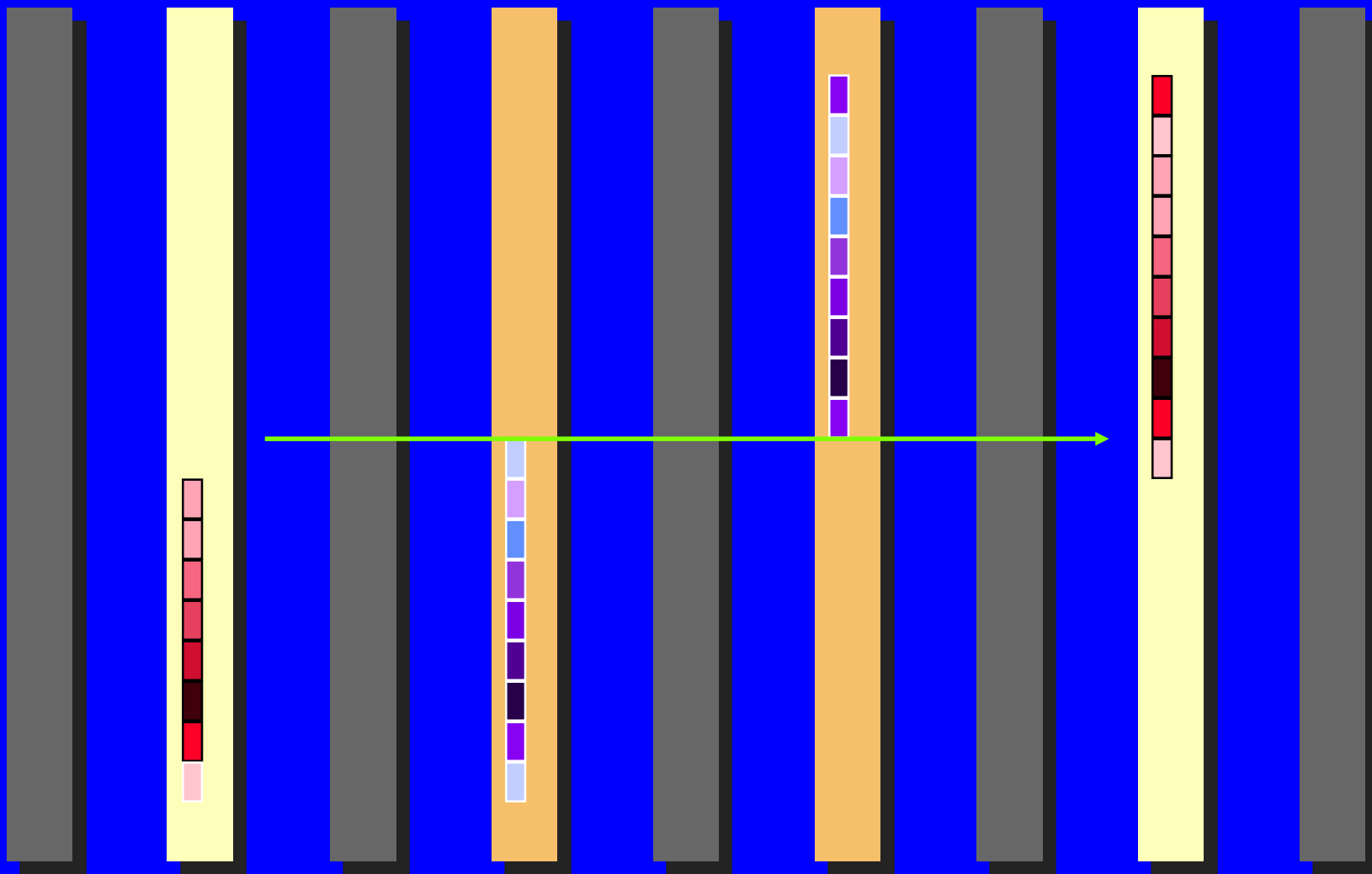


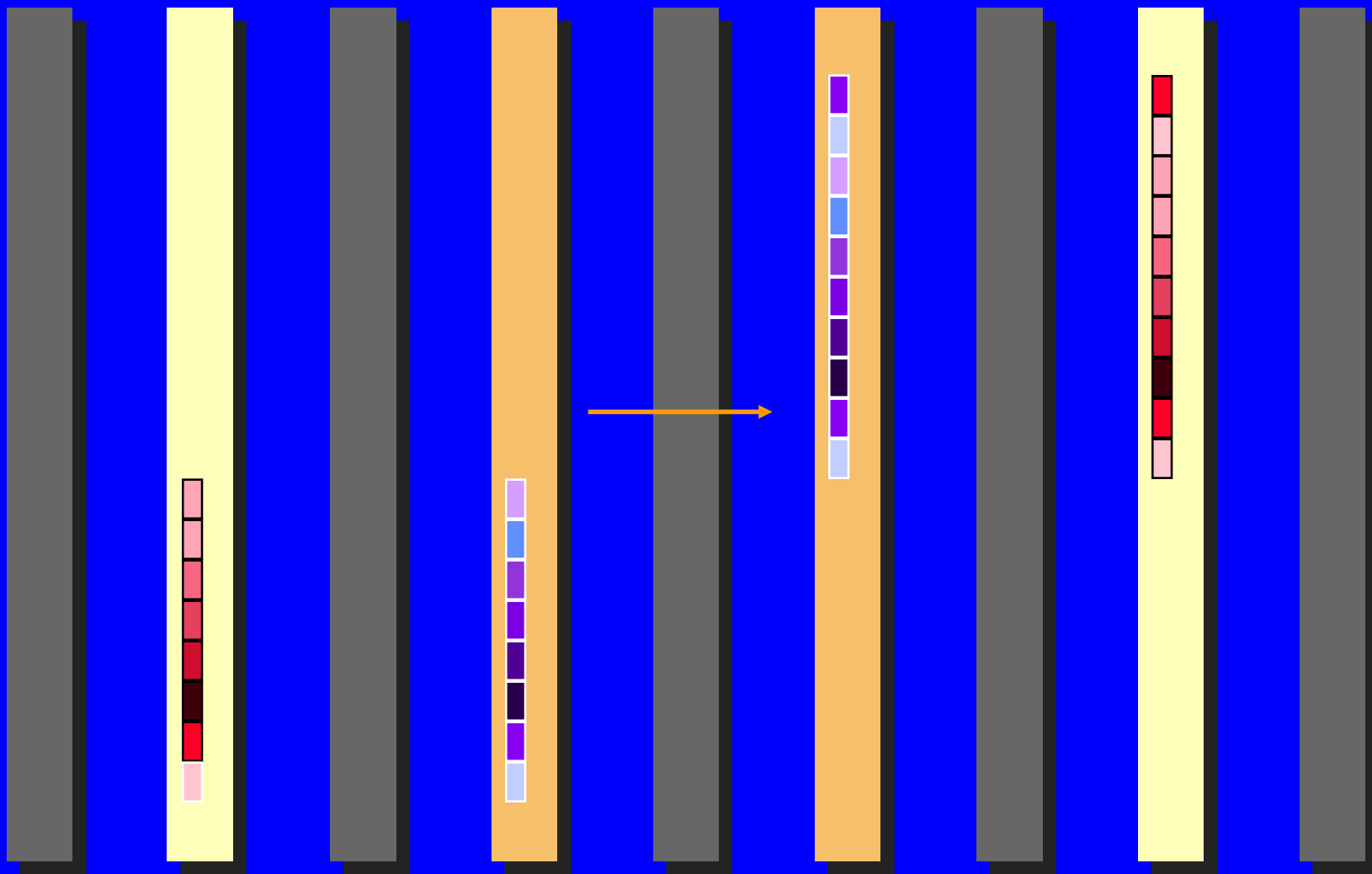




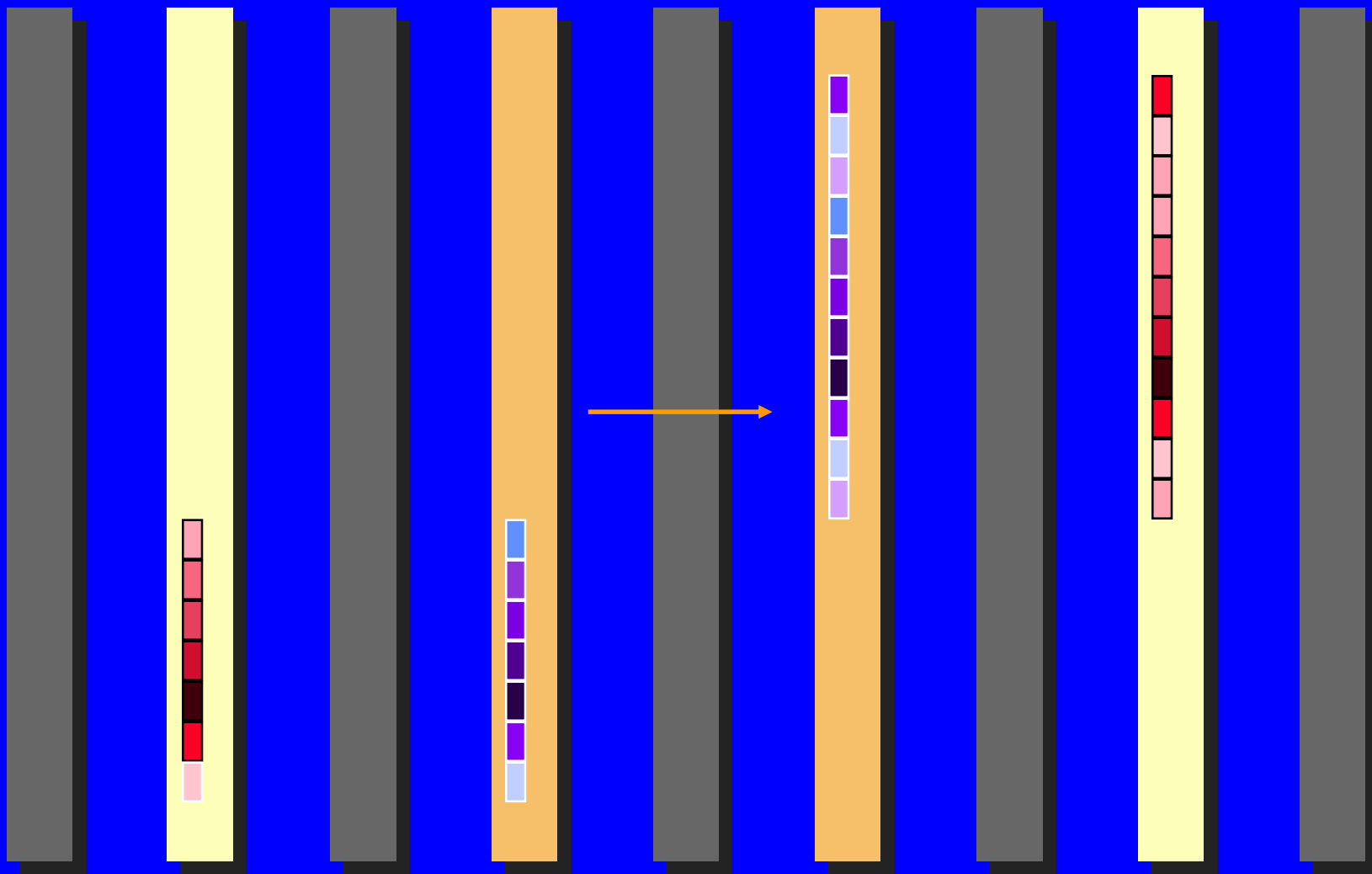




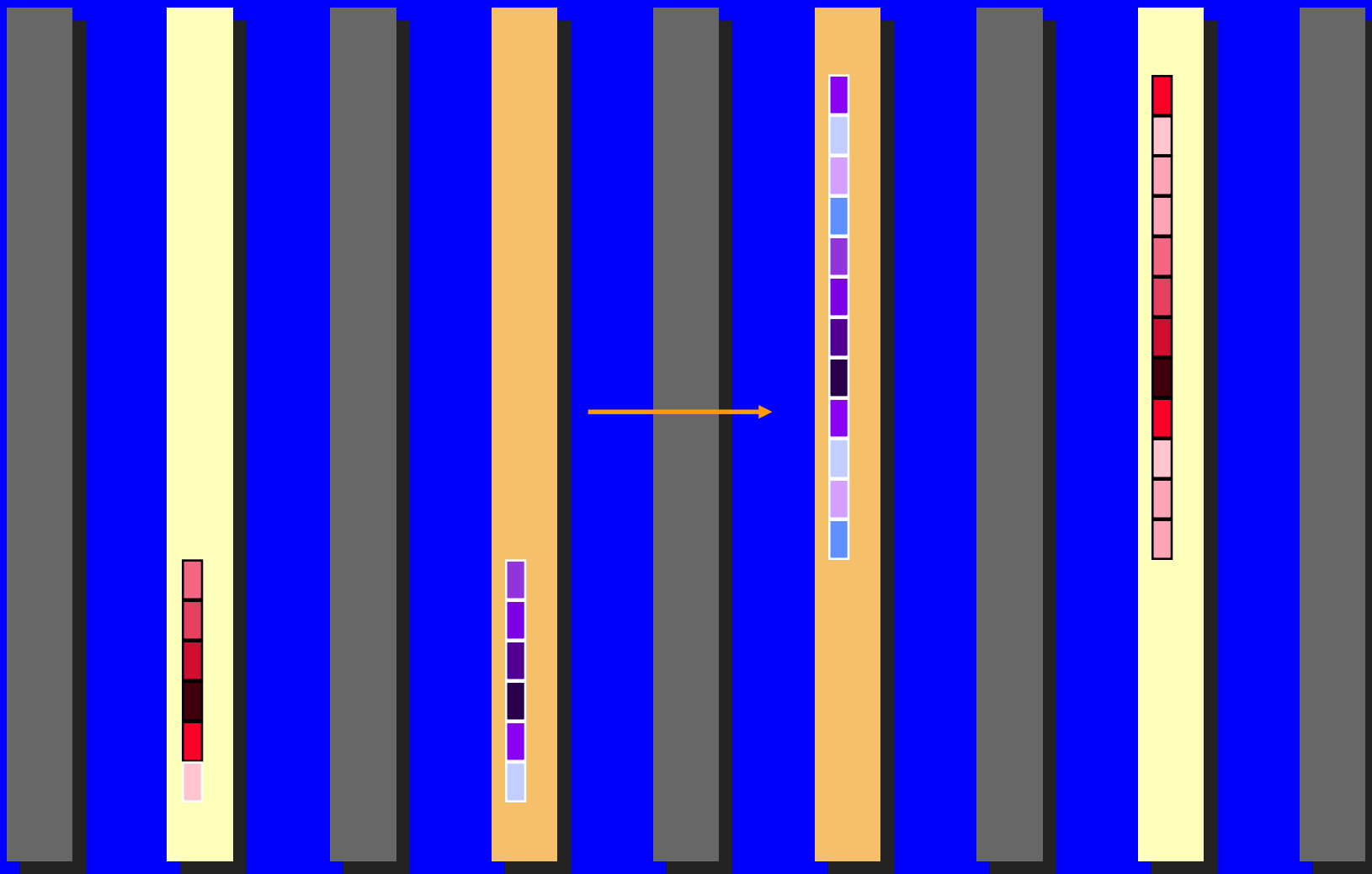




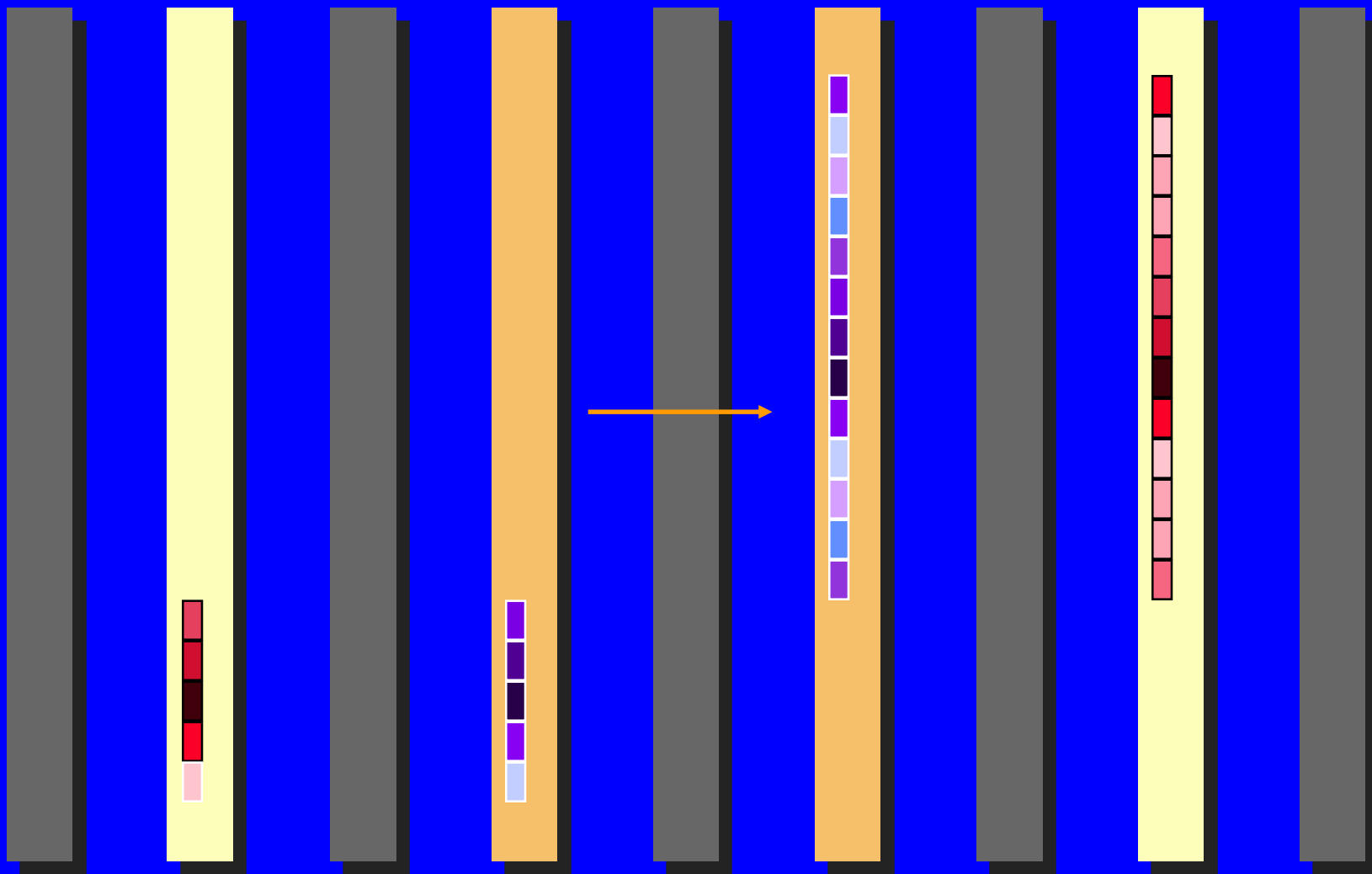




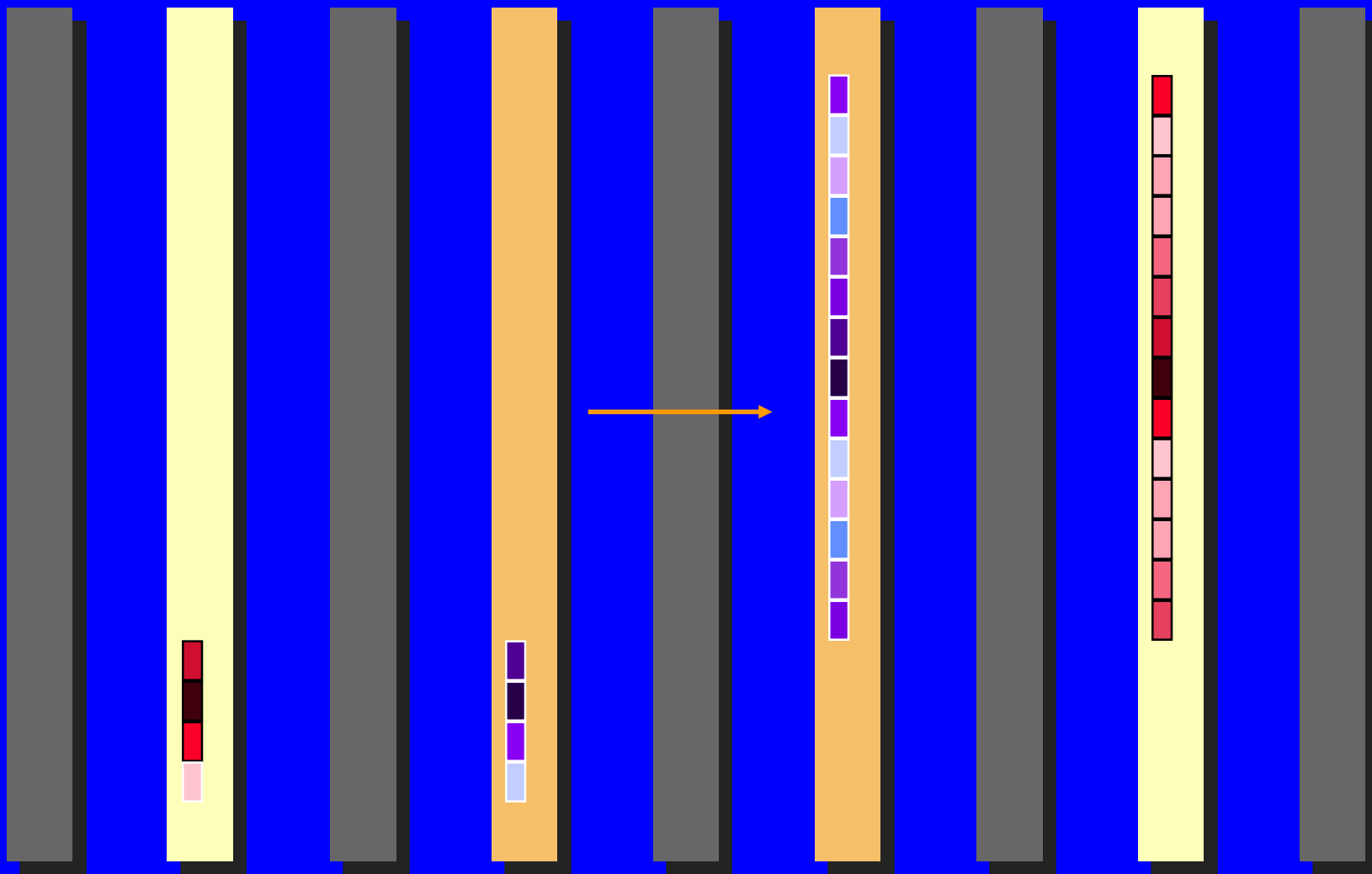


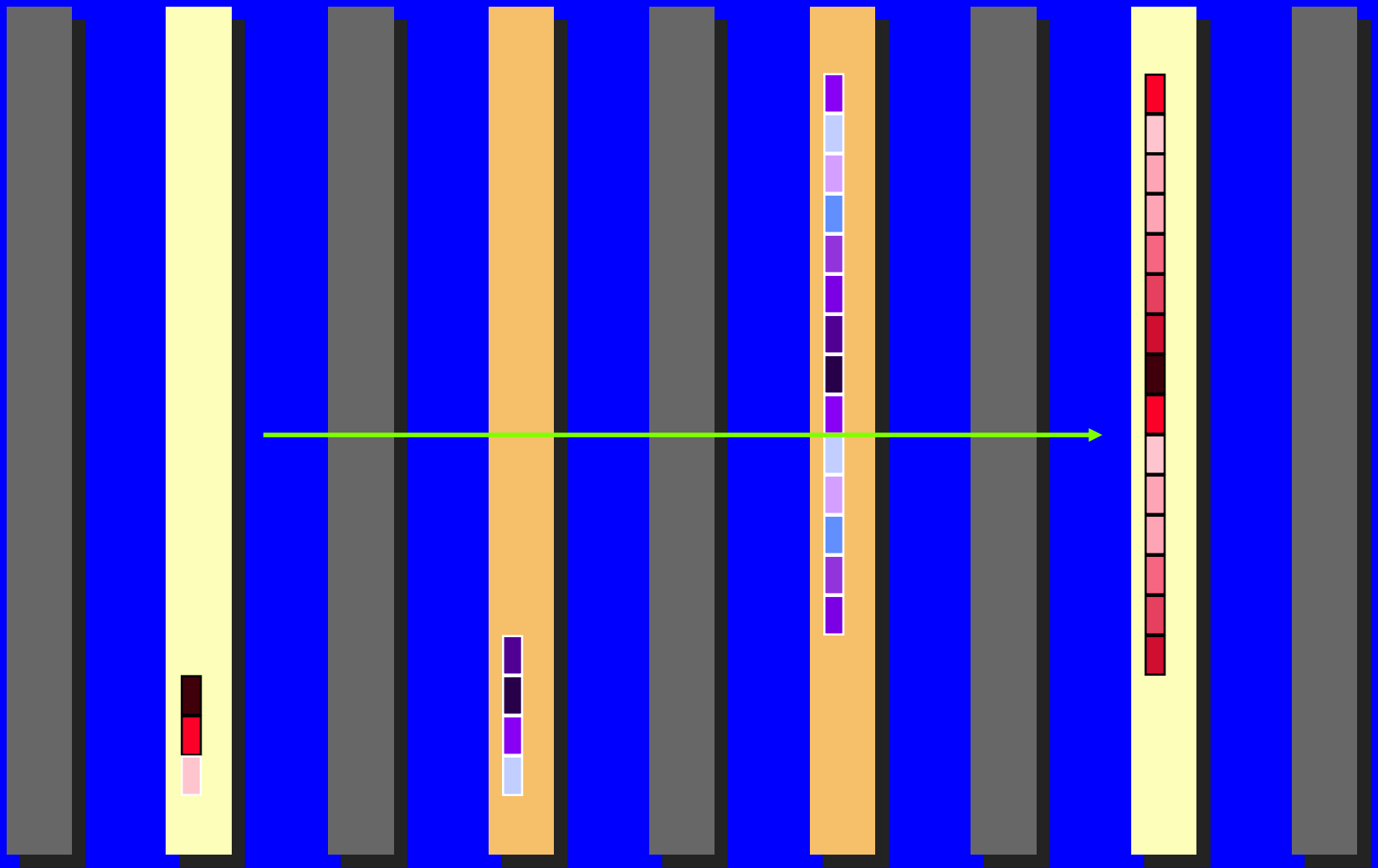


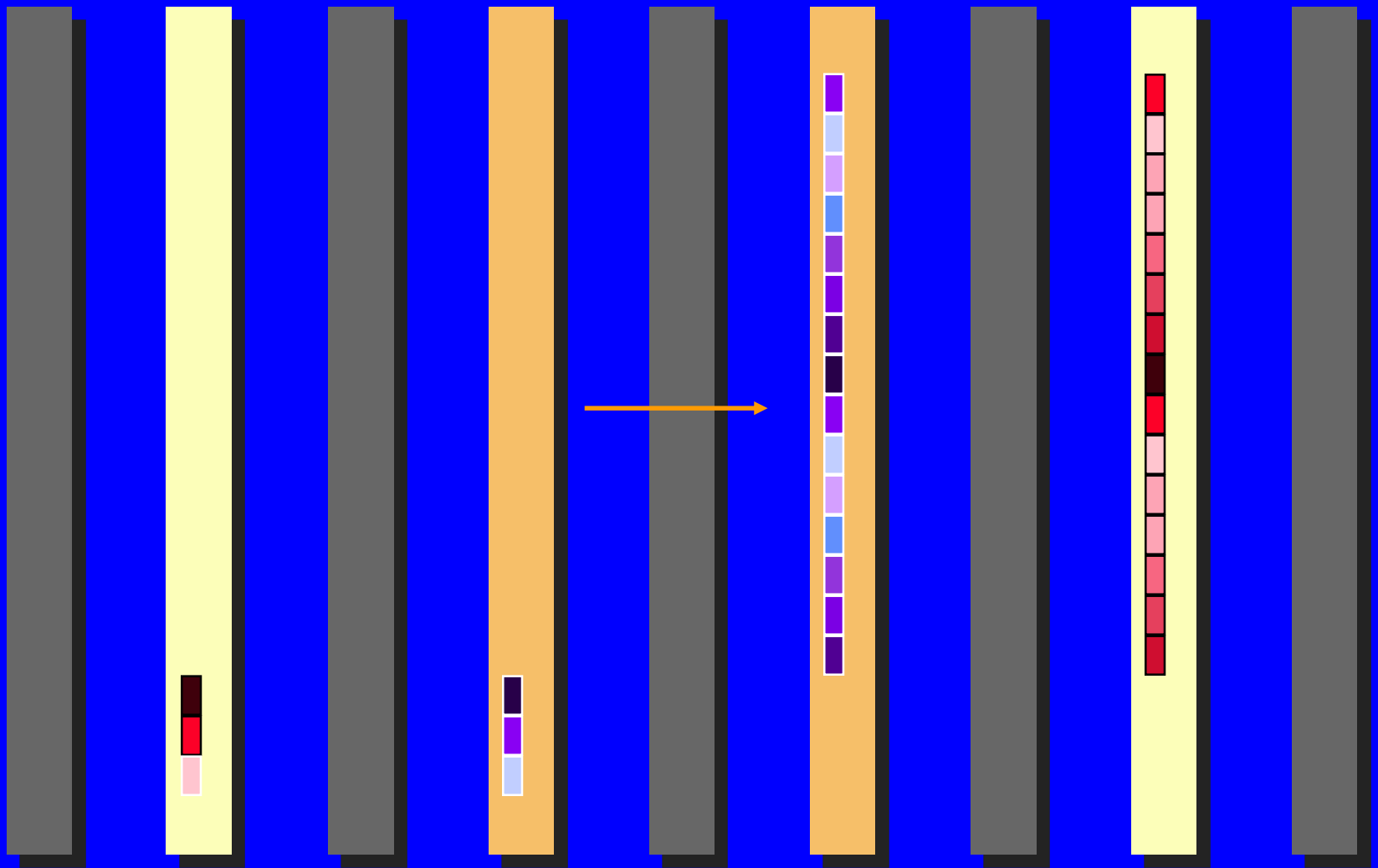








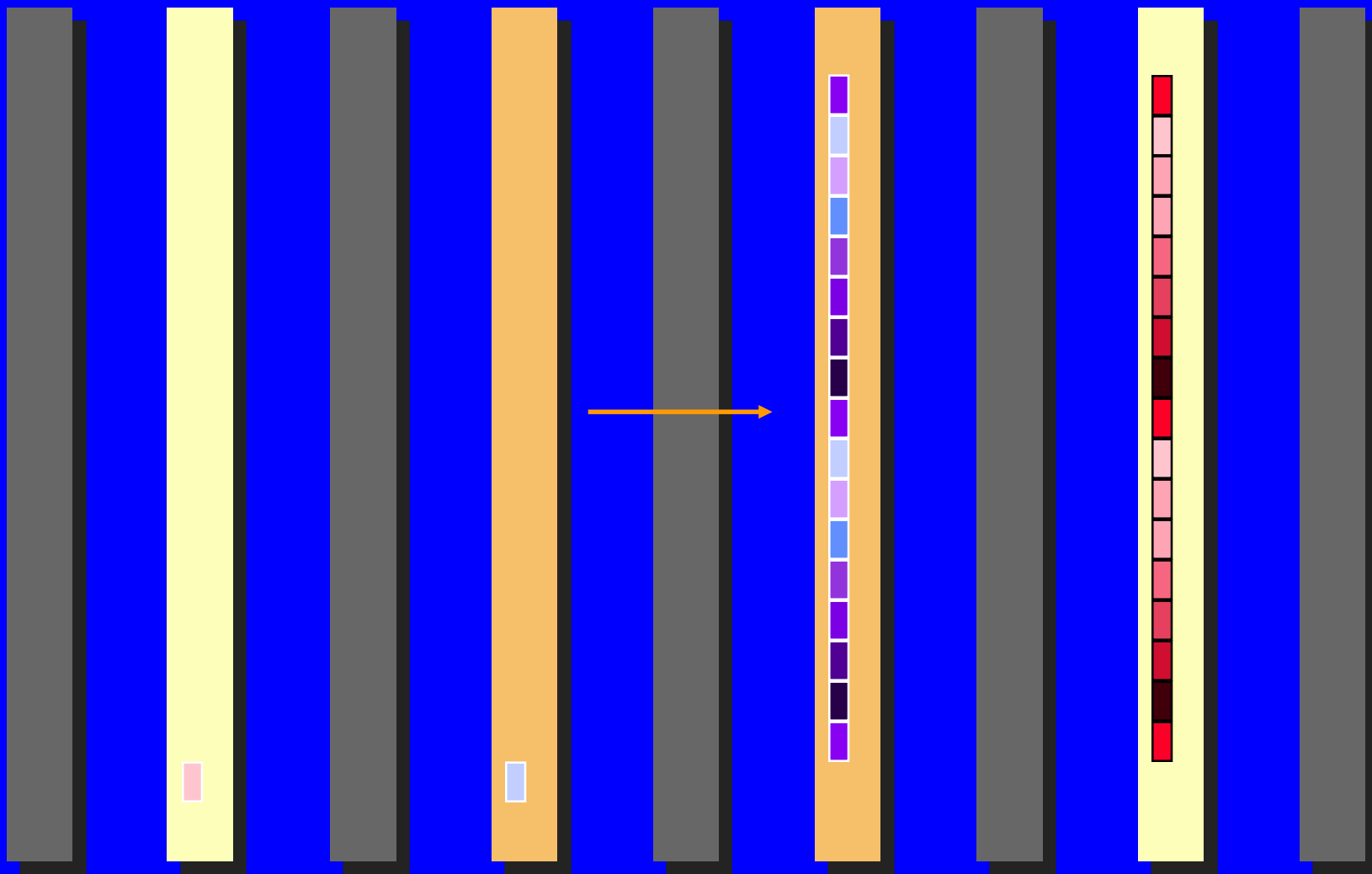


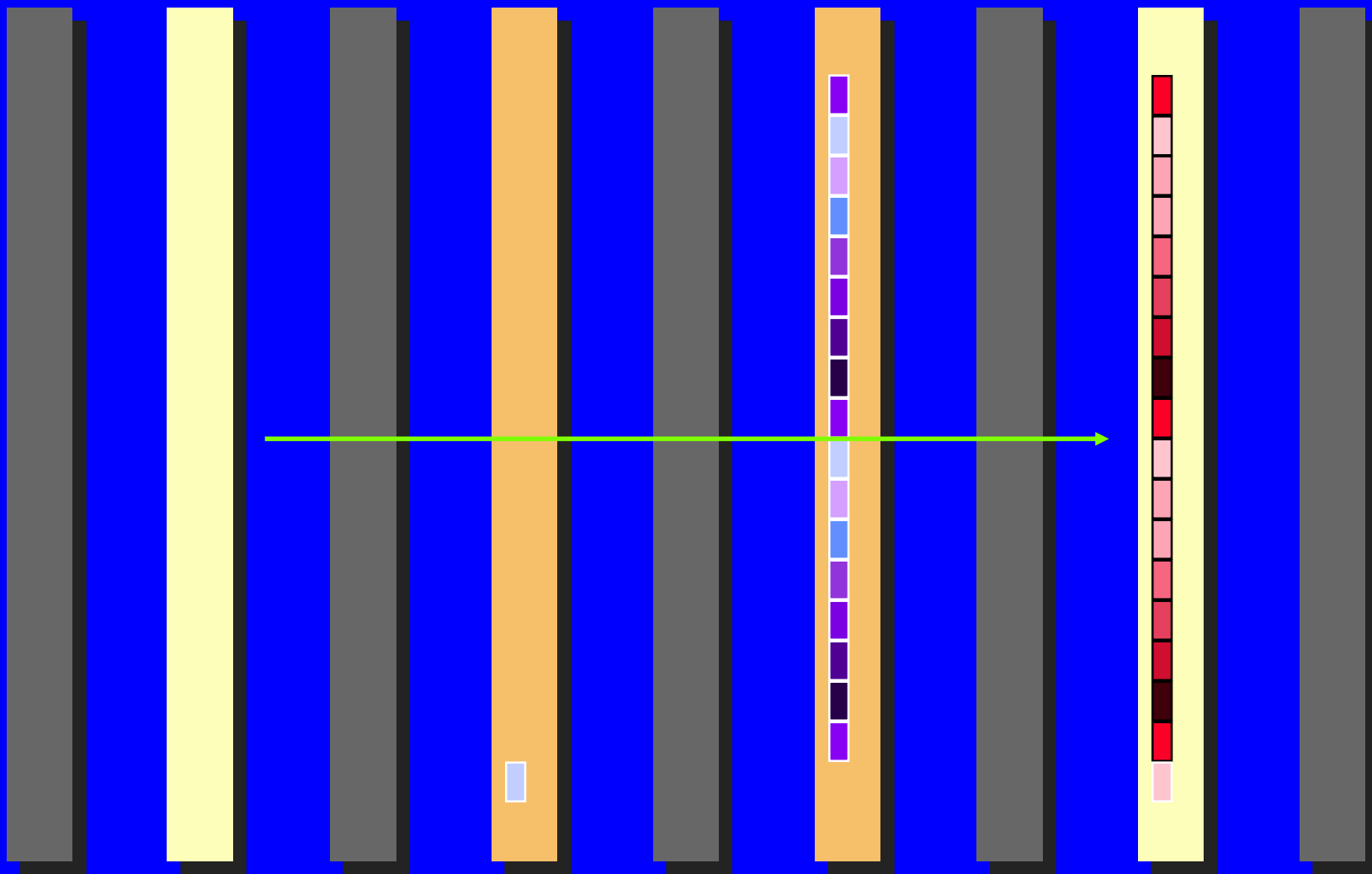


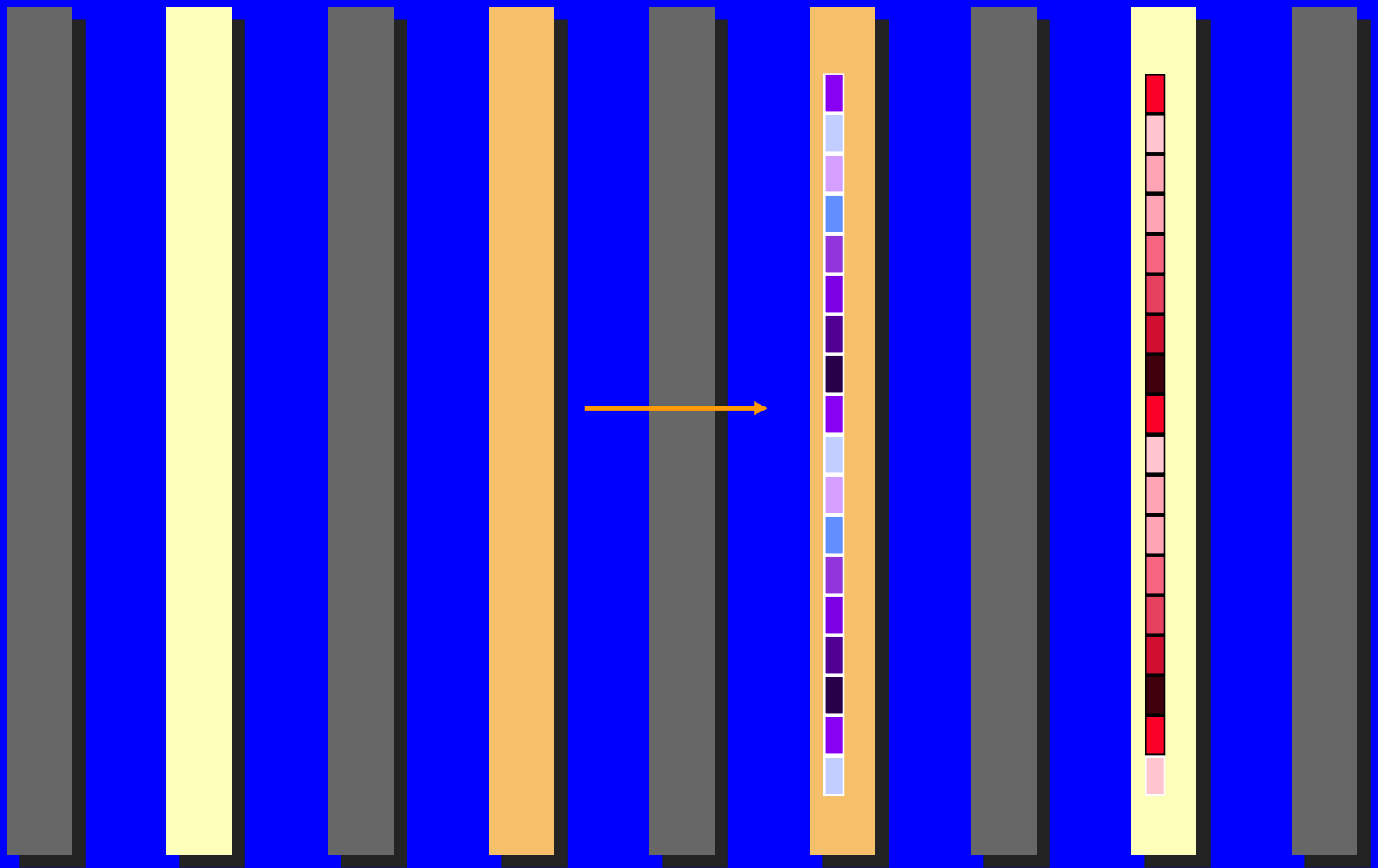


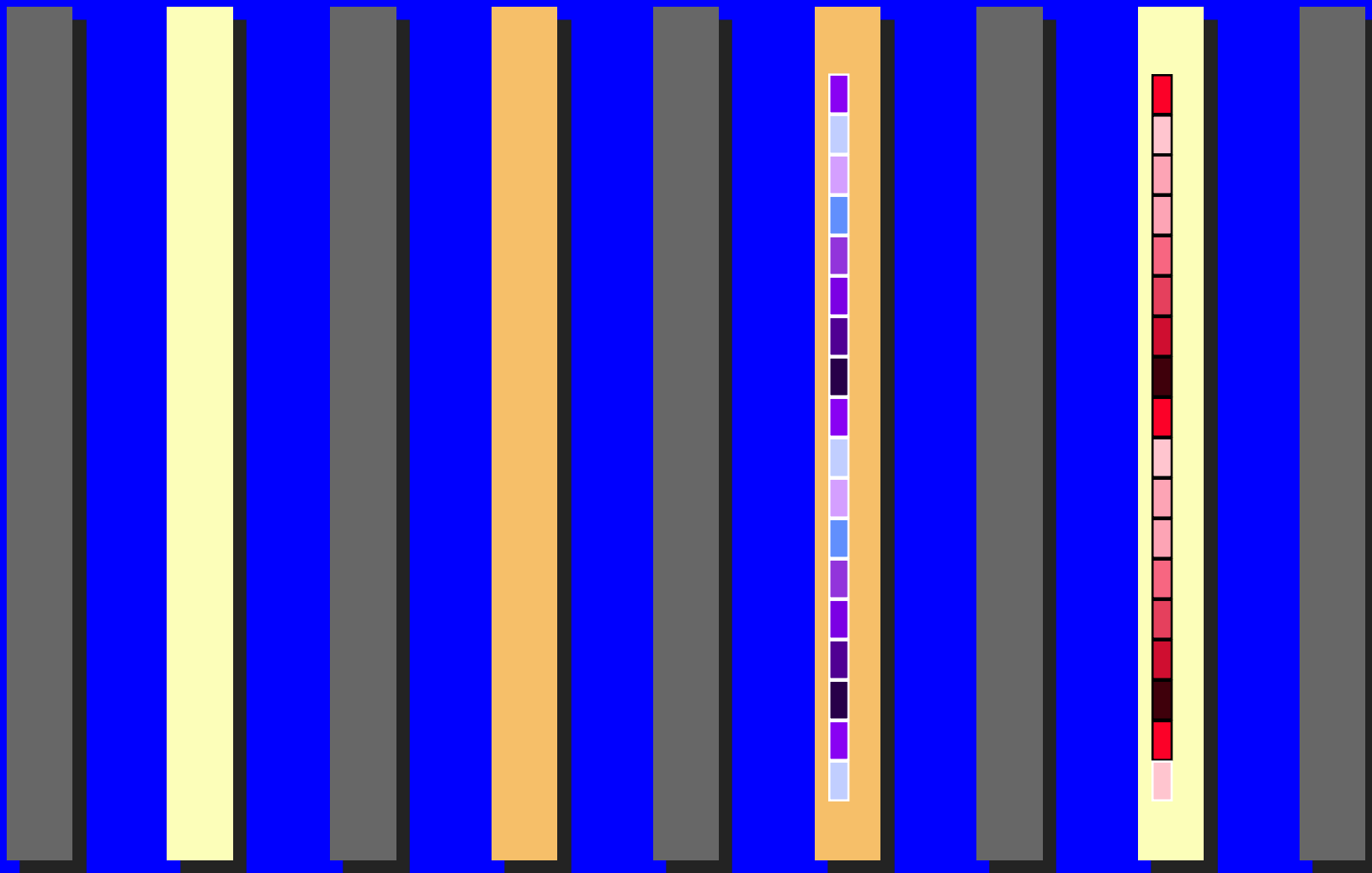












Outline

Part I: Theory

- Model of computation
- **Collective communications**
- A building block approach to library implementation

Part II: Practice

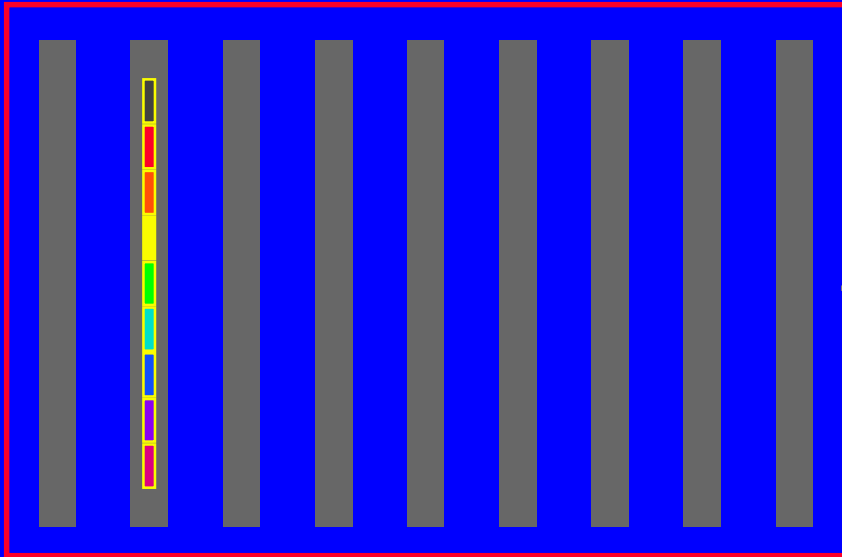
- Implementation on the Paragon
- Performance results
- Applications

Collective Communications

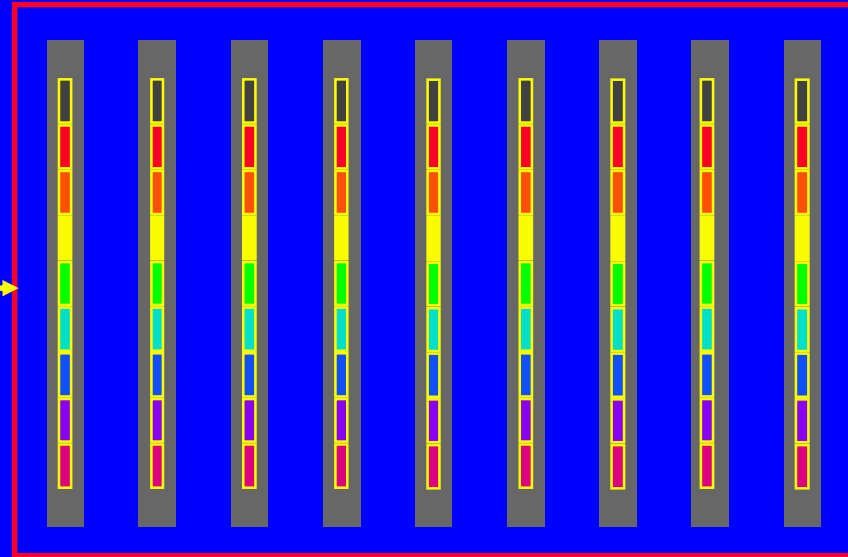
- Broadcast
- Reduce(-to-one)
- Scatter
- Gather
- Allgather
- Reduce-scatter
- Allreduce

Broadcast

Before

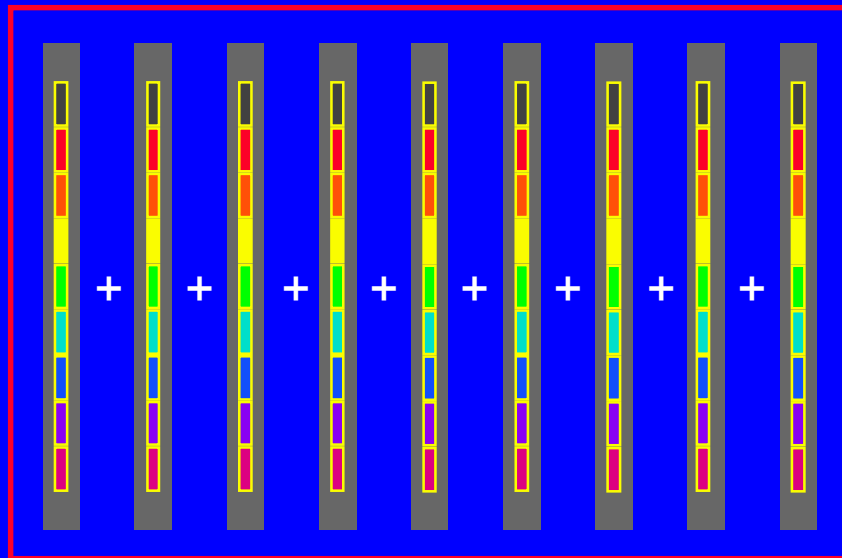


After

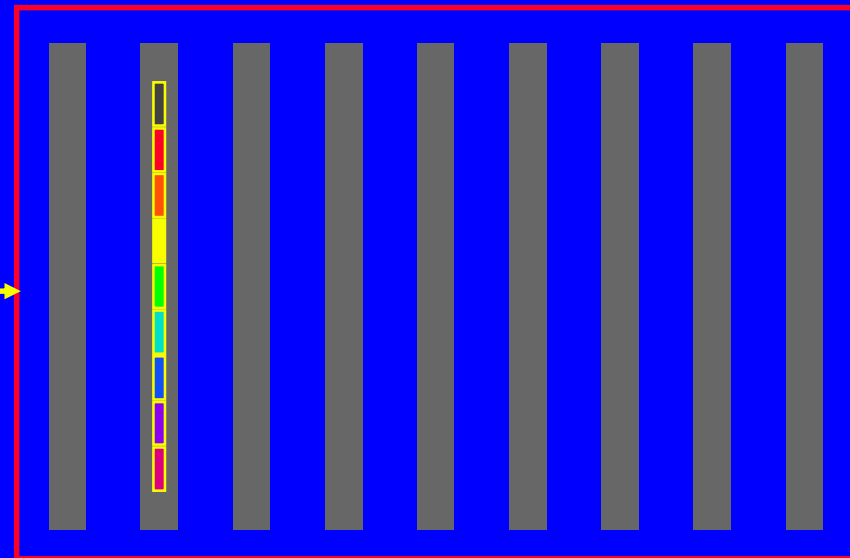


Reduce(-to-one)

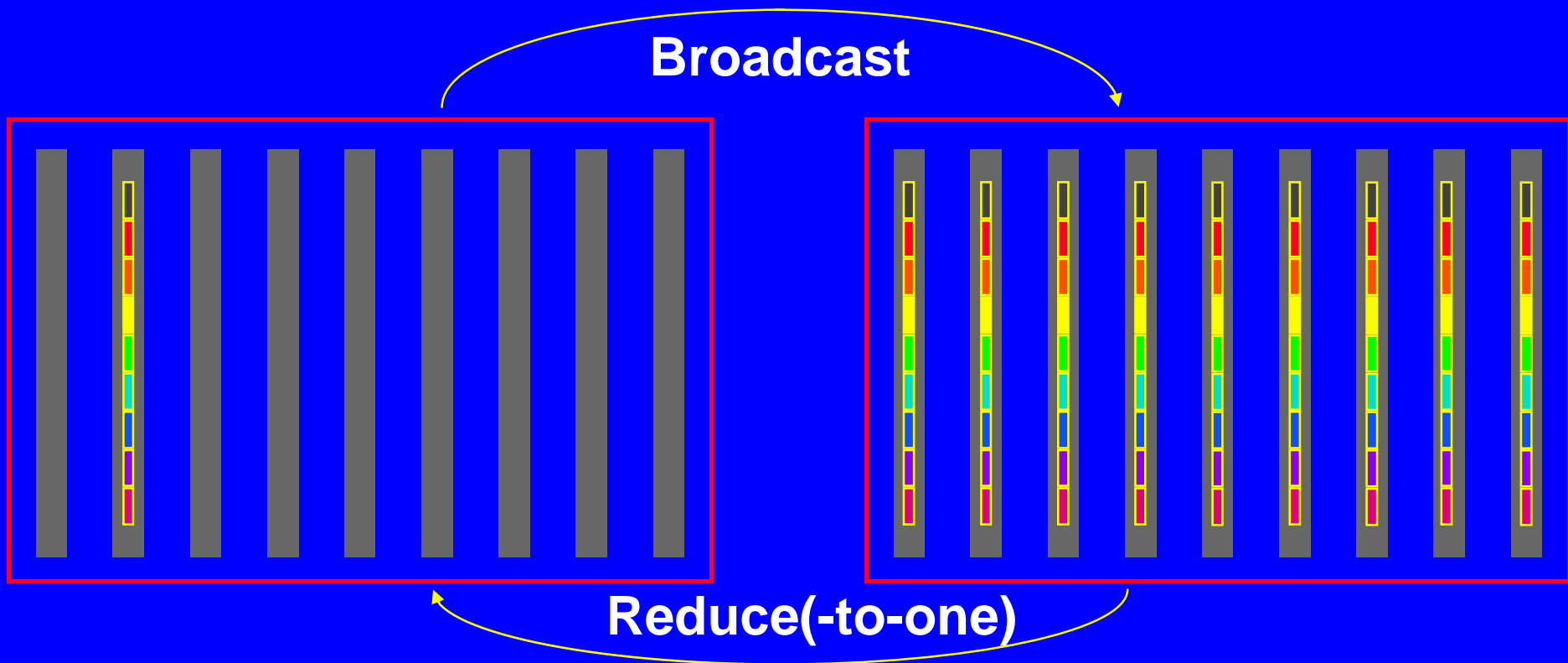
Before



After

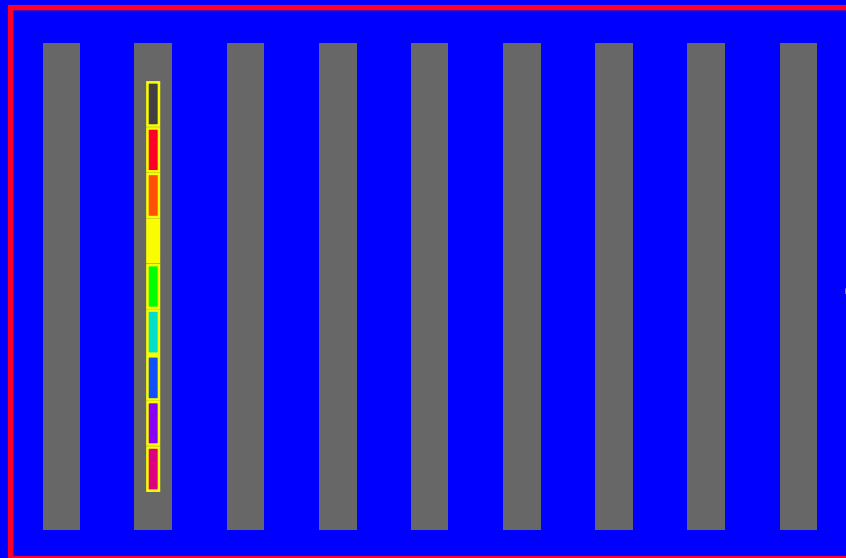


Broadcast/Reduce(-to-one)

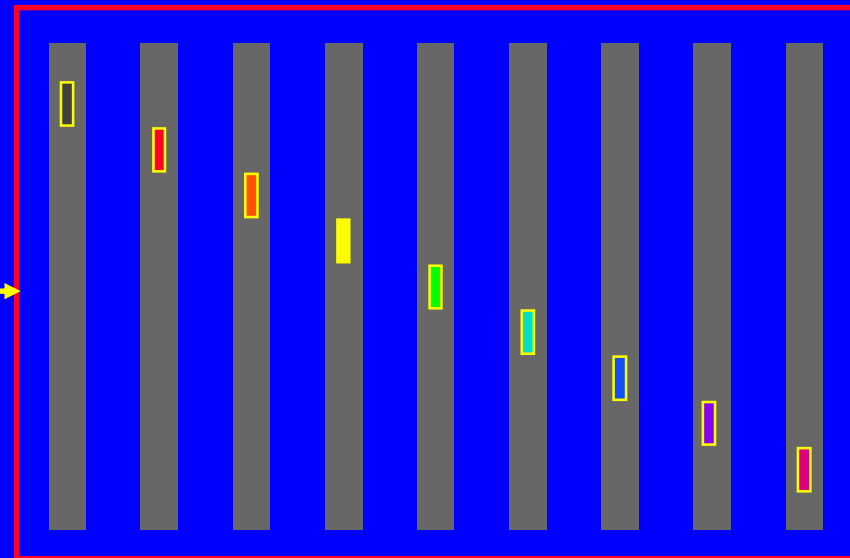


Scatter

Before

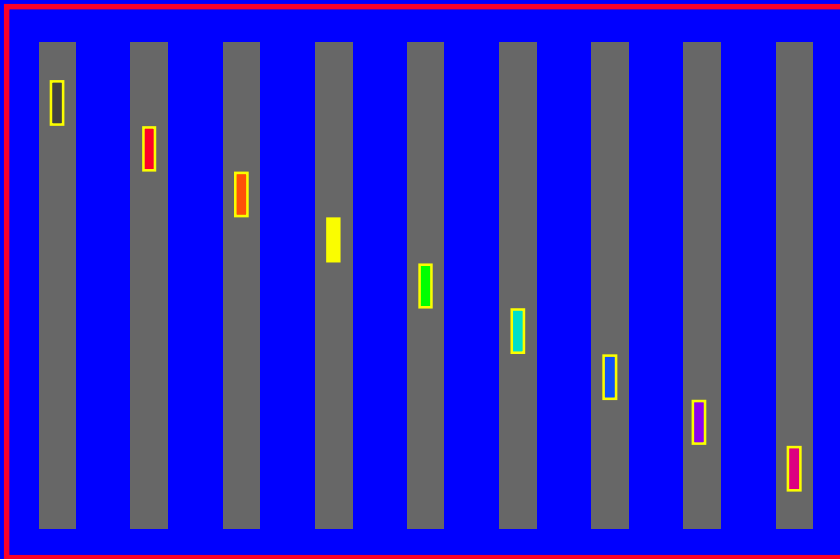


After

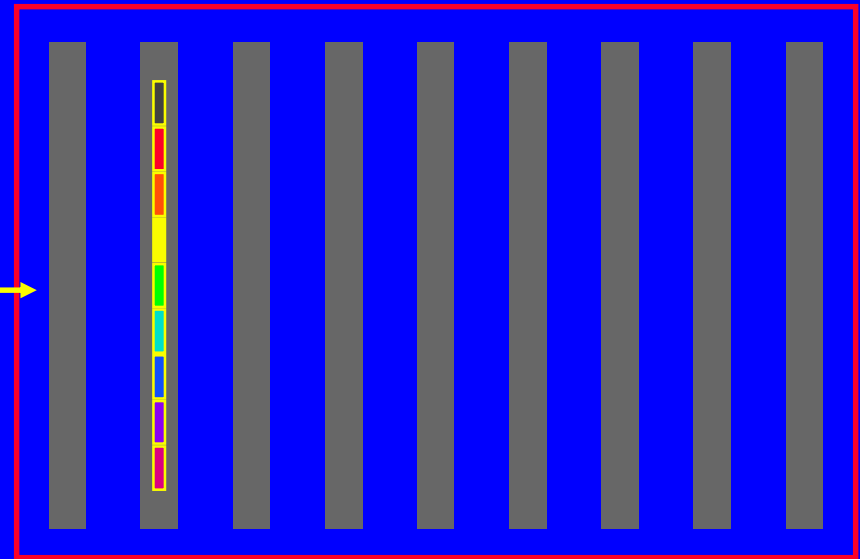


Gather

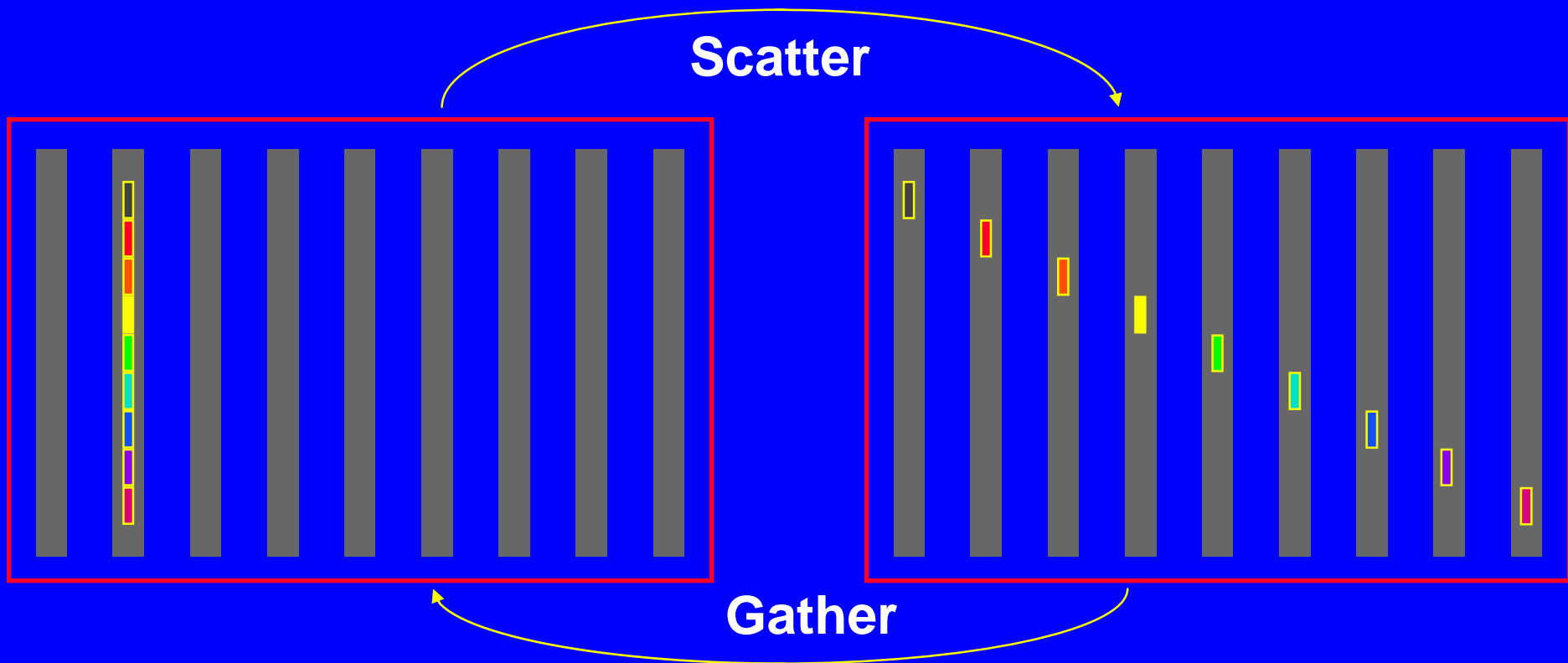
Before



After

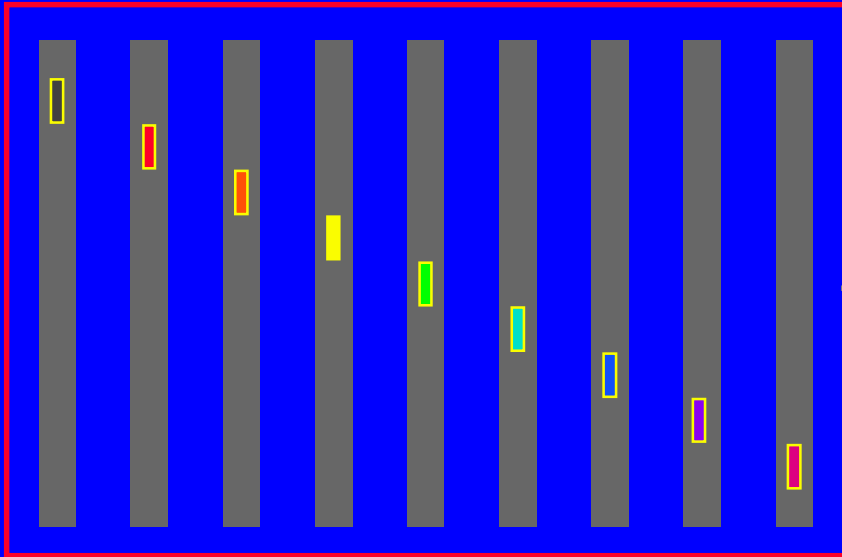


Scatter/Gather

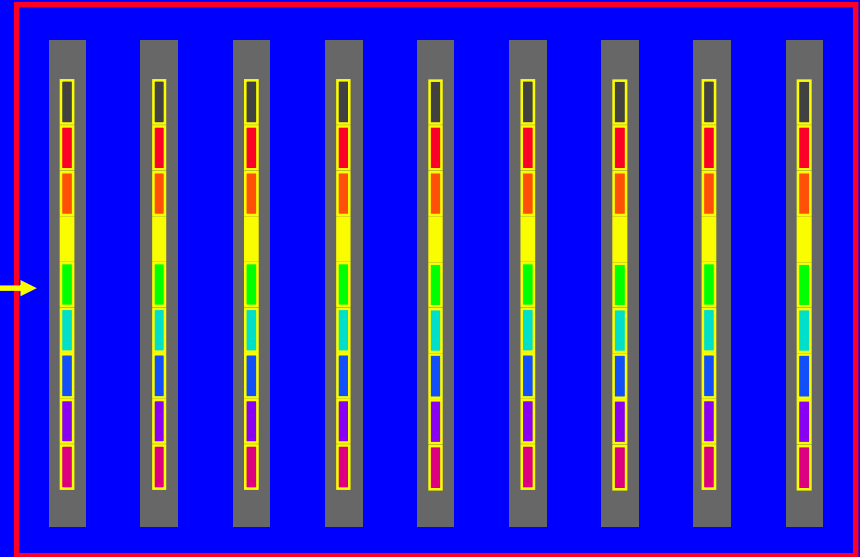


Allgather

Before

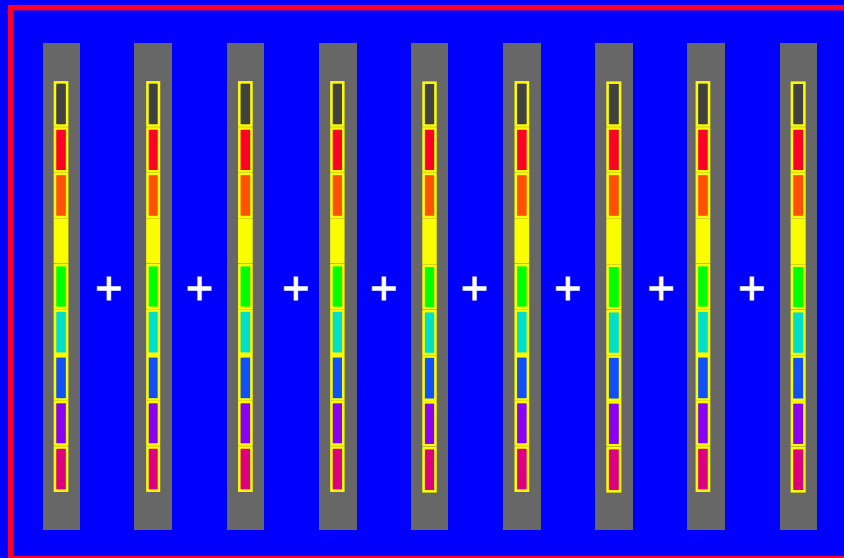


After

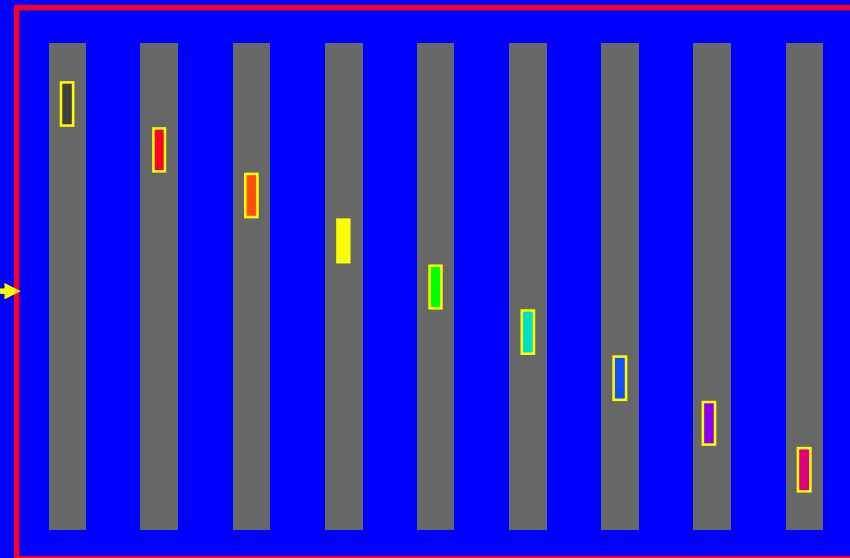


Reduce-scatter

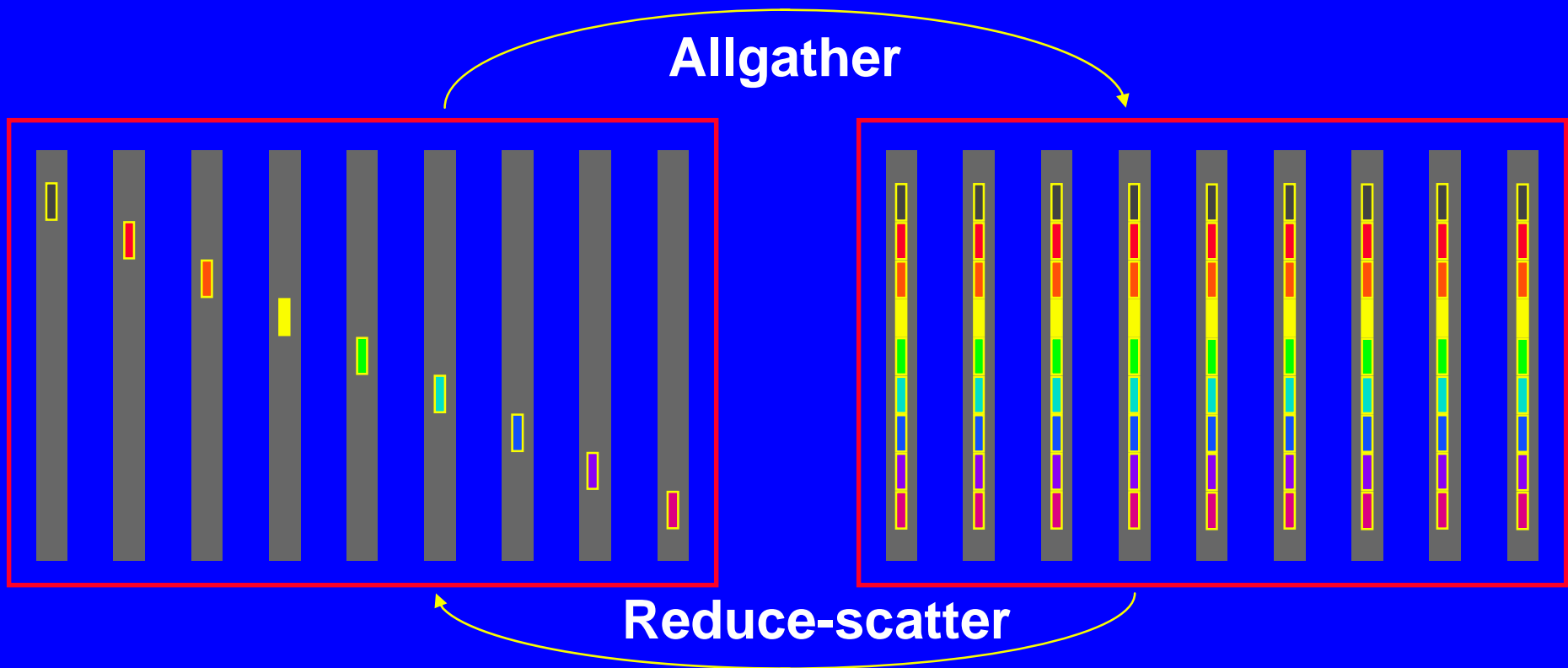
Before



After

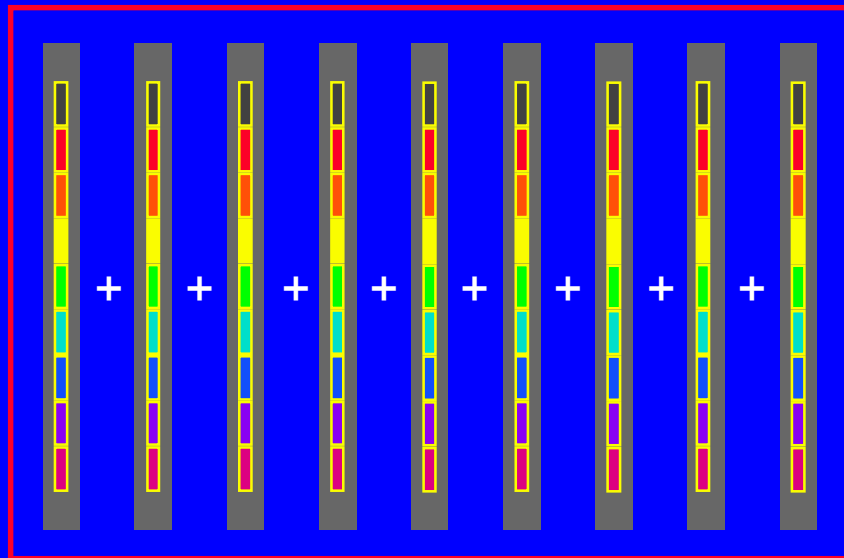


Allgather/Reduce-scatter

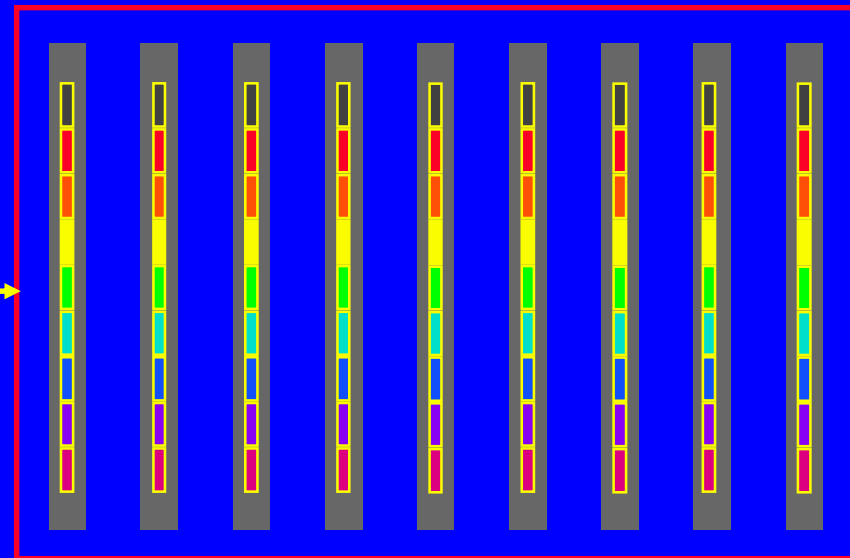


Allreduce

Before



After



Lower bounds (startup)

- Broadcast
- Reduce(-to-one)
- Scatter/Gather
- Allgather
- Reduce-scatter
- Allreduce

$$\lceil \log(p) \rceil \alpha$$

$$\lceil \log(p) \rceil \alpha$$

$$\lceil \log(p) \rceil \alpha$$

$$\lceil \log(p) \rceil \alpha$$

$$\lceil \log(p) \rceil \alpha$$

$$\lceil \log(p) \rceil \alpha$$

Lower bounds (bandwidth)

- Broadcast
- Reduce(-to-one)
- Scatter/Gather
- Allgather
- Reduce-scatter
- Allreduce

$$n\beta$$

$$n\beta + \frac{p-1}{p}n\gamma$$

$$\frac{p-1}{p}n\beta$$

$$\frac{p-1}{p}n\beta$$

$$\frac{p-1}{p}n\beta + \frac{p-1}{p}n\gamma$$

$$2\frac{p-1}{p}n\beta + \frac{p-1}{p}n\gamma$$

Outline

Part I: Theory

- Model of computation
- Collective communications
- **A building block approach to library implementation**

Part II: Practice

- Implementation on the Paragon
- Performance results
- Applications

A building block approach to library implementation

- Short vector case
- Long vector case
- Hybrid algorithms

Short vector case

- **Primary concern:**
 - algorithms must have low latency cost
- **Secondary concerns:**
 - algorithms must work for arbitrary number of nodes
 - » in particular, not just for power-of-two numbers of nodes
 - algorithms should avoid network conflicts
 - » not absolutely necessary, but nice if possible

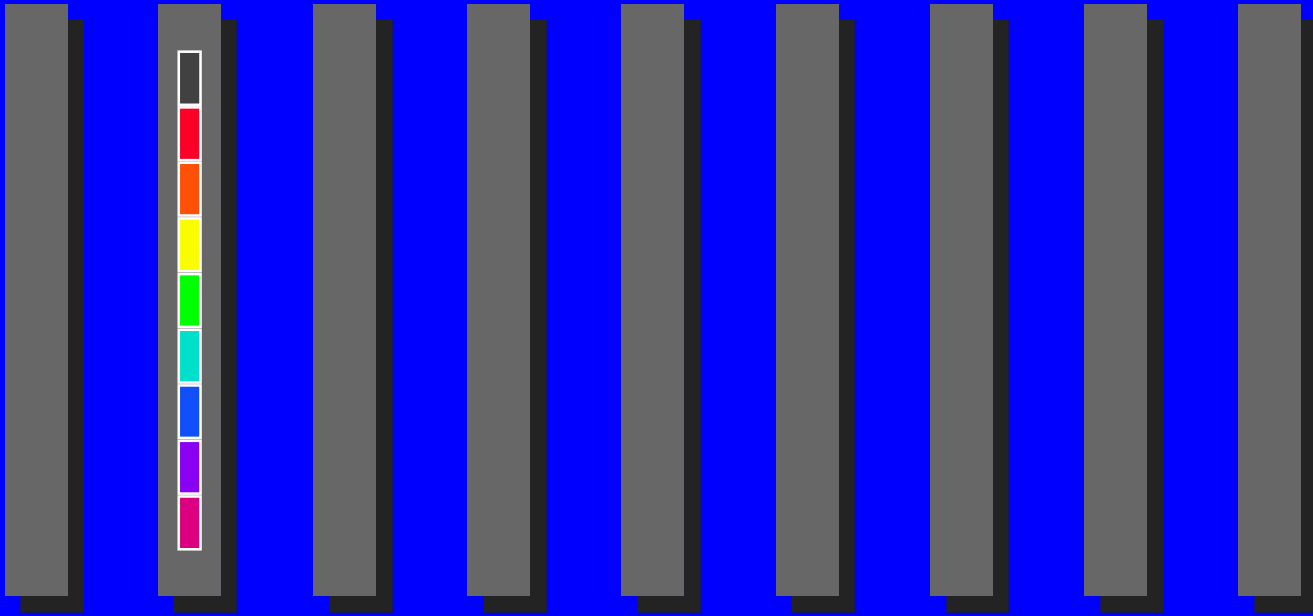
Minimum spanning tree based algorithms

- We will show how the following building blocks:
 - broadcast/combine-to-one
 - scatter/gather

can be implemented using minimum spanning trees embedded in the logical linear array while attaining

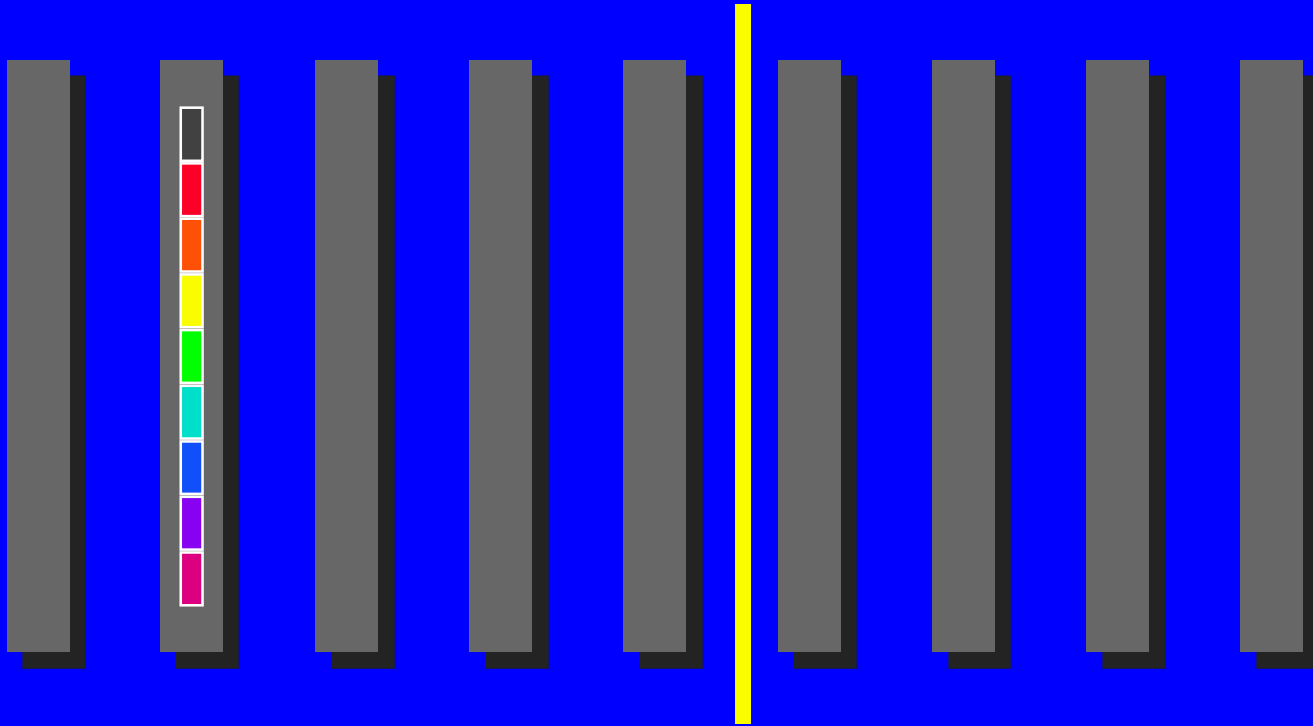
- minimal latency
- implementation for arbitrary numbers of nodes
- no network conflicts

General principles



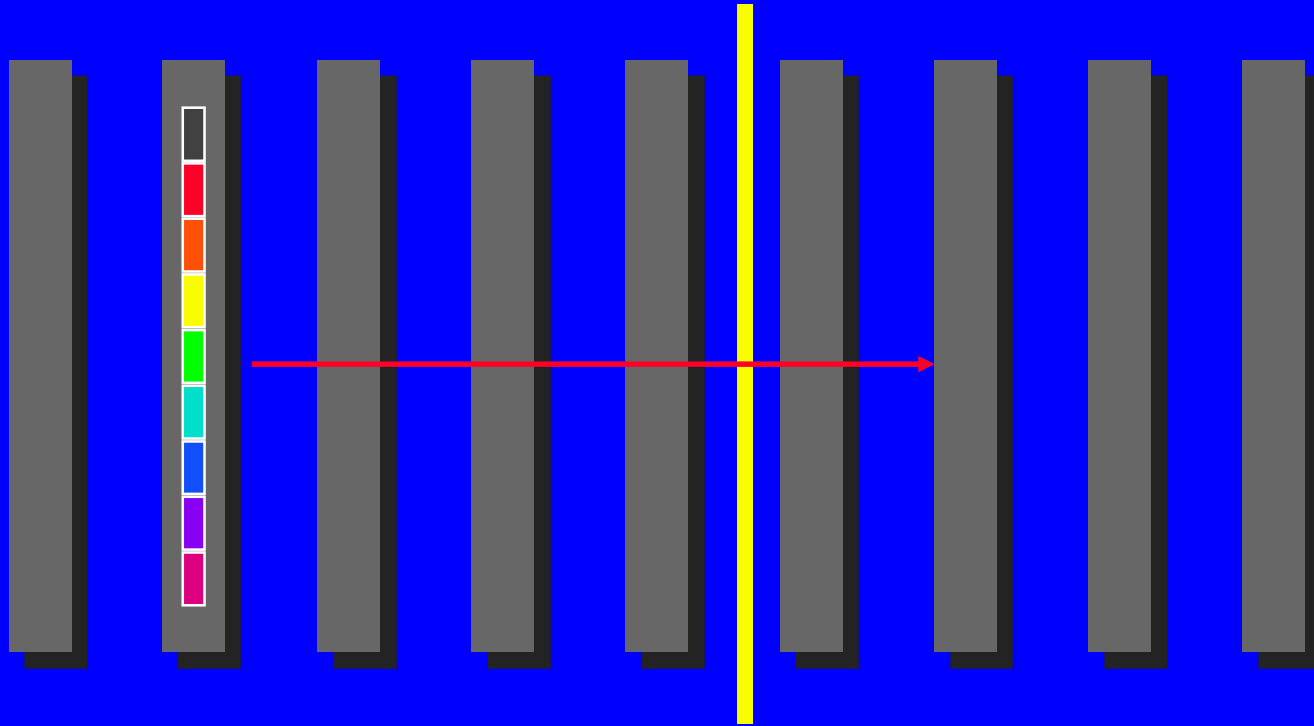
- message starts on one processor

General principles



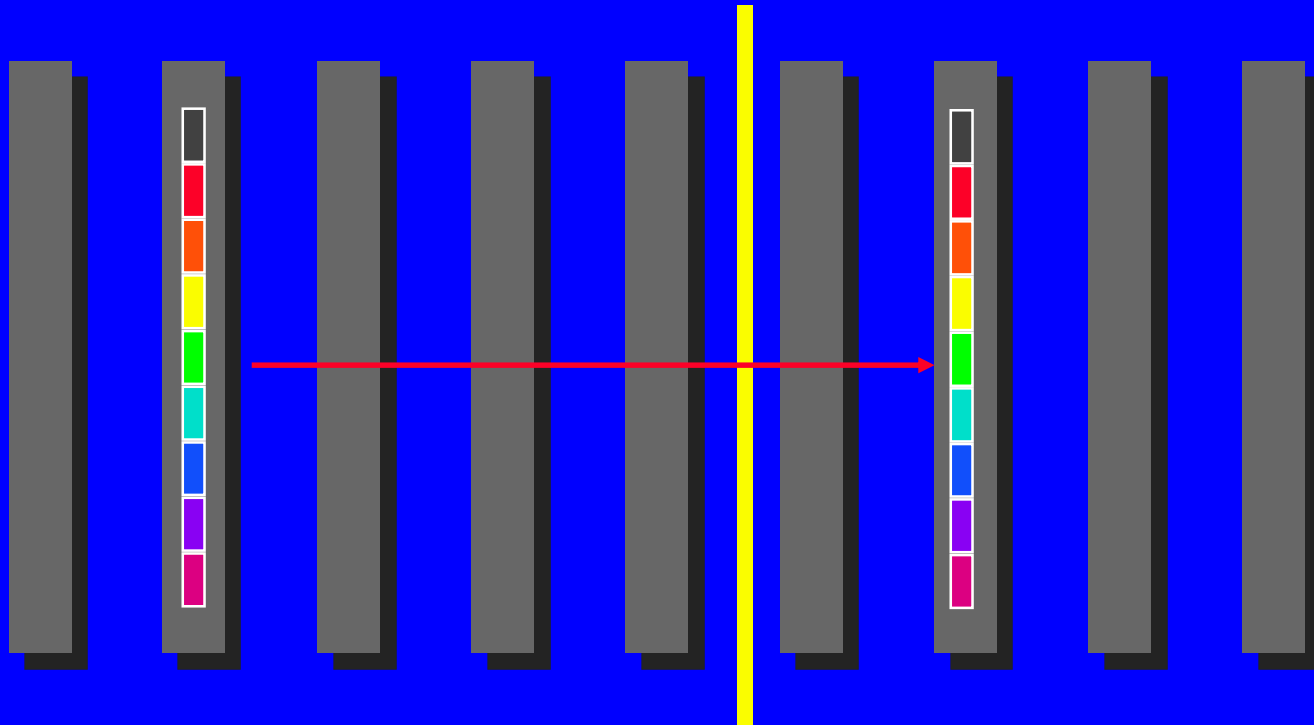
- divide logical linear array in half

General principles



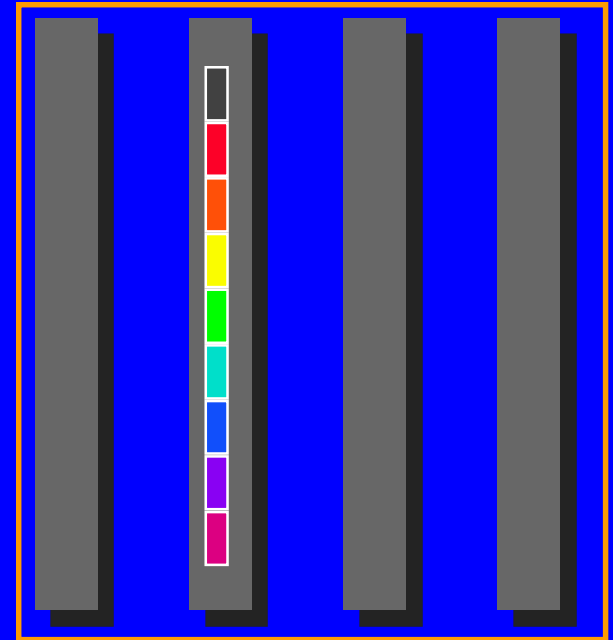
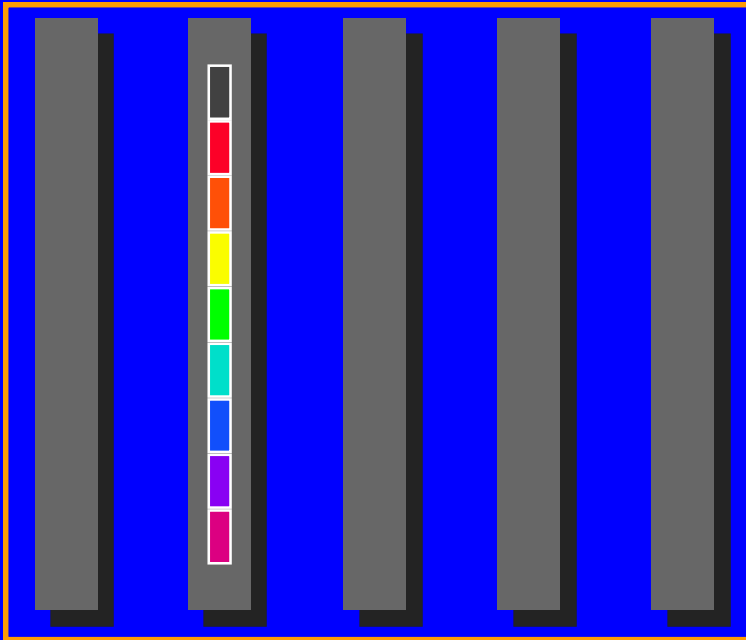
- send message to the half of the network that does not contain the current node (root) that holds the message

General principles



- send message to the half of the network that does not contain the current node (root) that holds the message

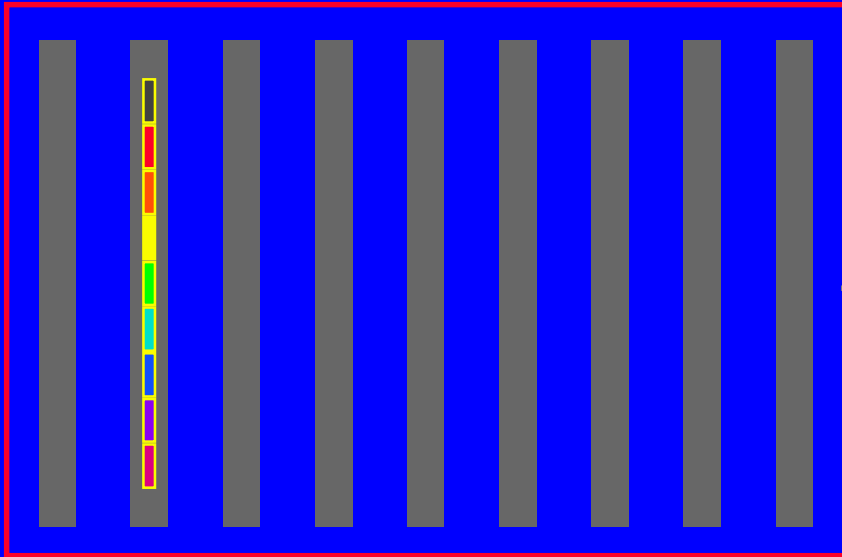
General principles



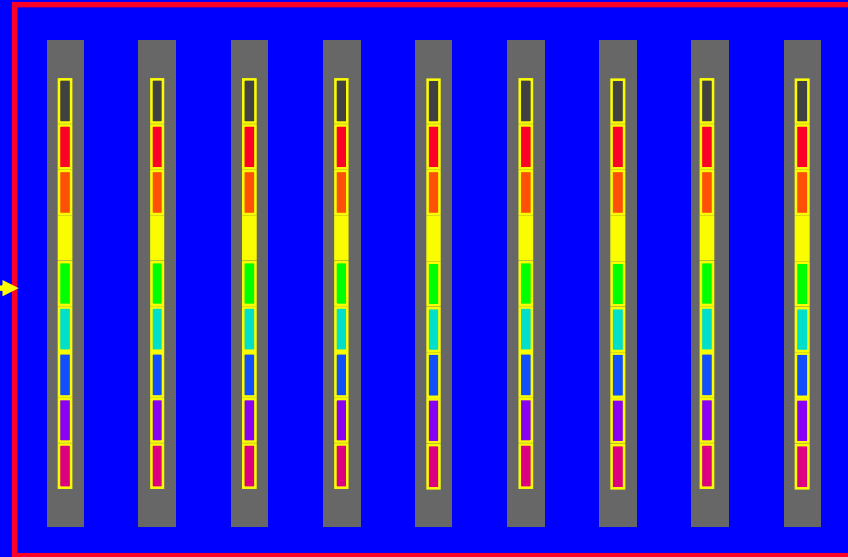
- continue recursively in each of the two halves

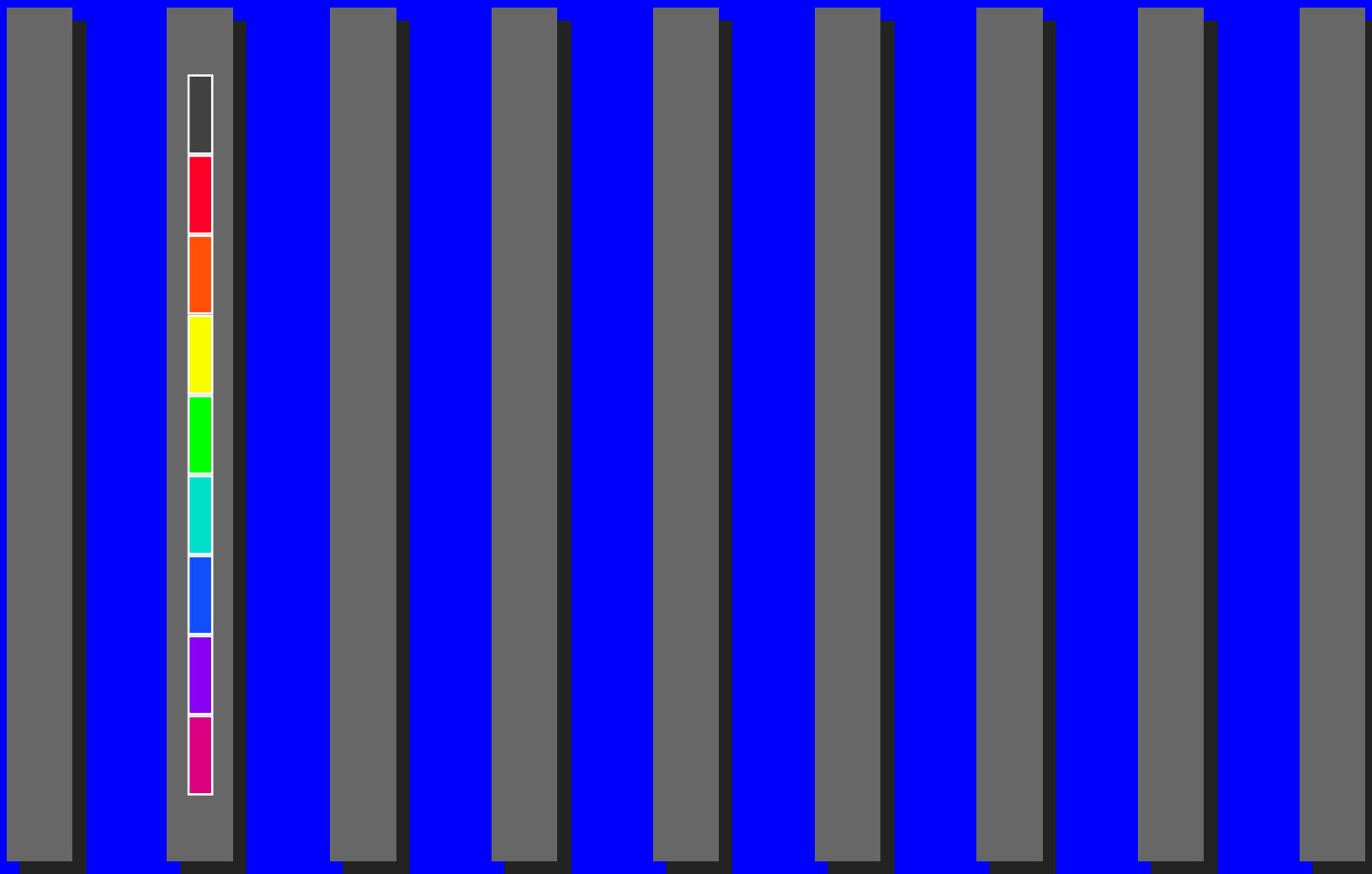
Broadcast

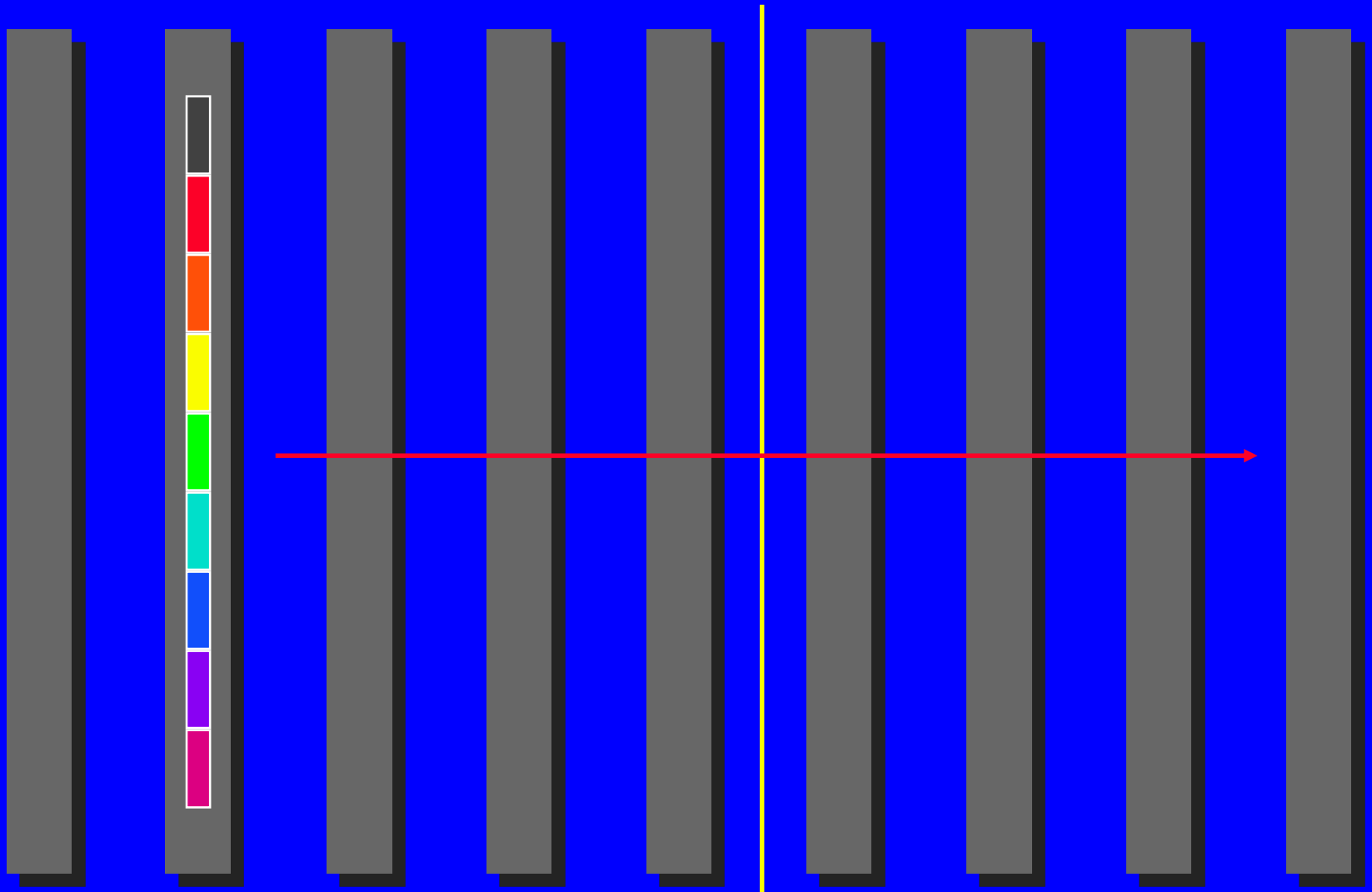
Before

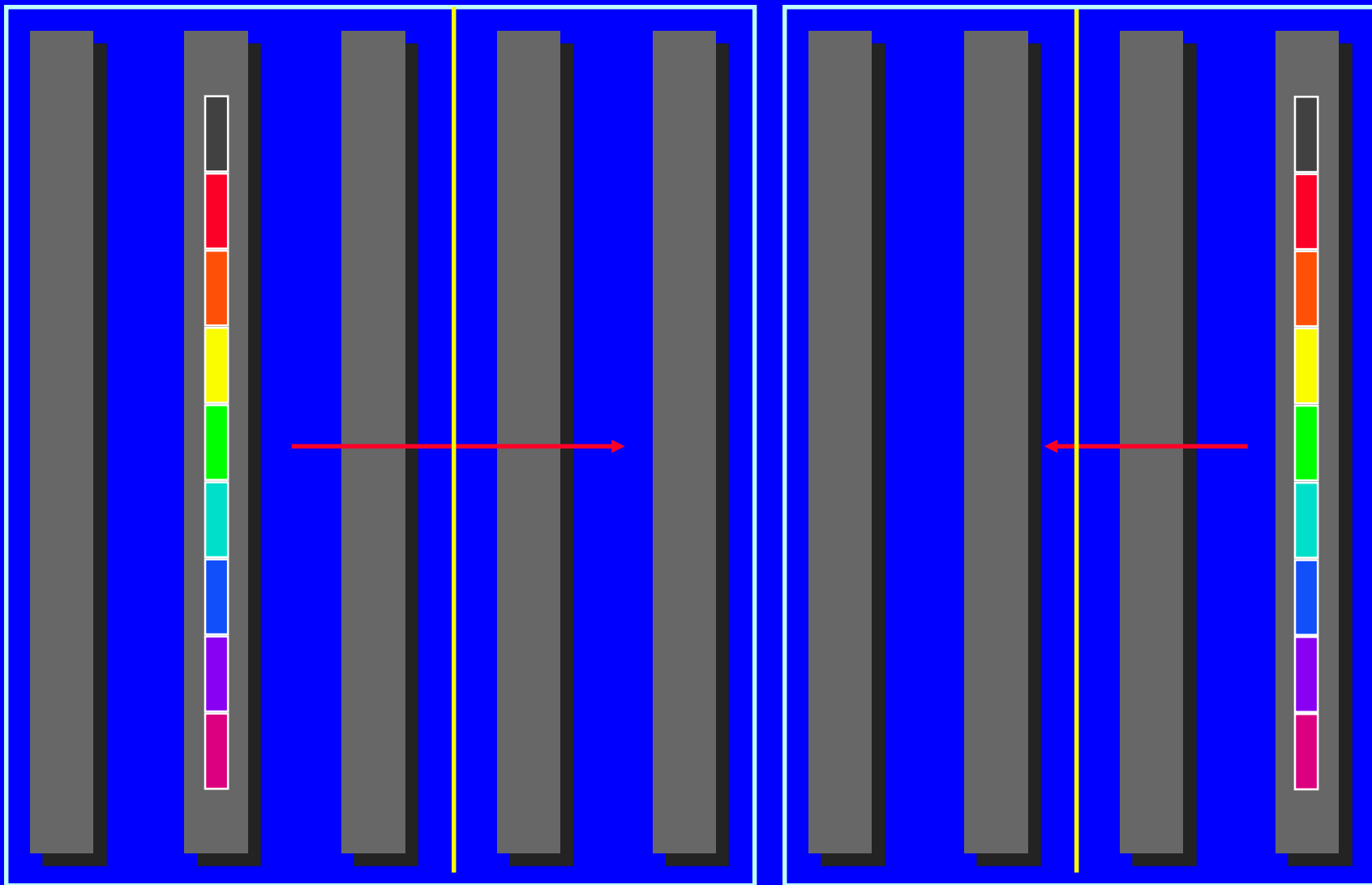


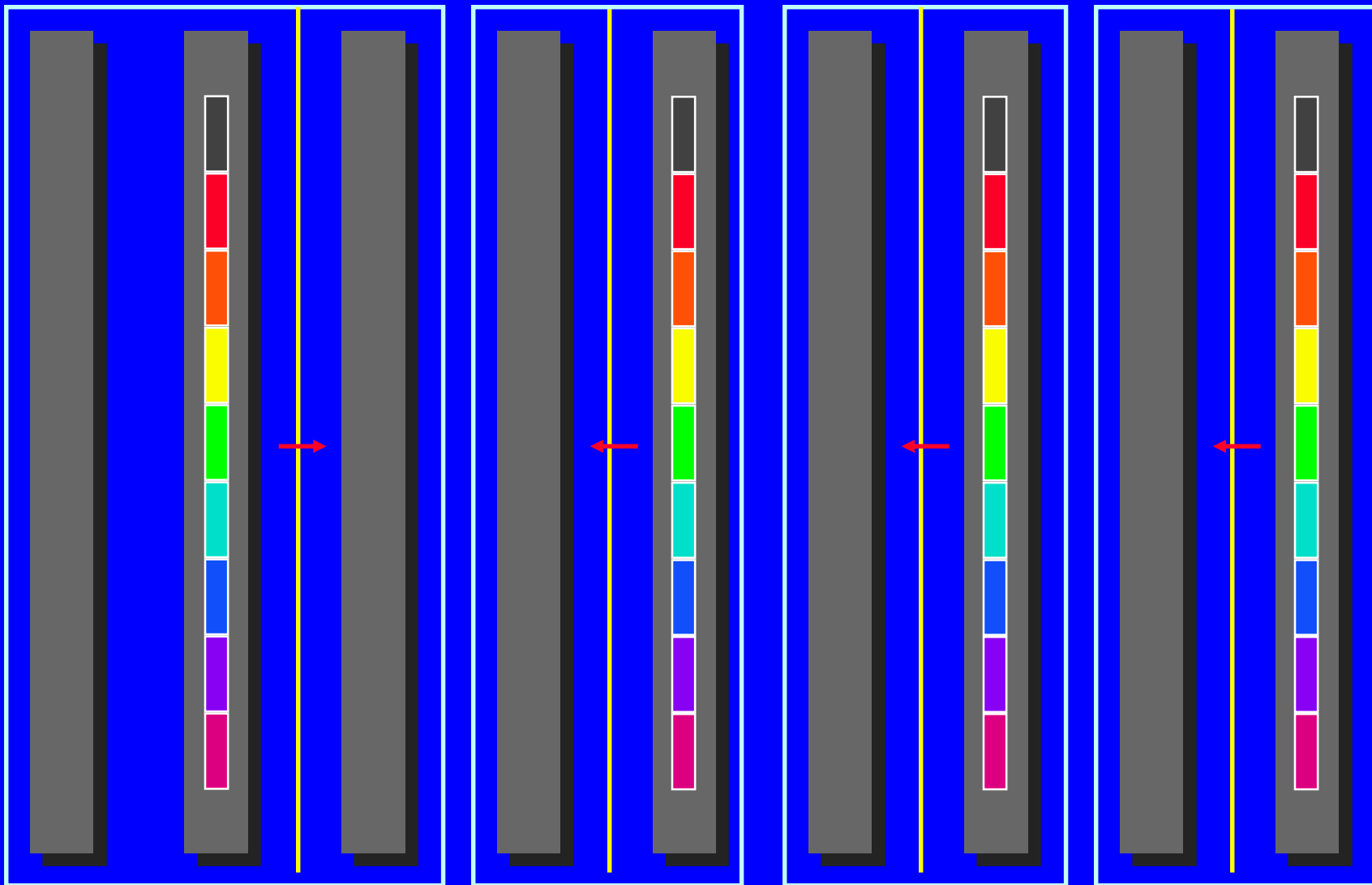
After

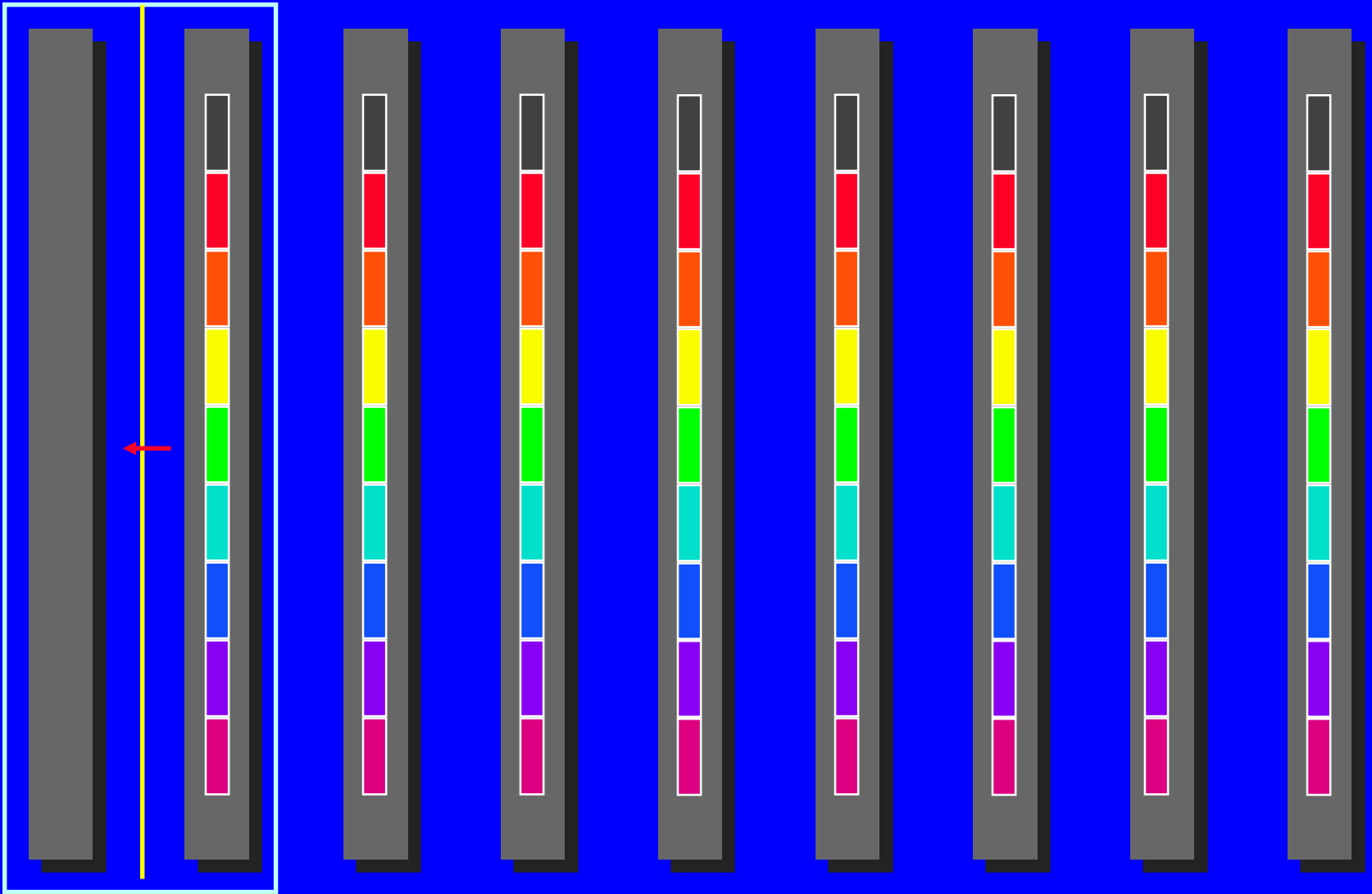


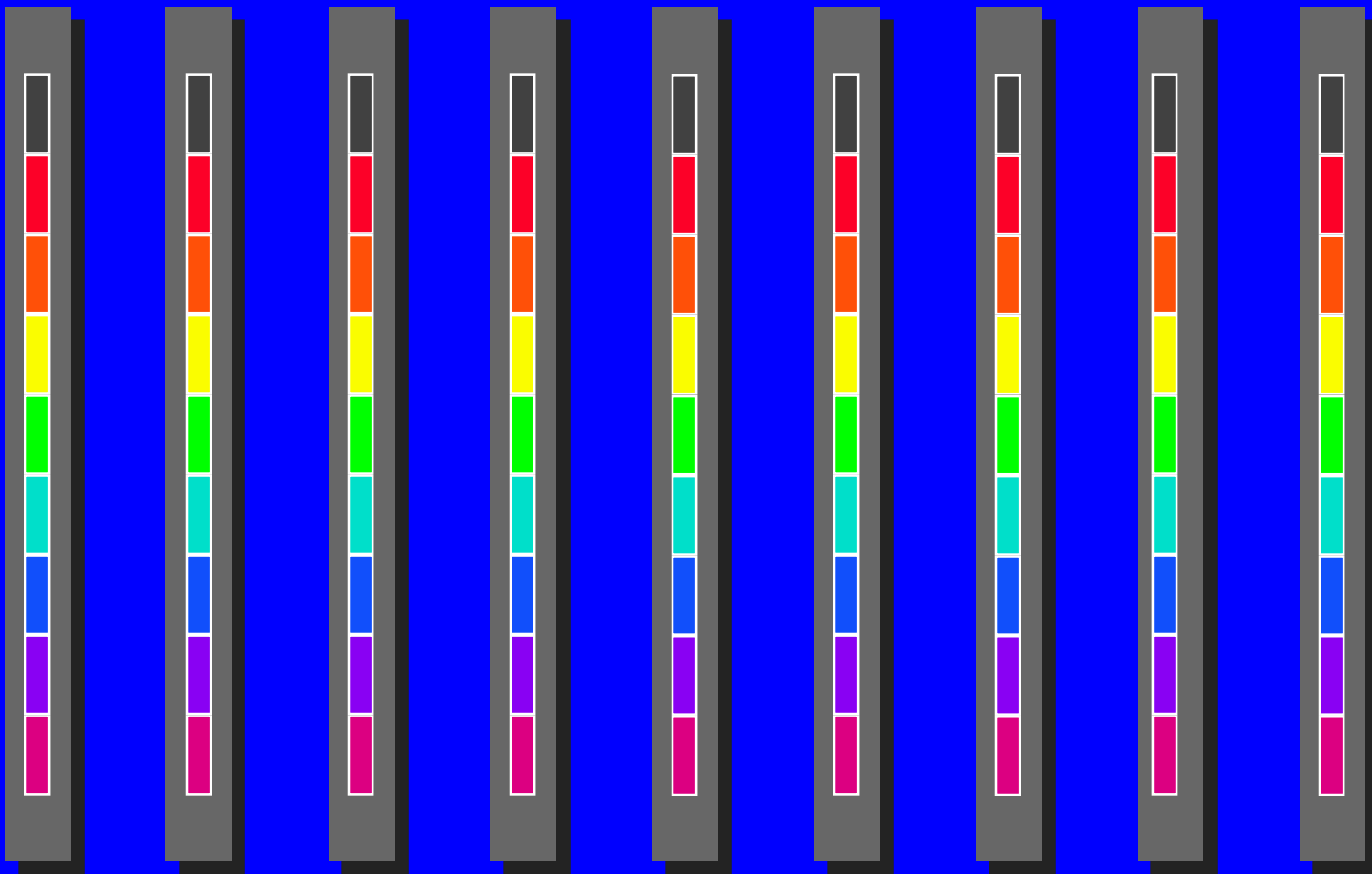






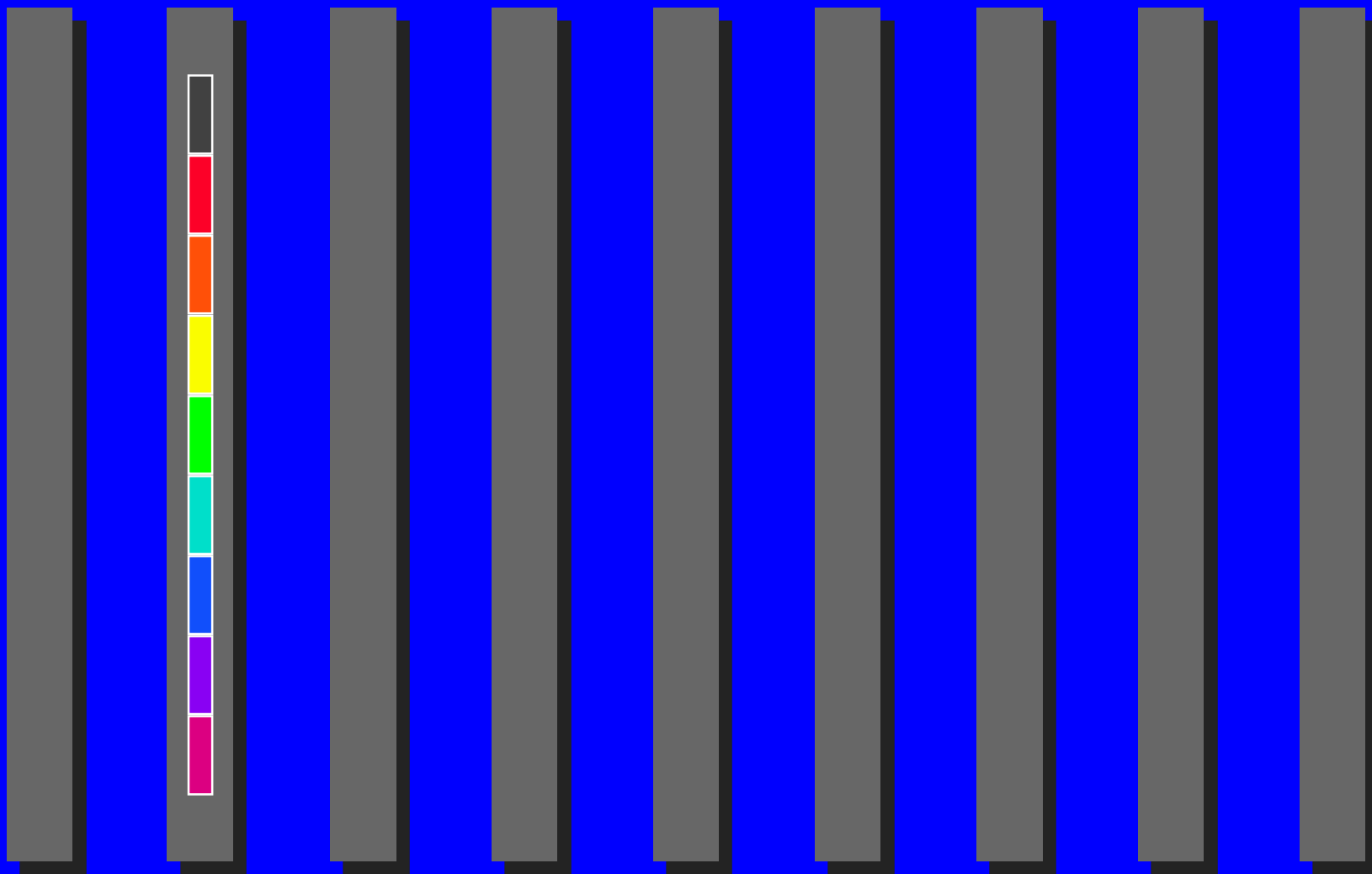


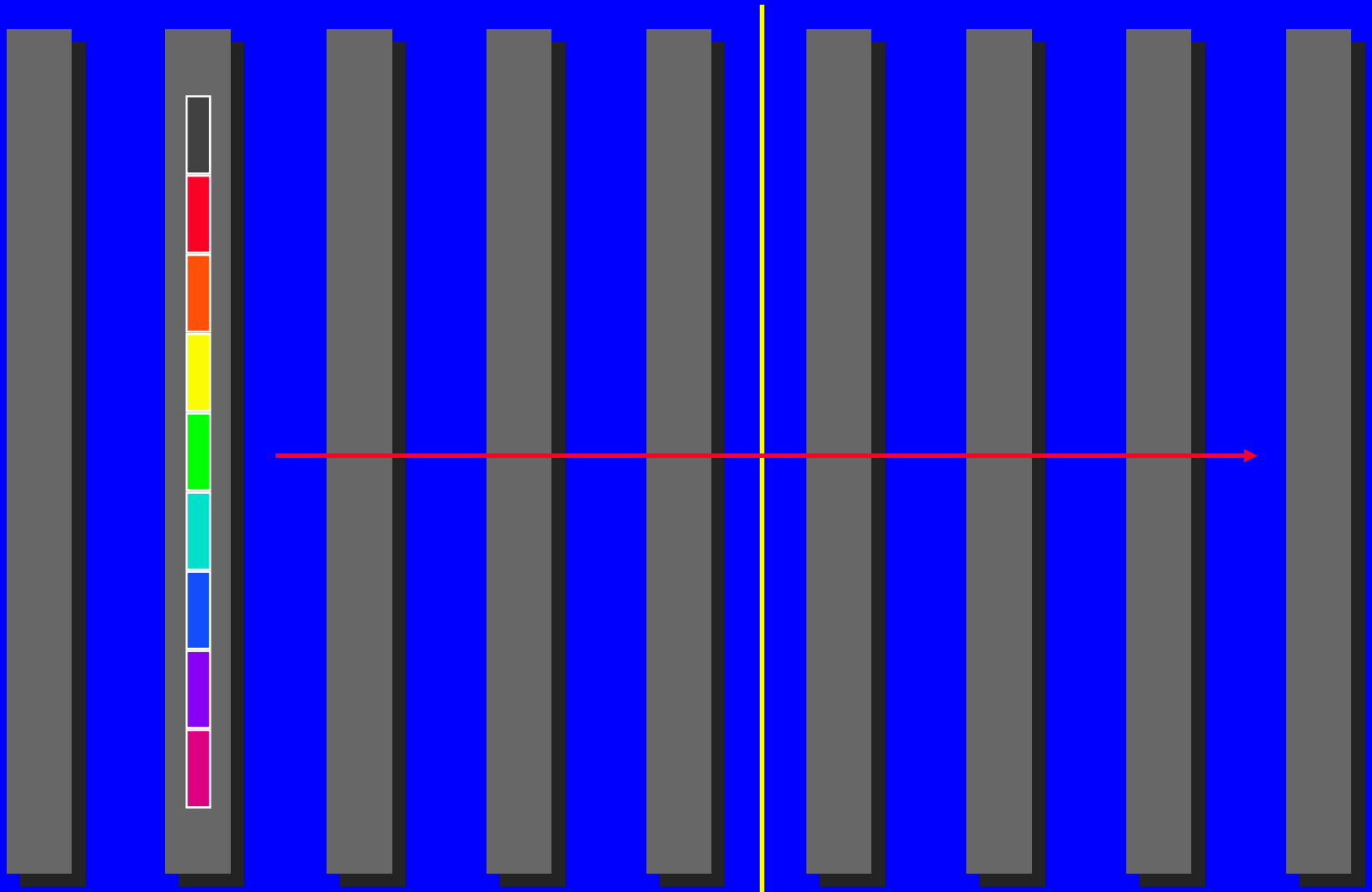


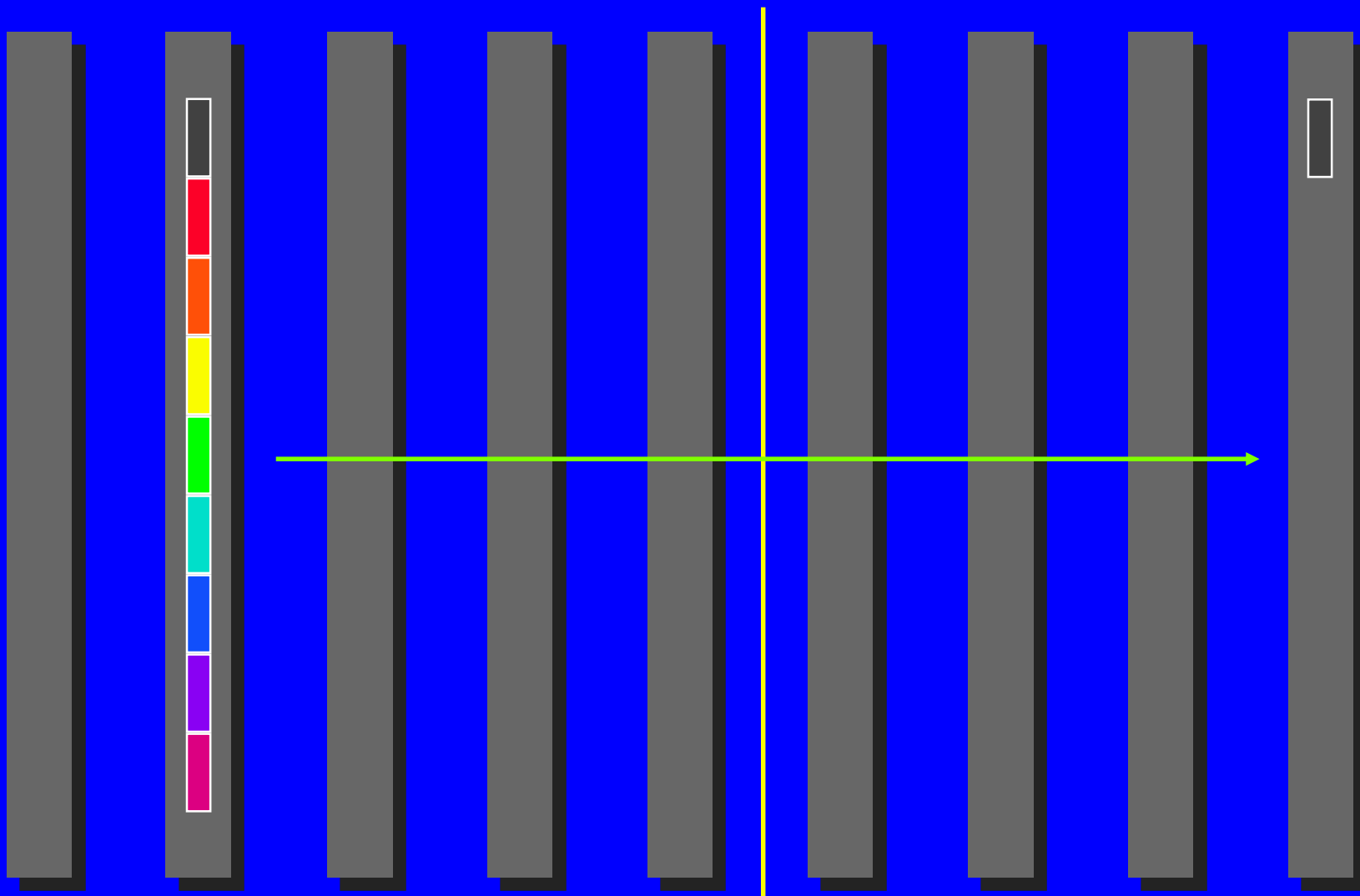


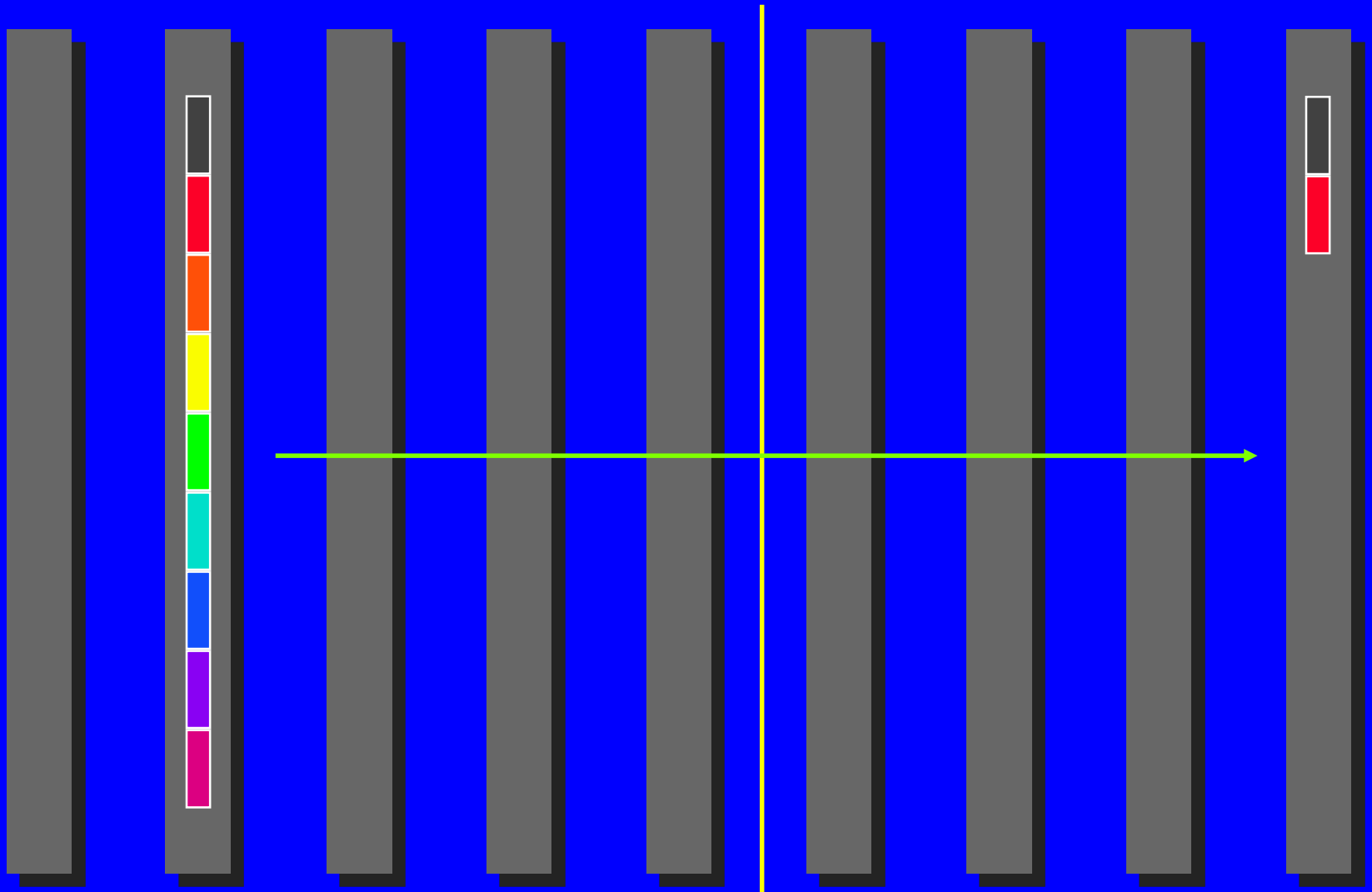
Let us view this more closely

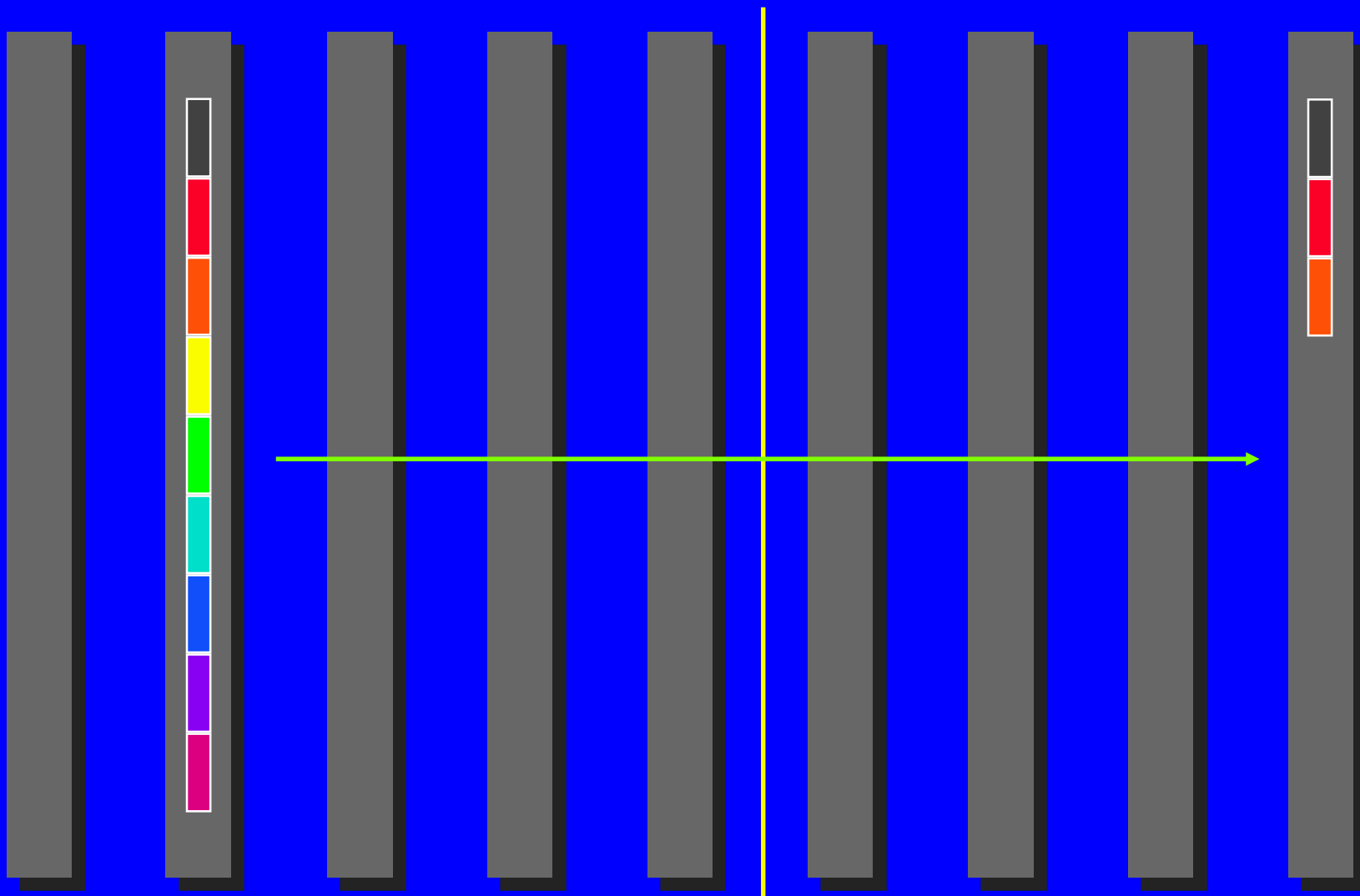
- **Red** arrows indicate startup of communication (leading to latency, α)
- **Green** arrows indicate packets in transit (leading to a bandwidth related cost proportional to β and the length of the packet)

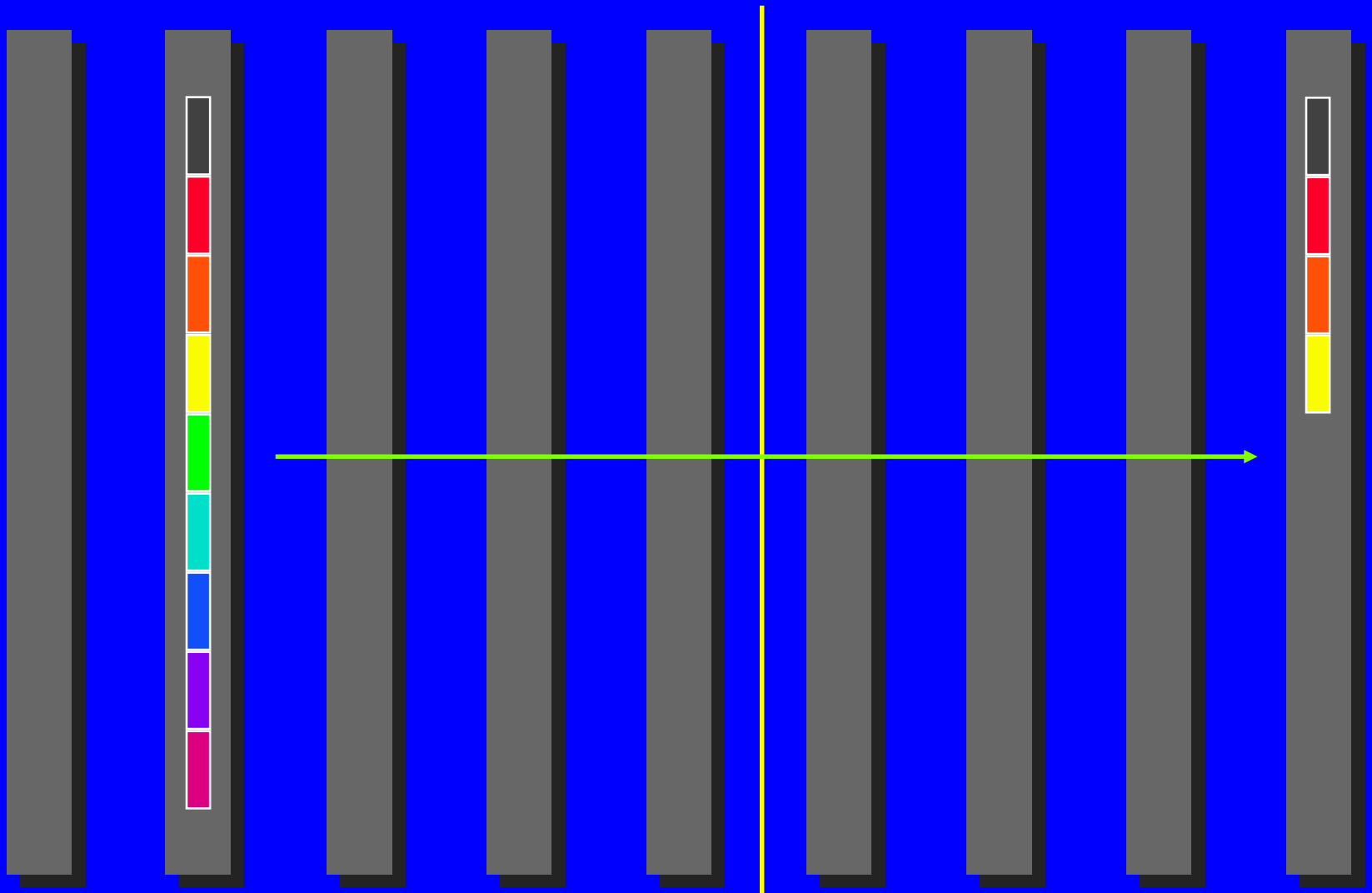


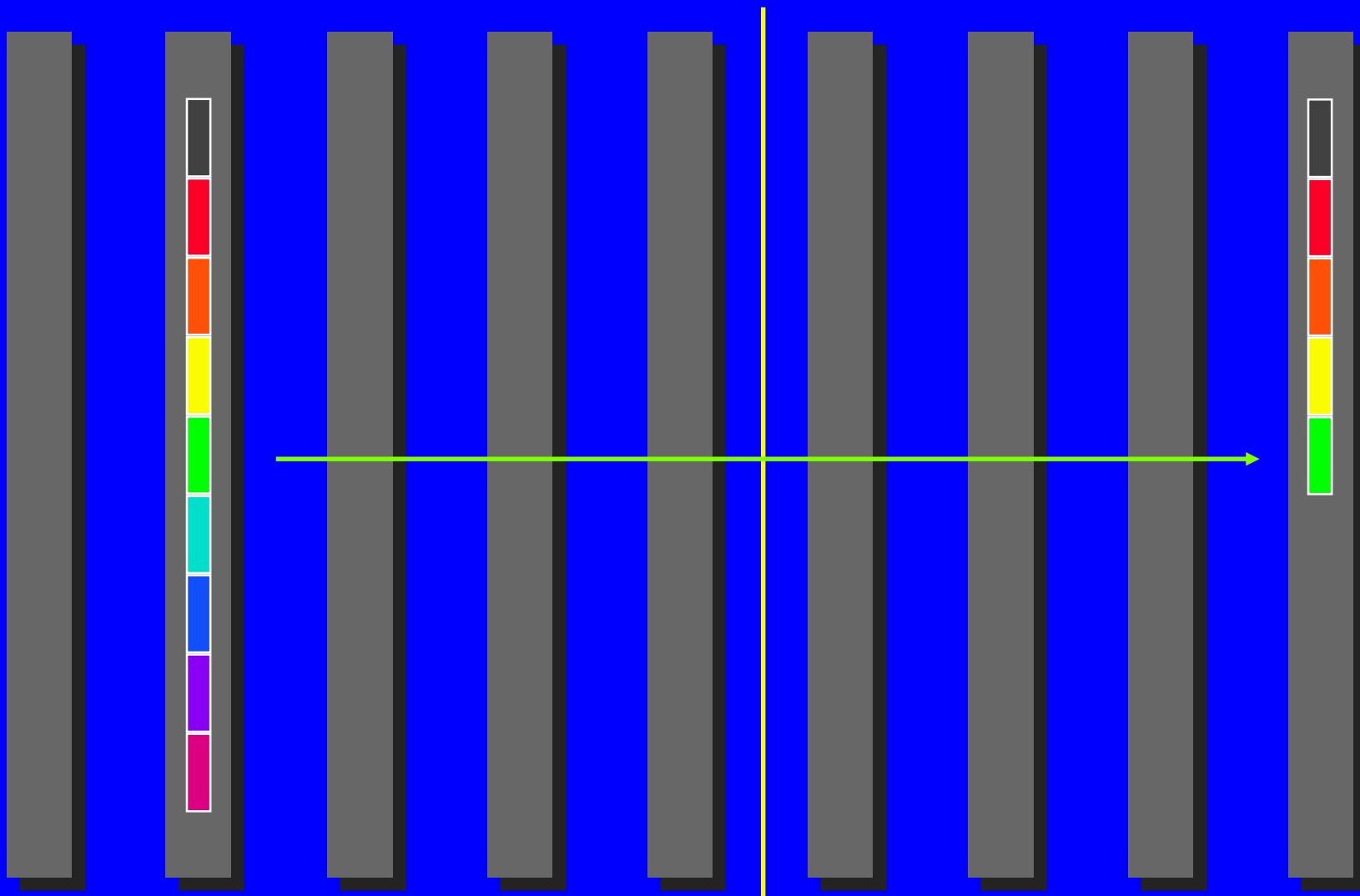


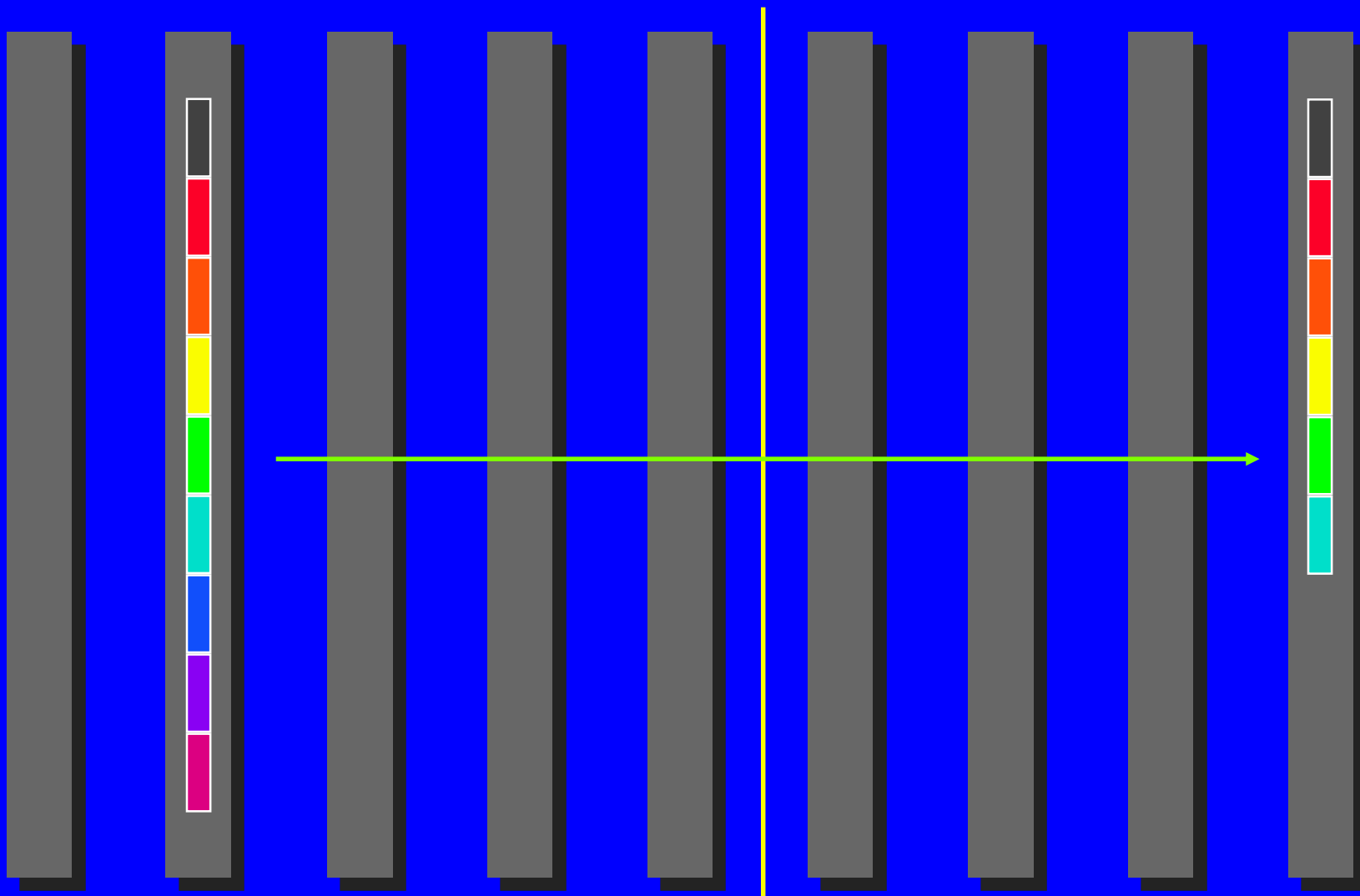


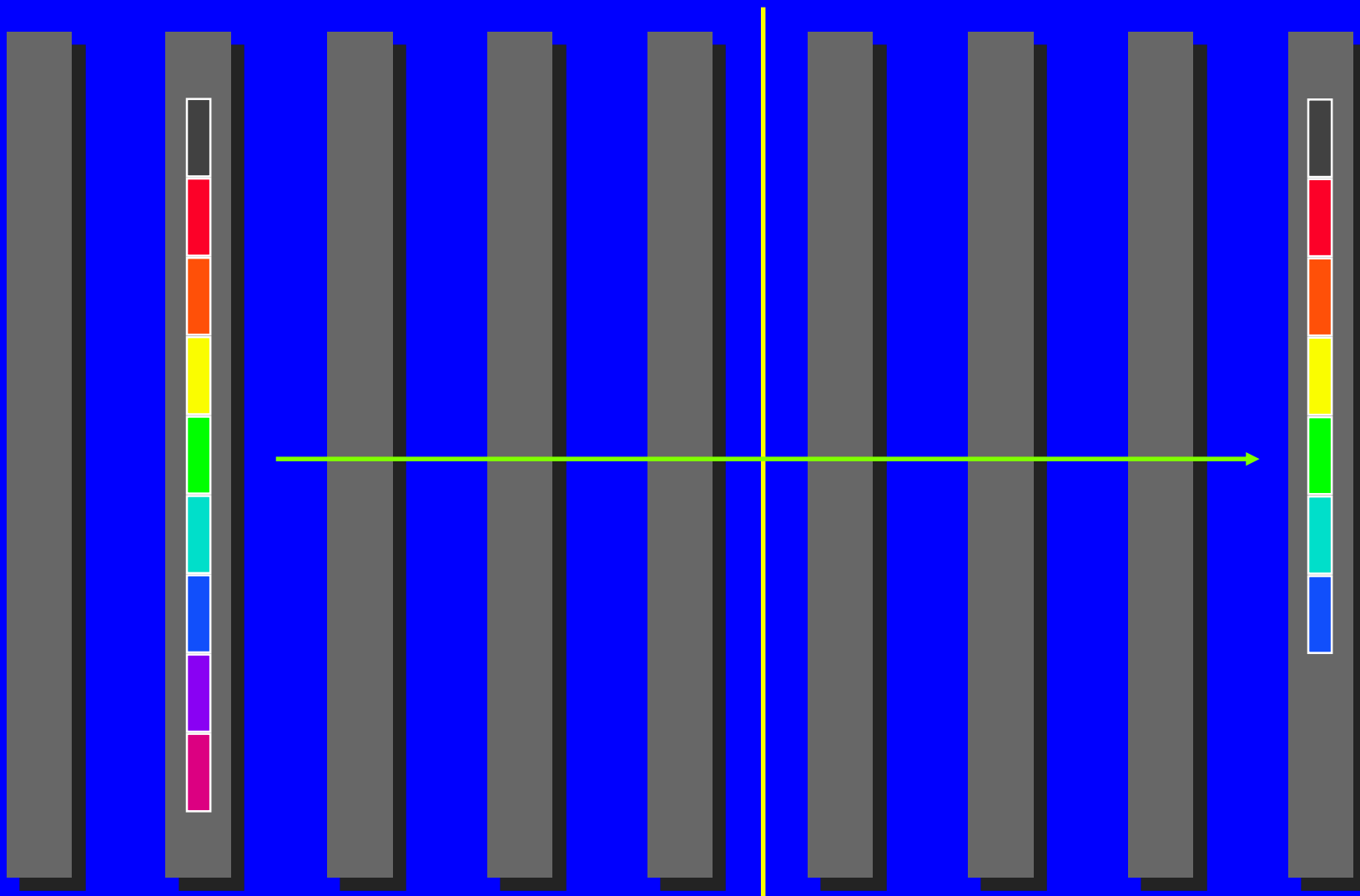


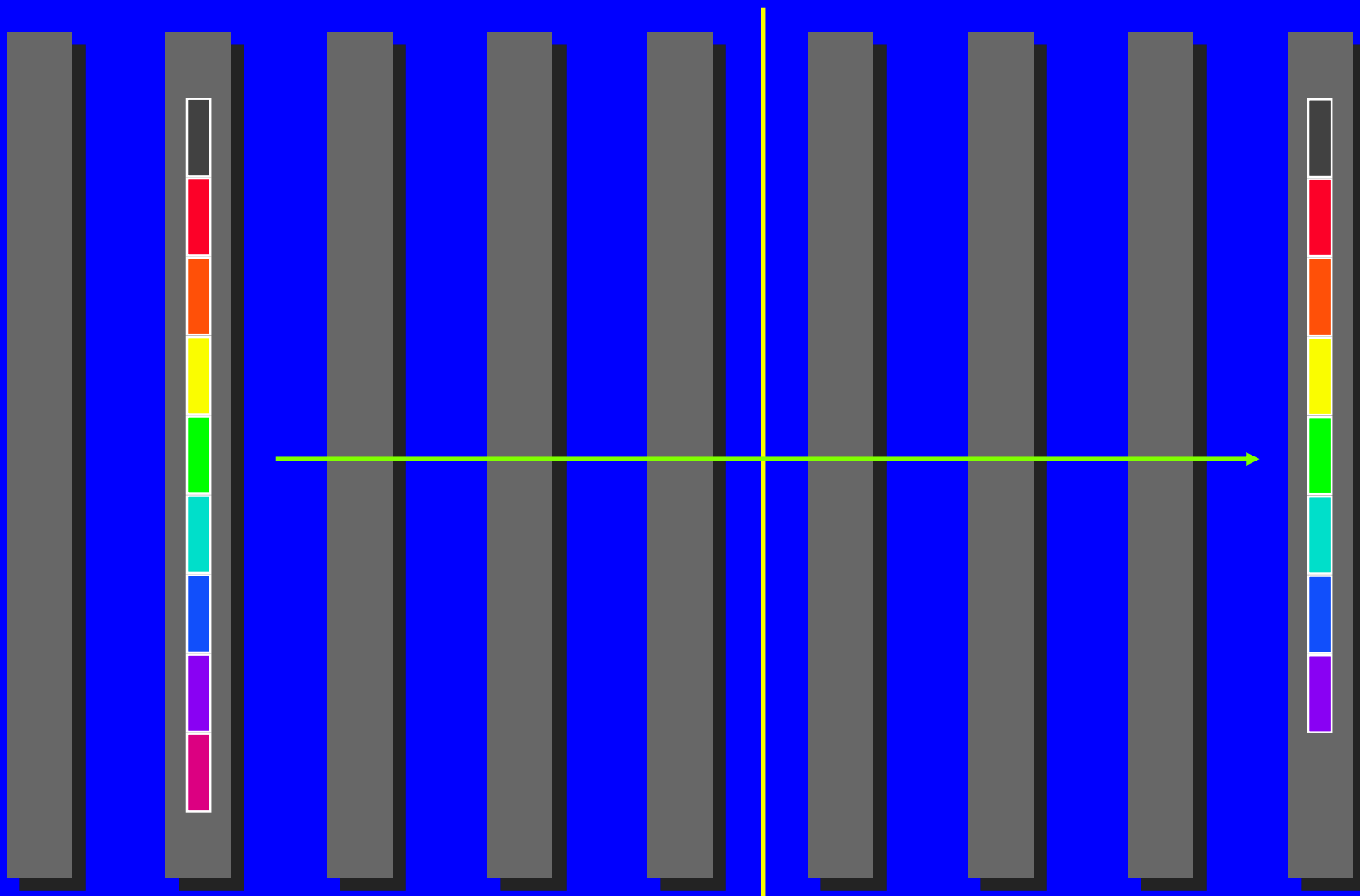


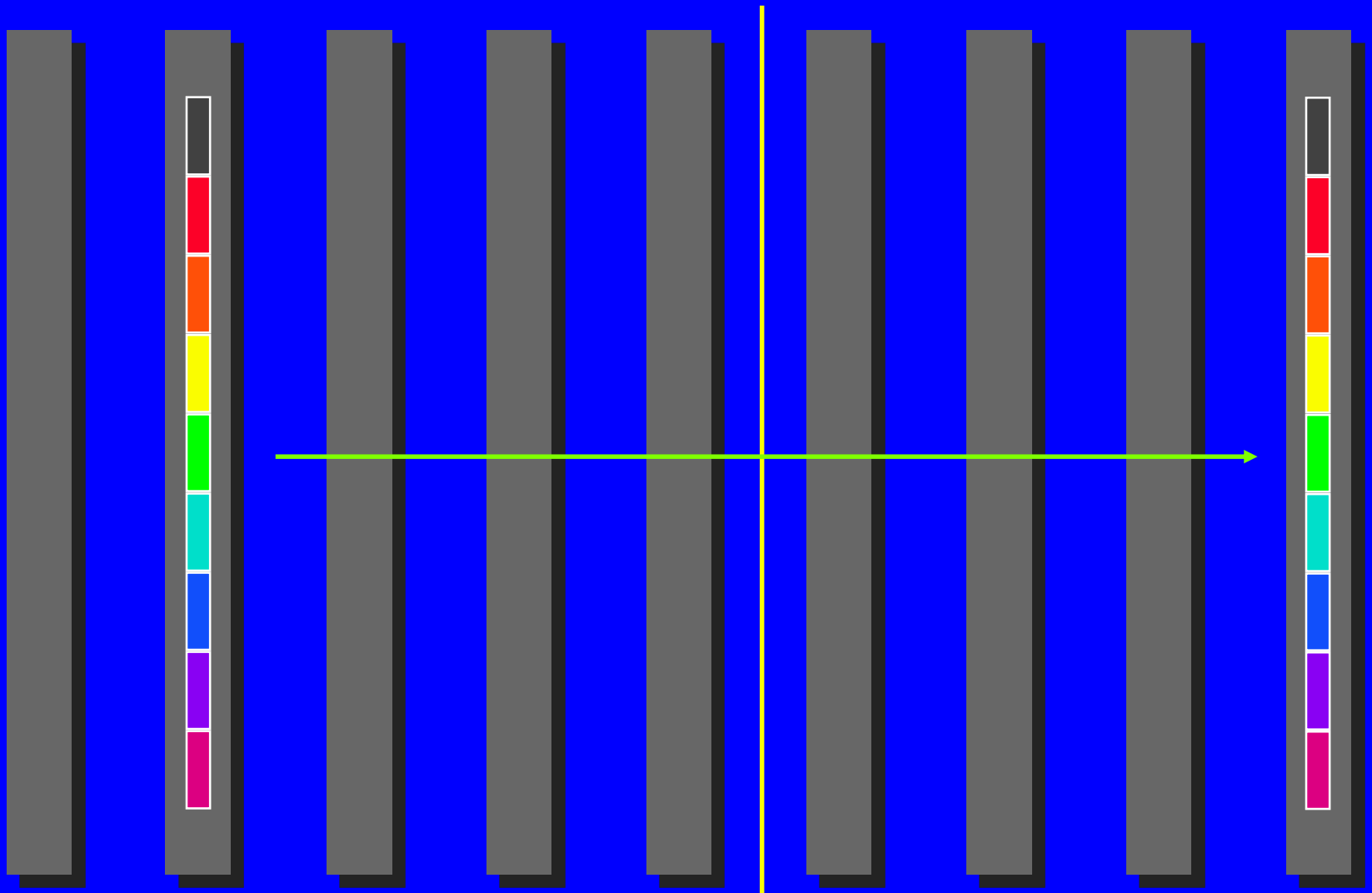


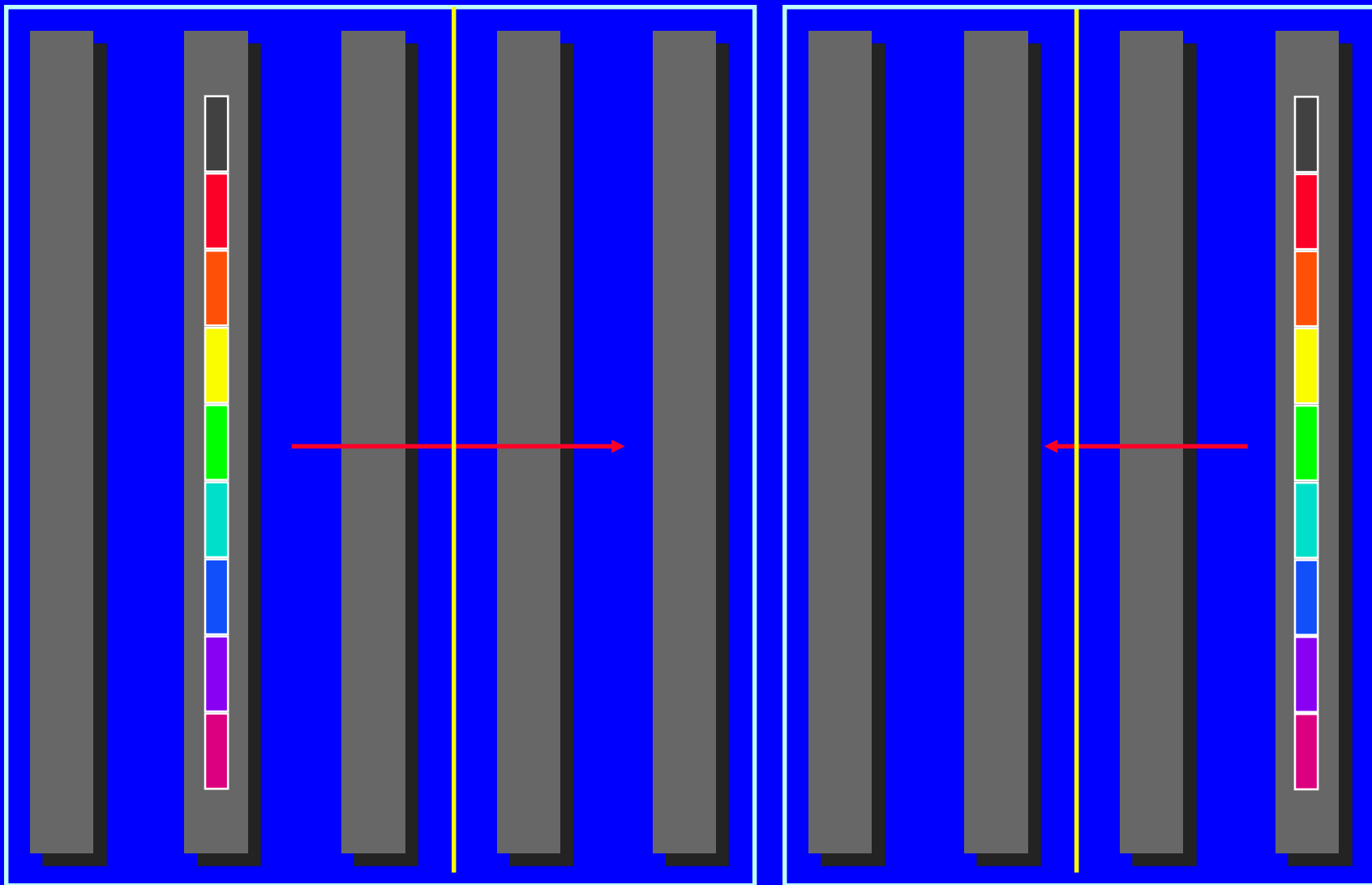


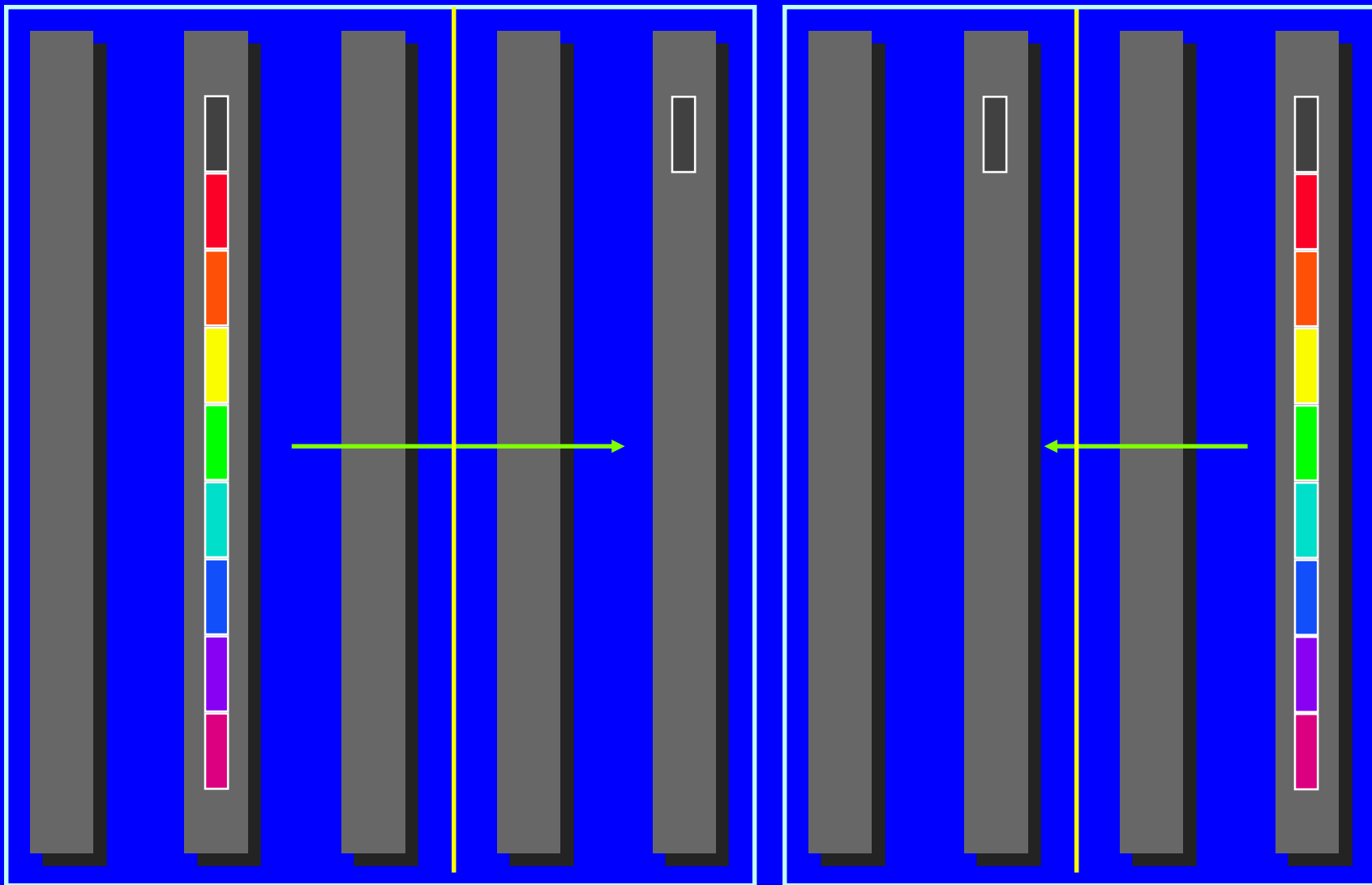


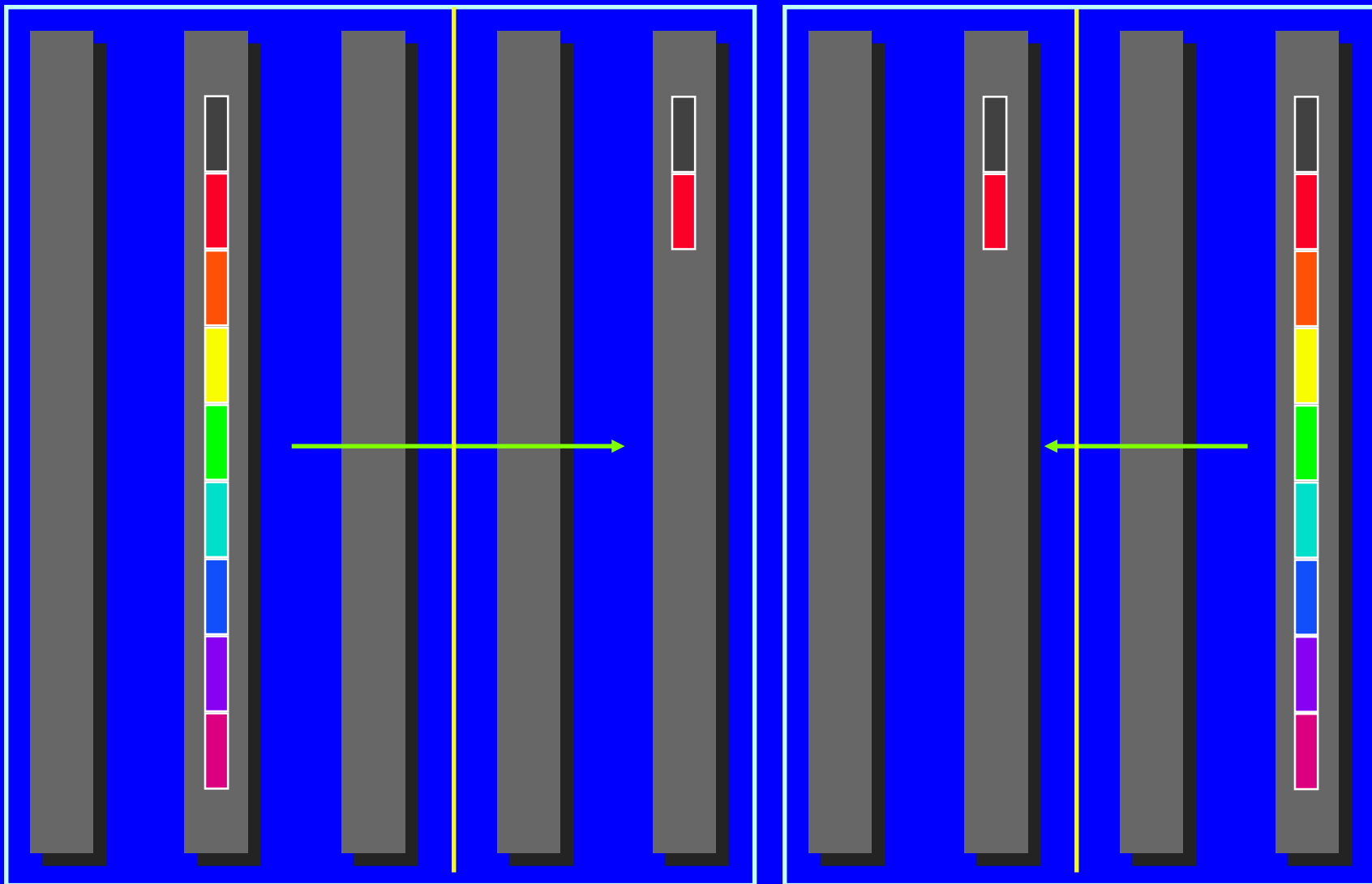


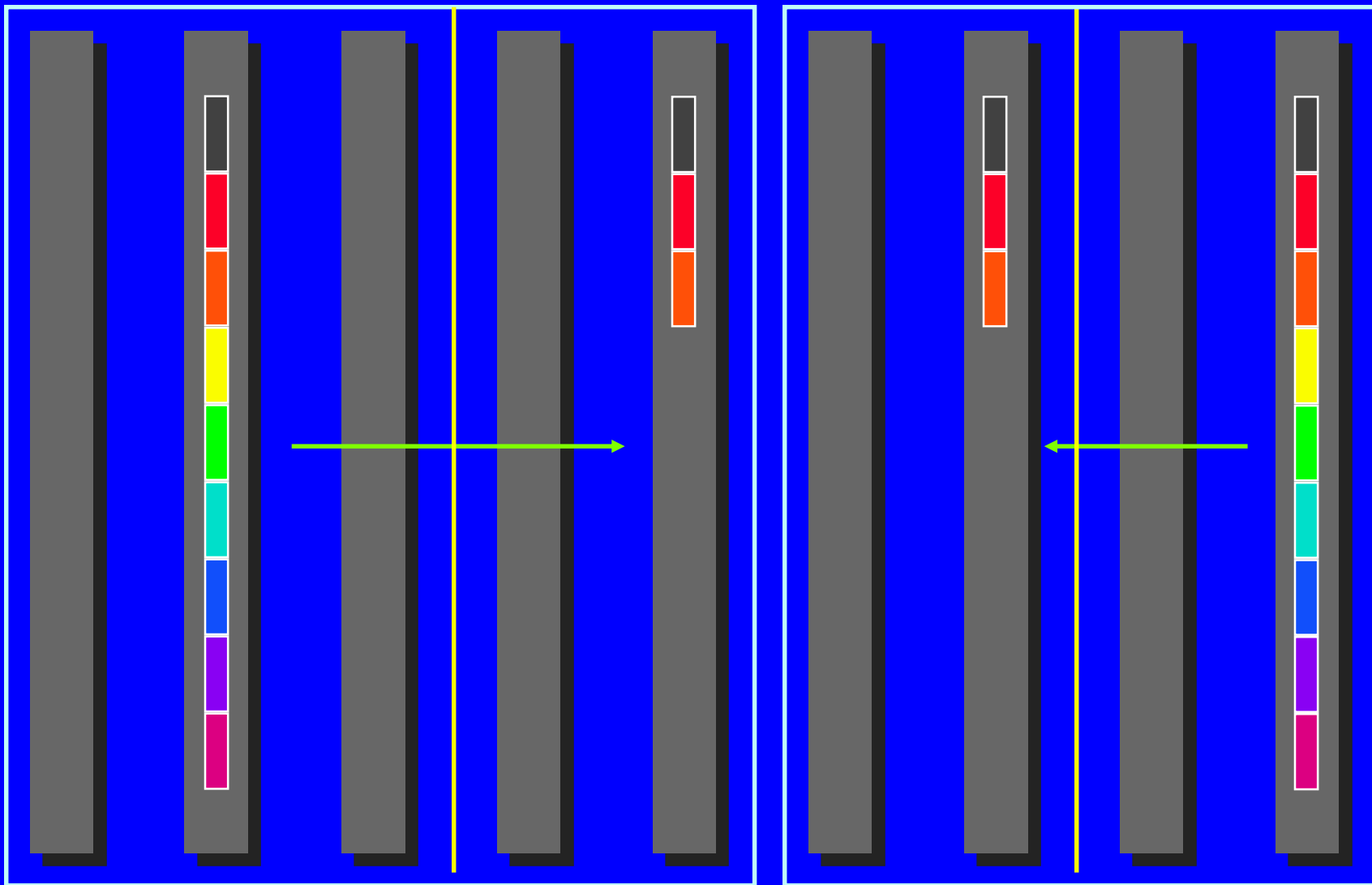


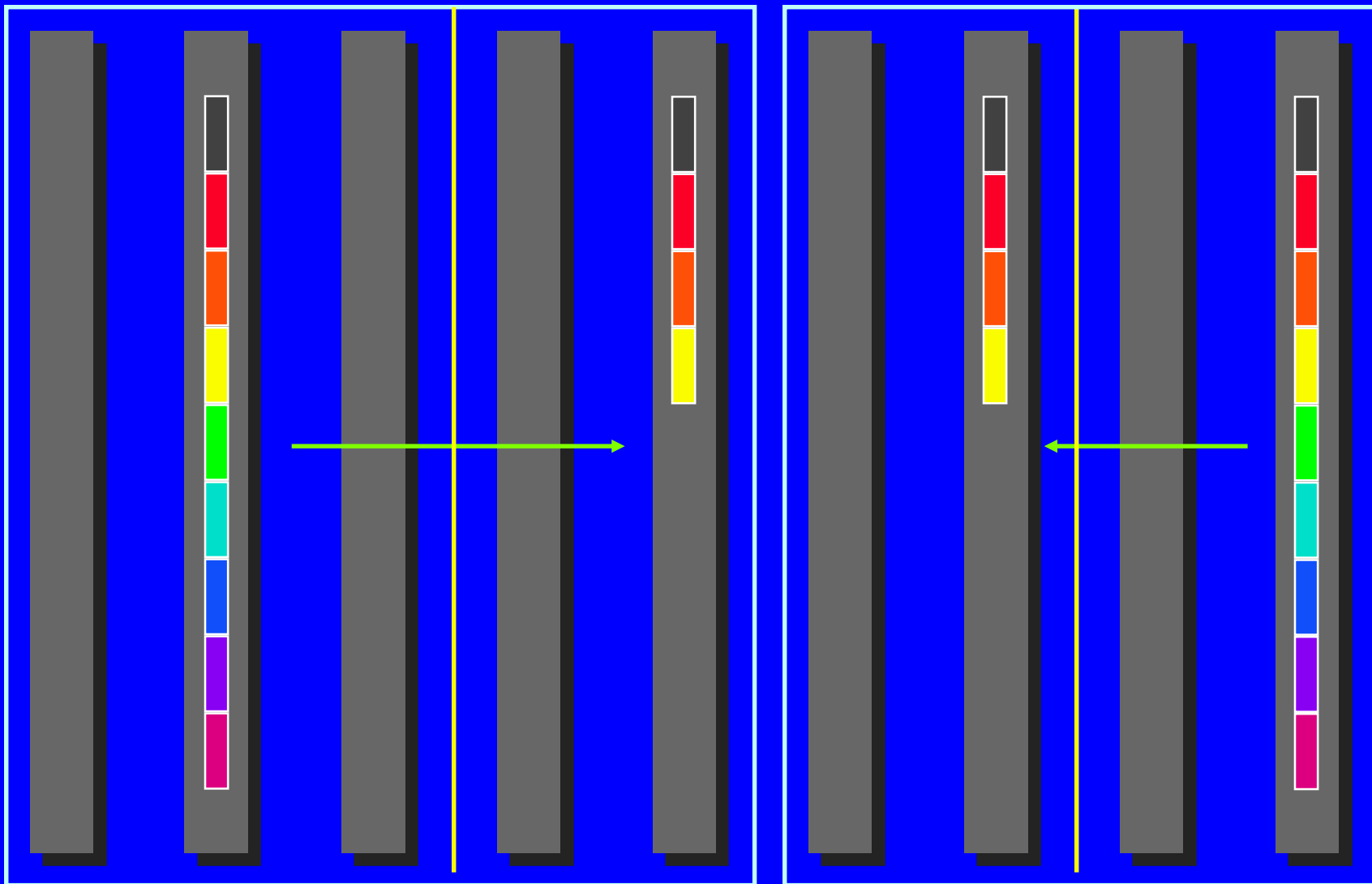


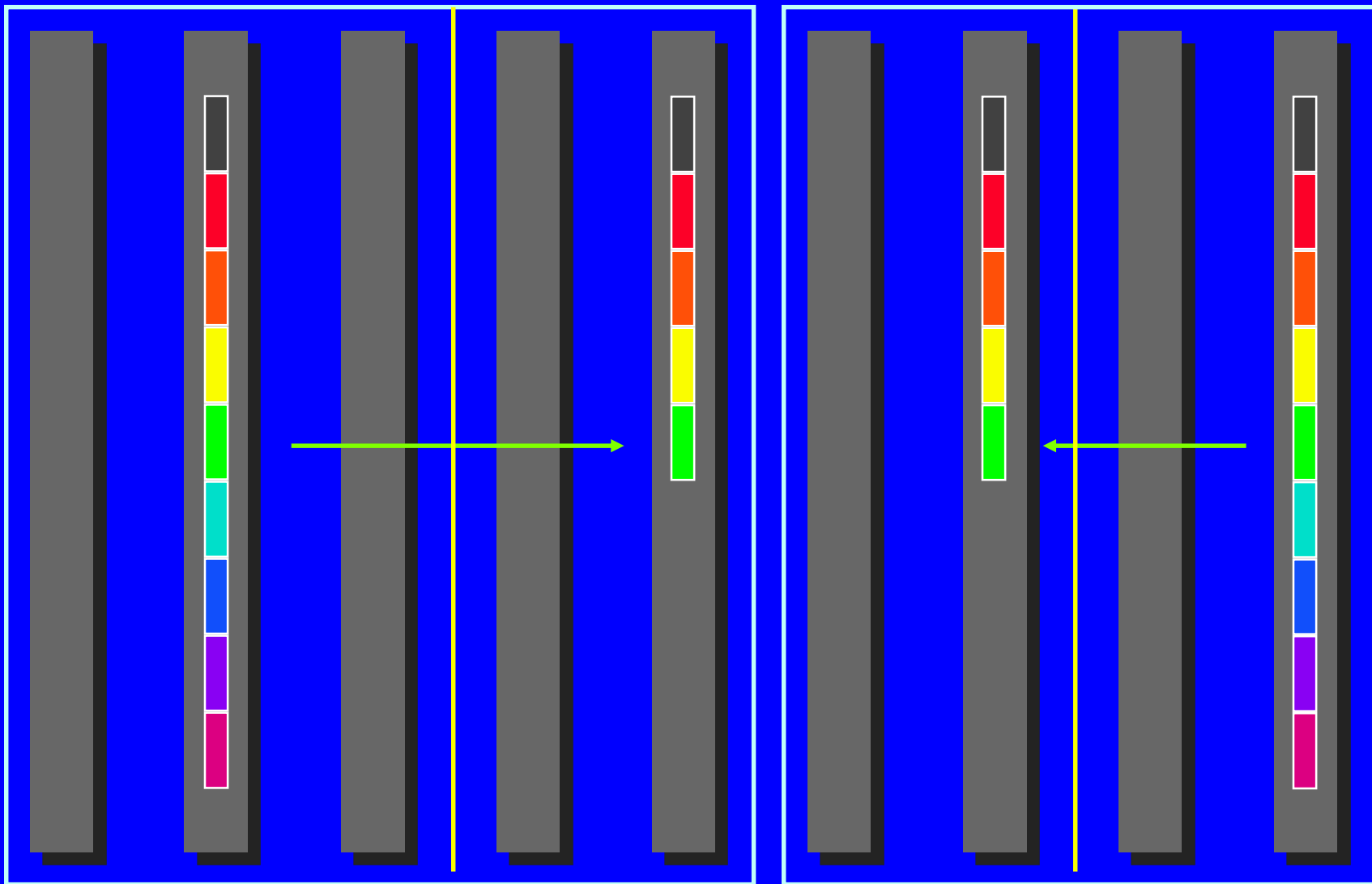


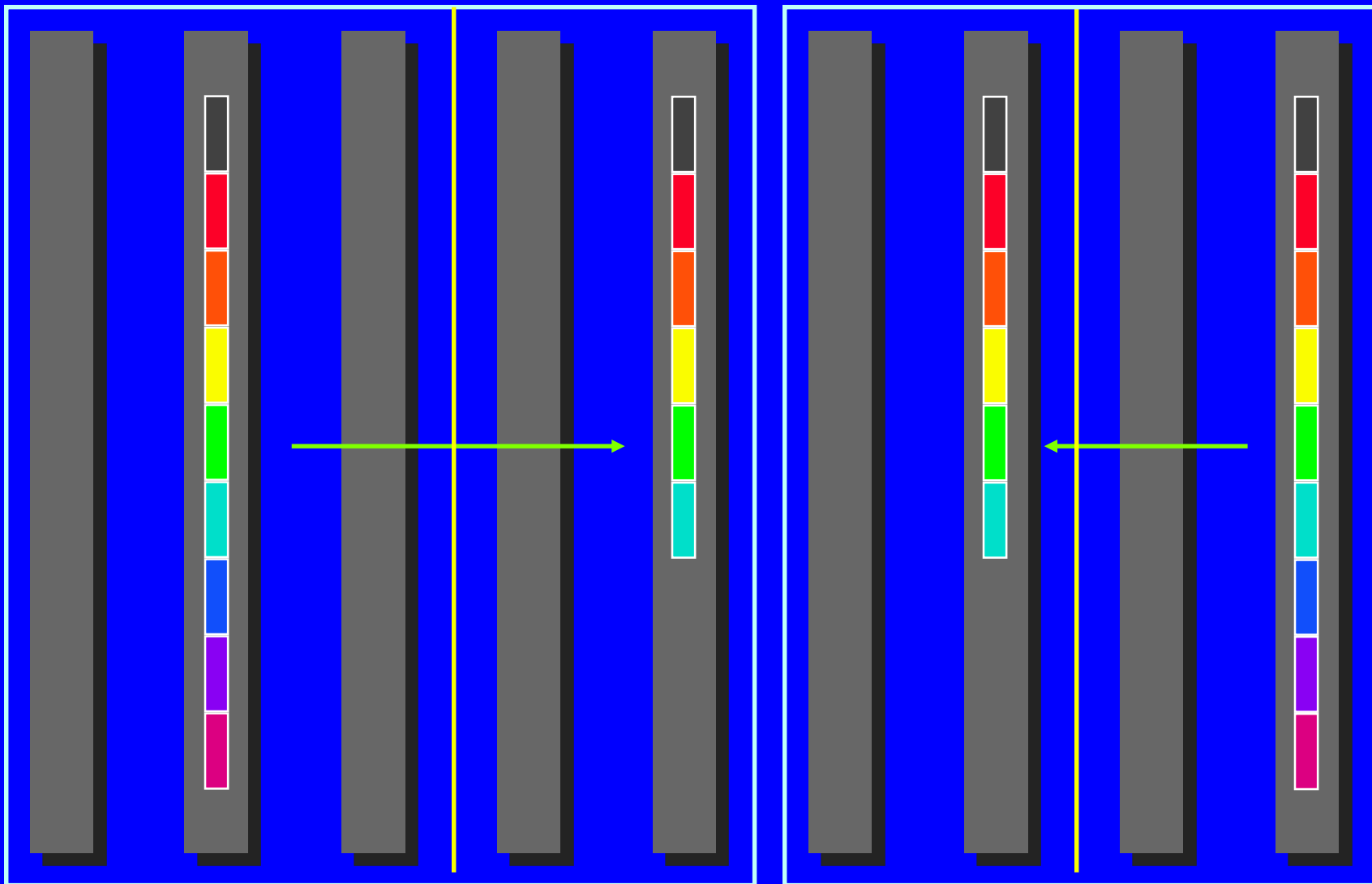


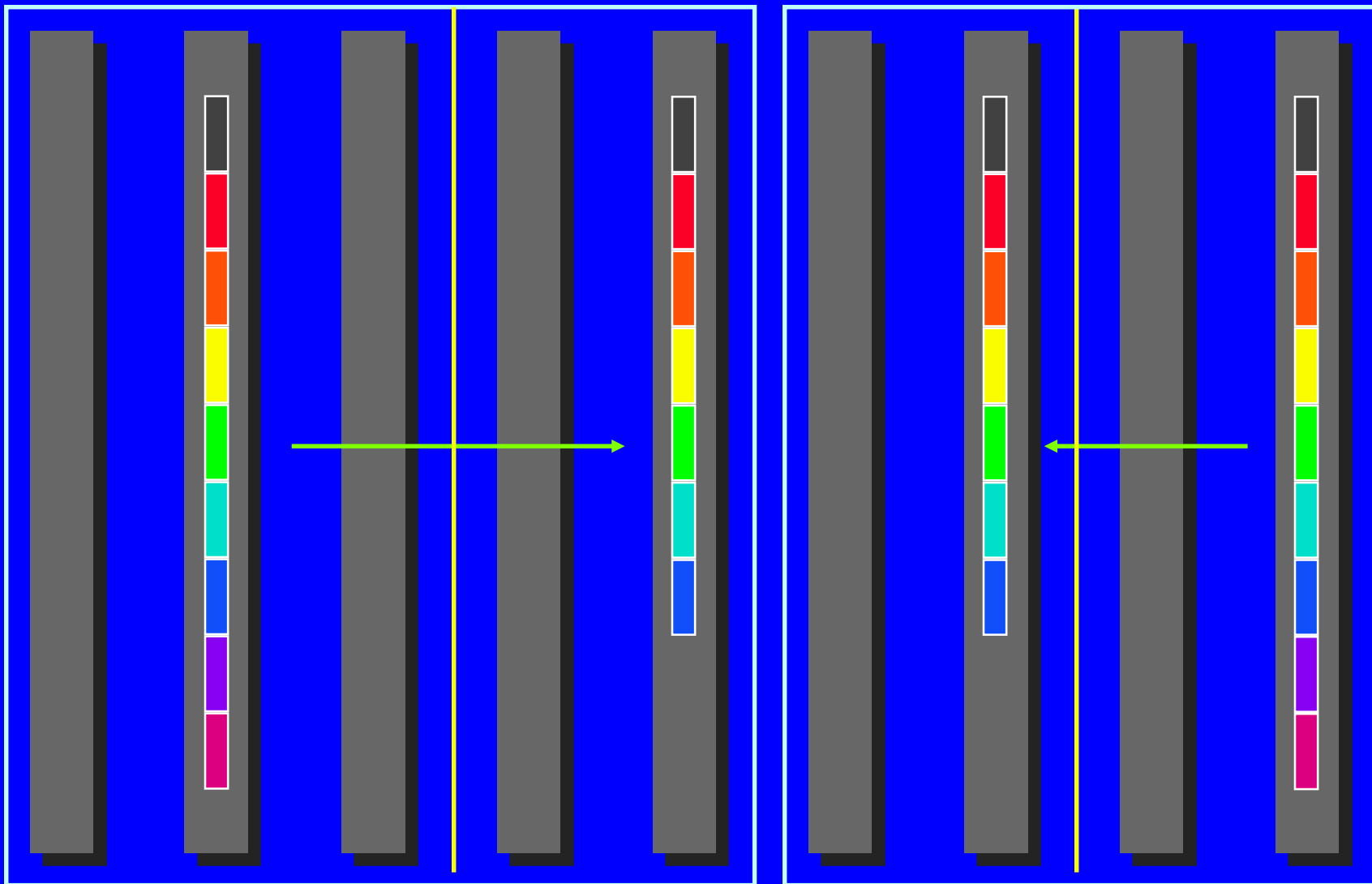


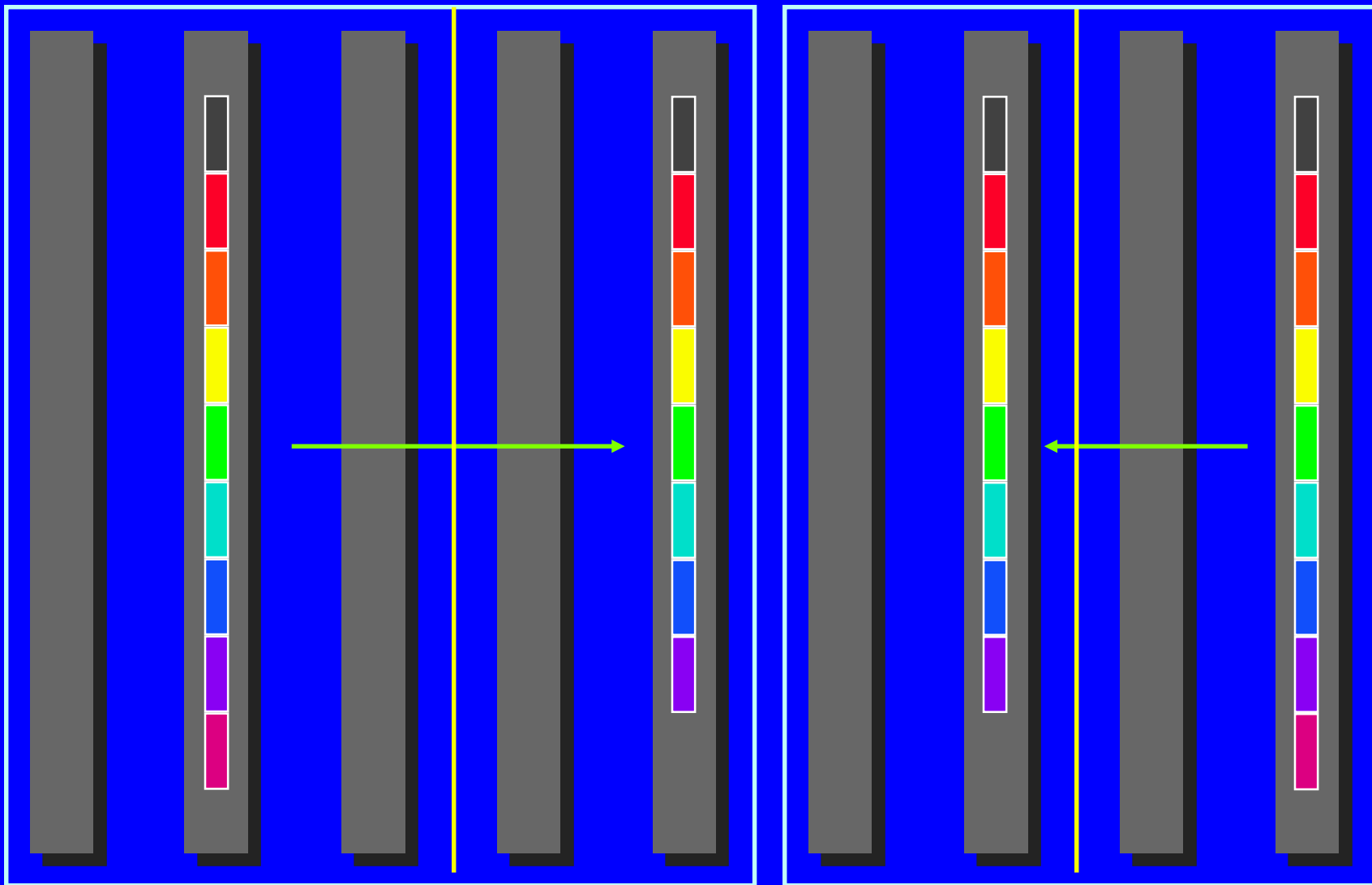


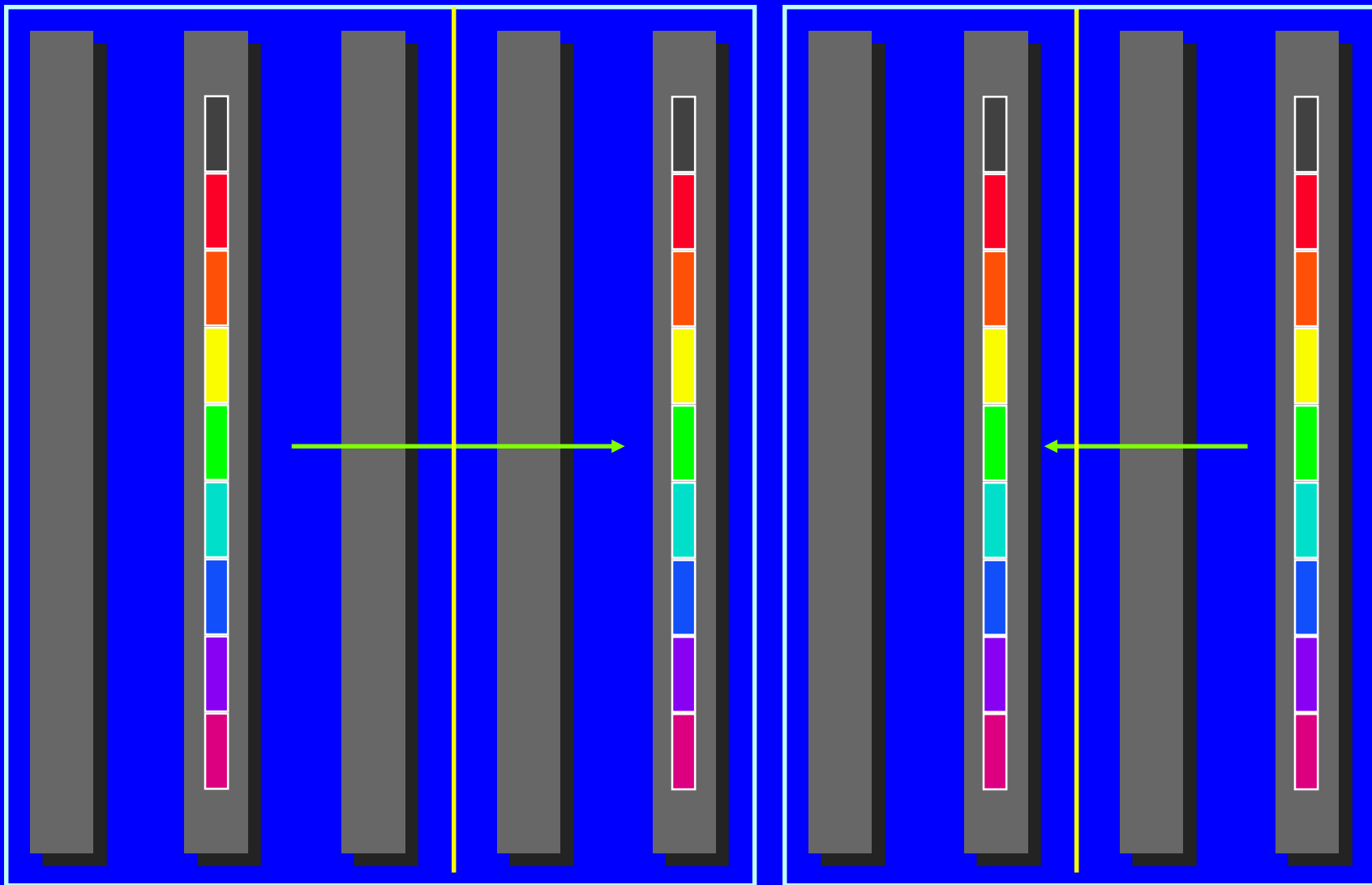


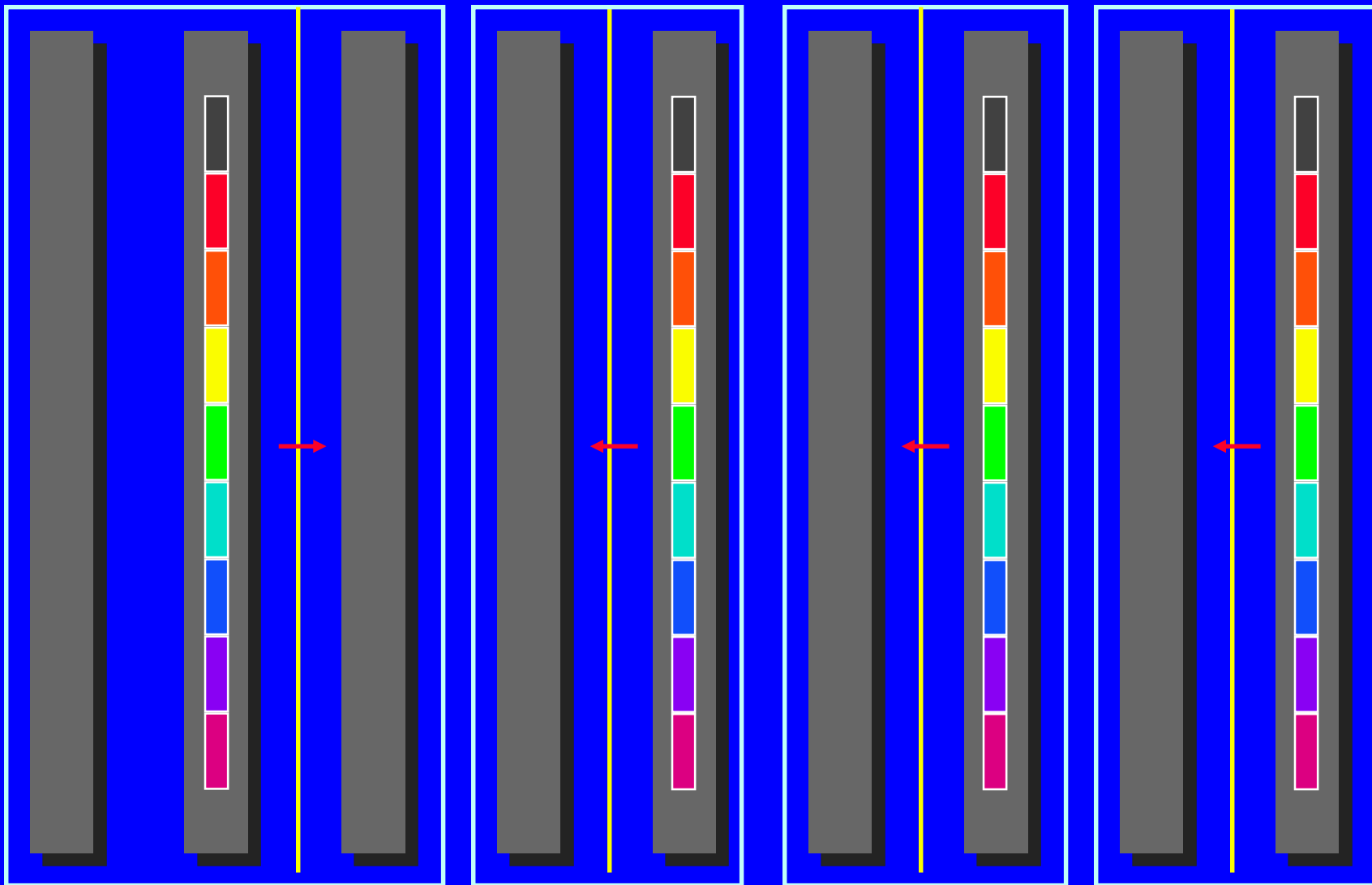


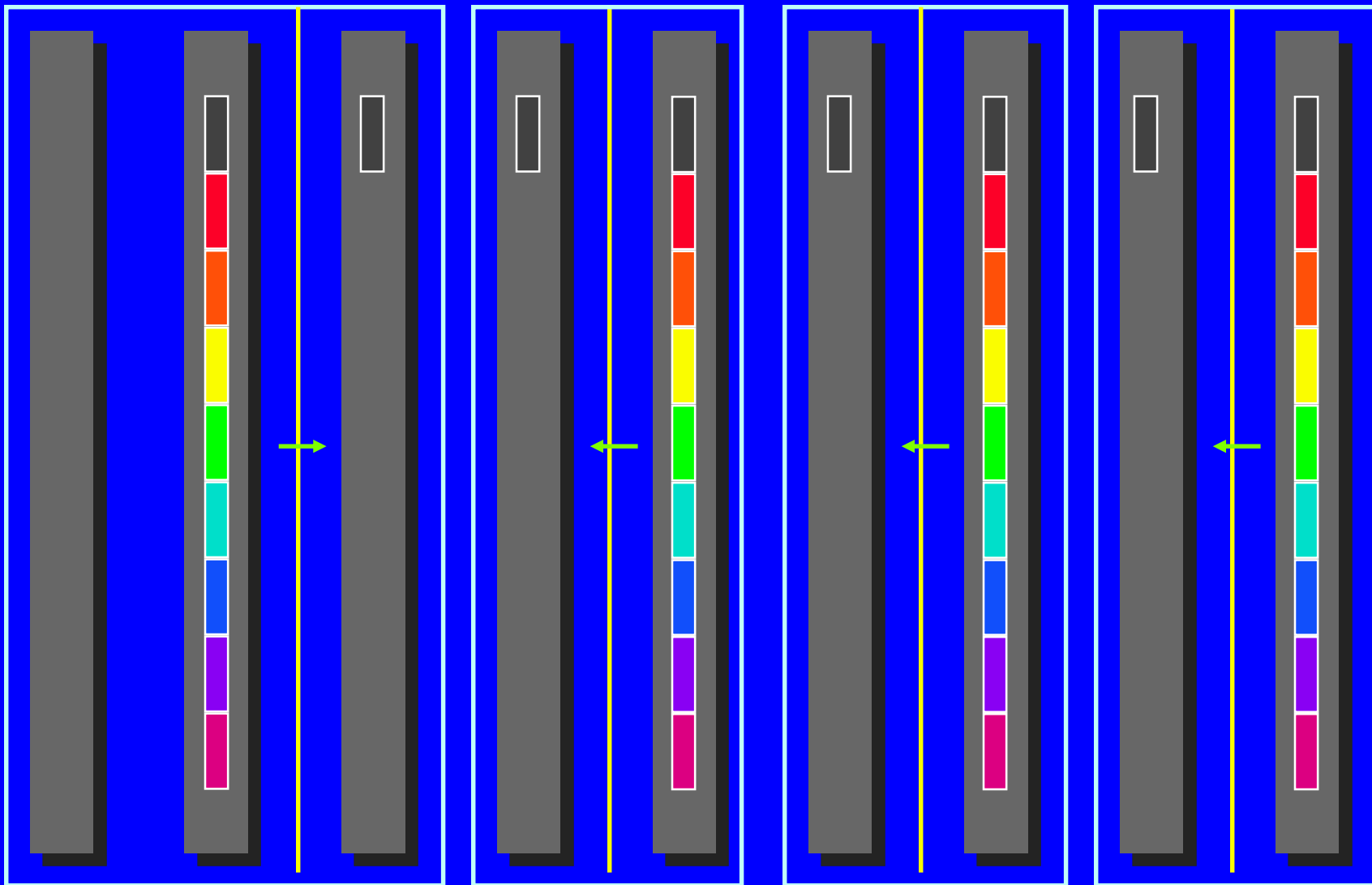


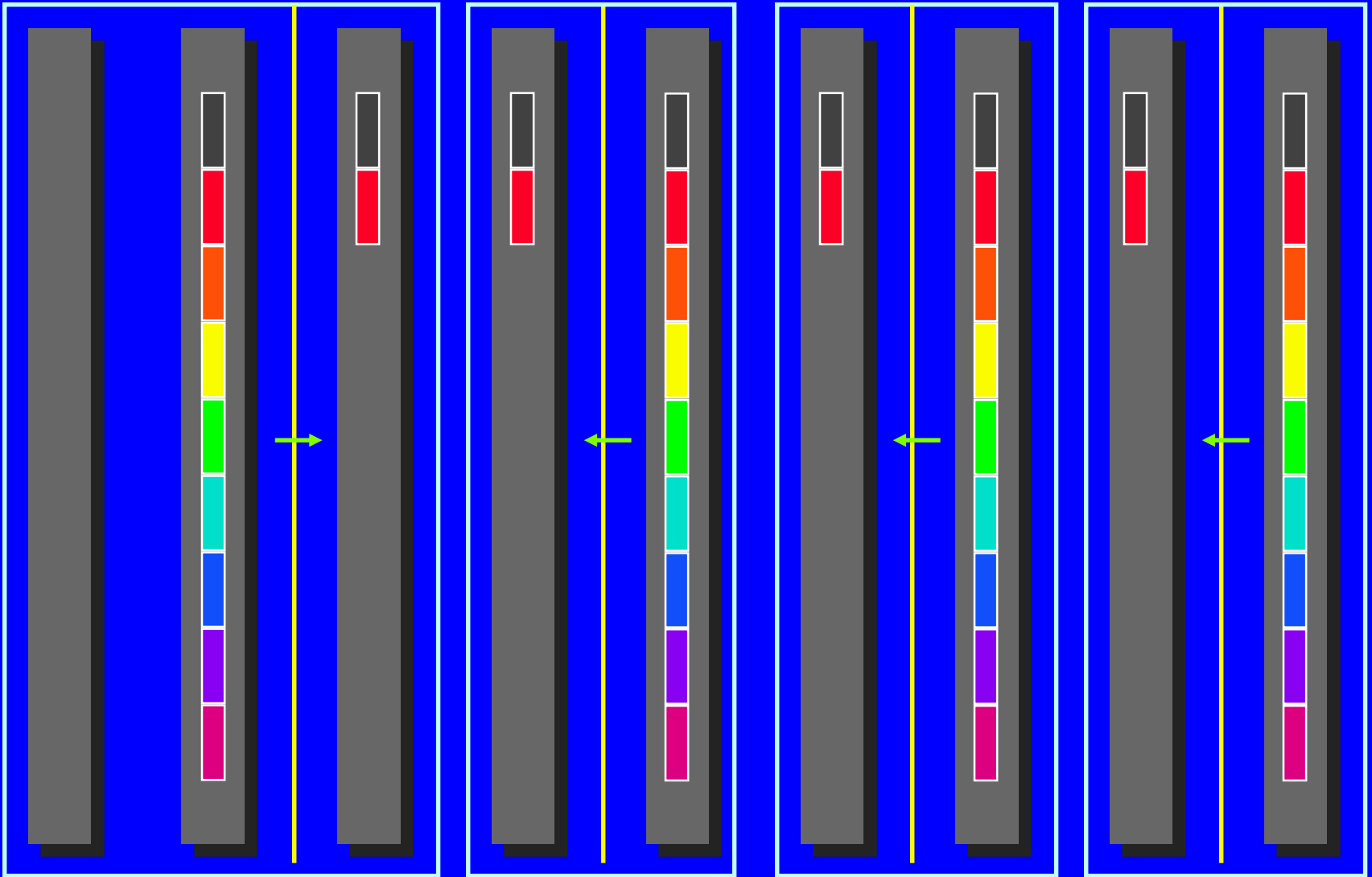


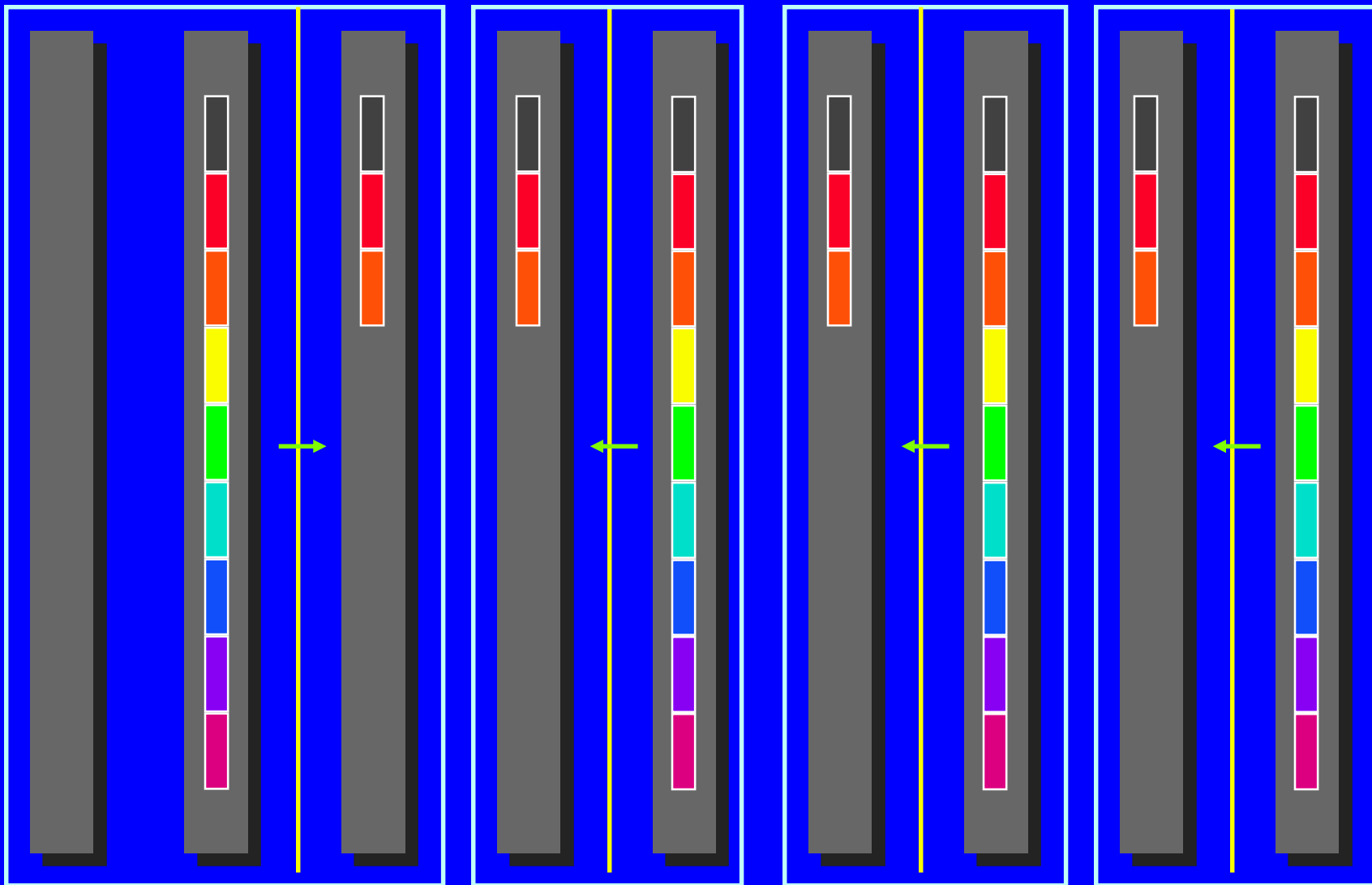


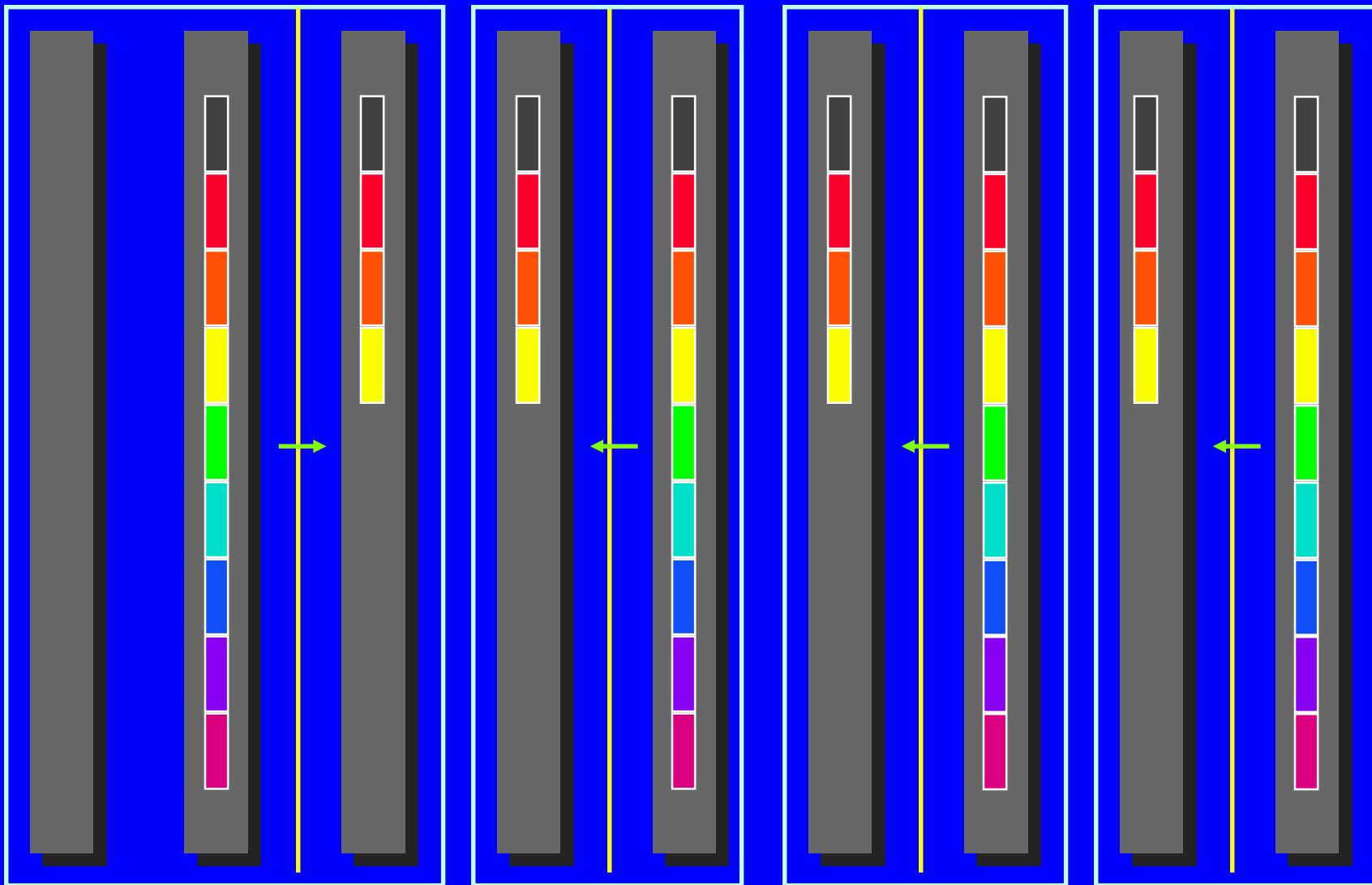


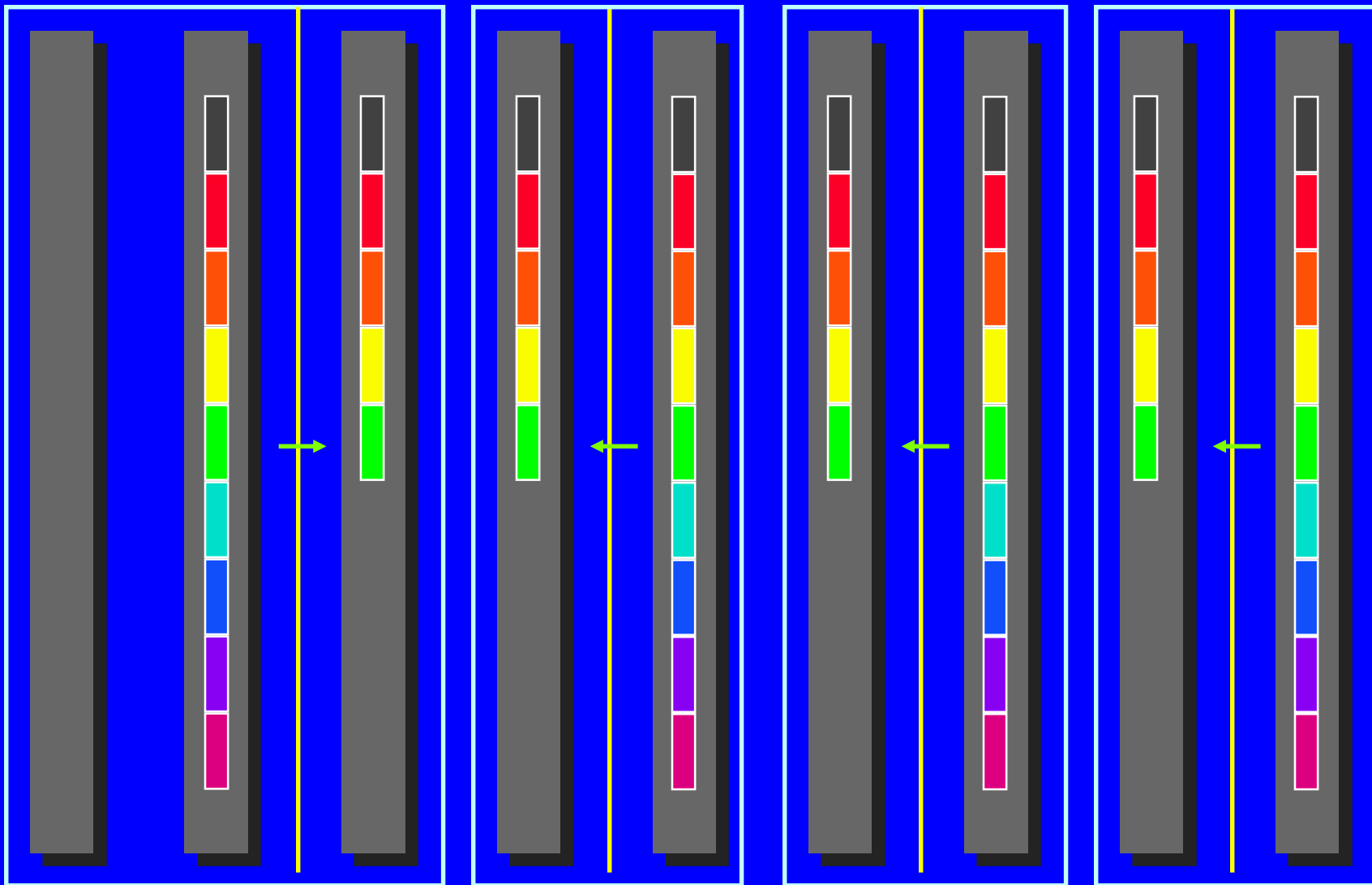


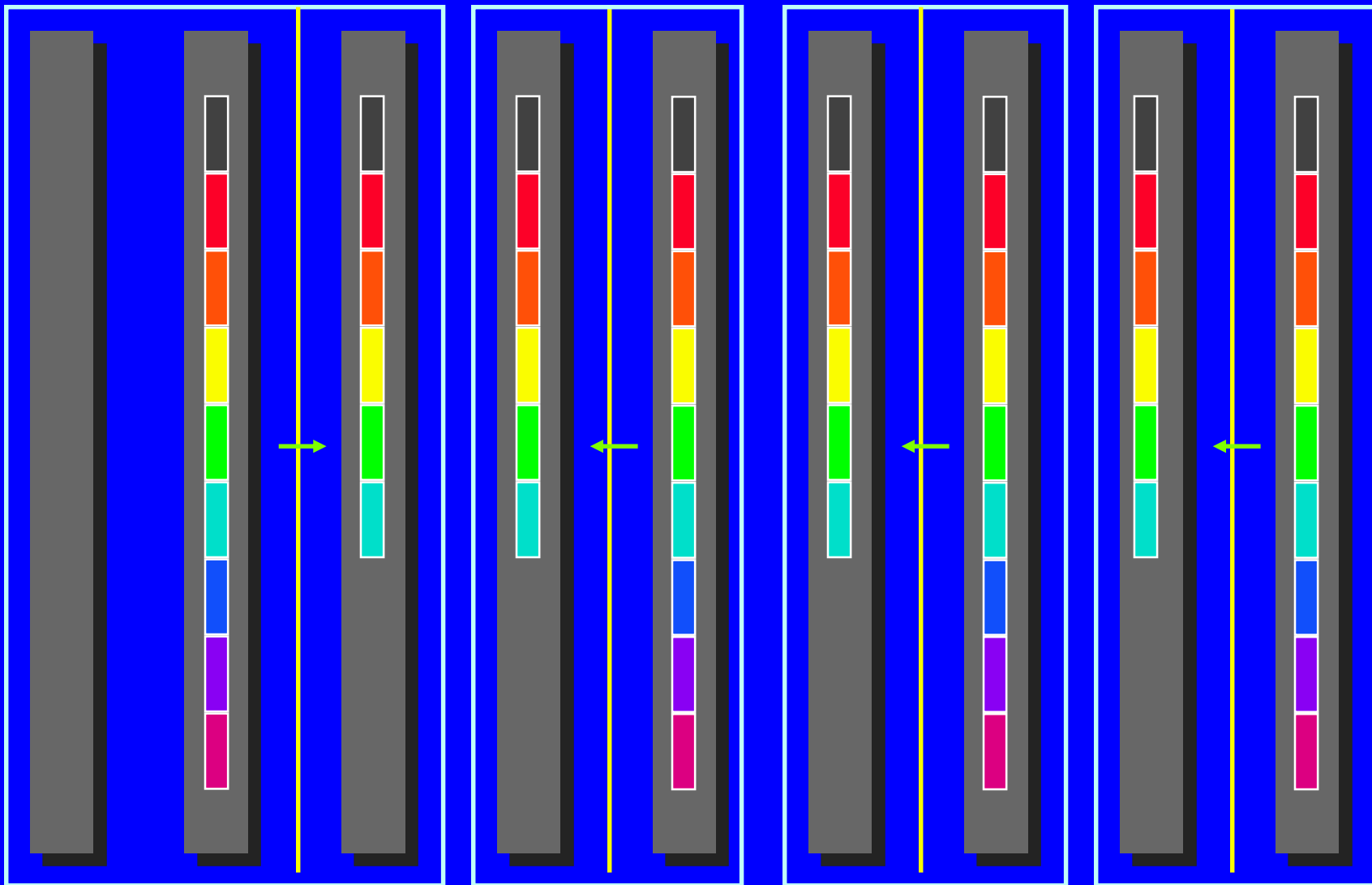


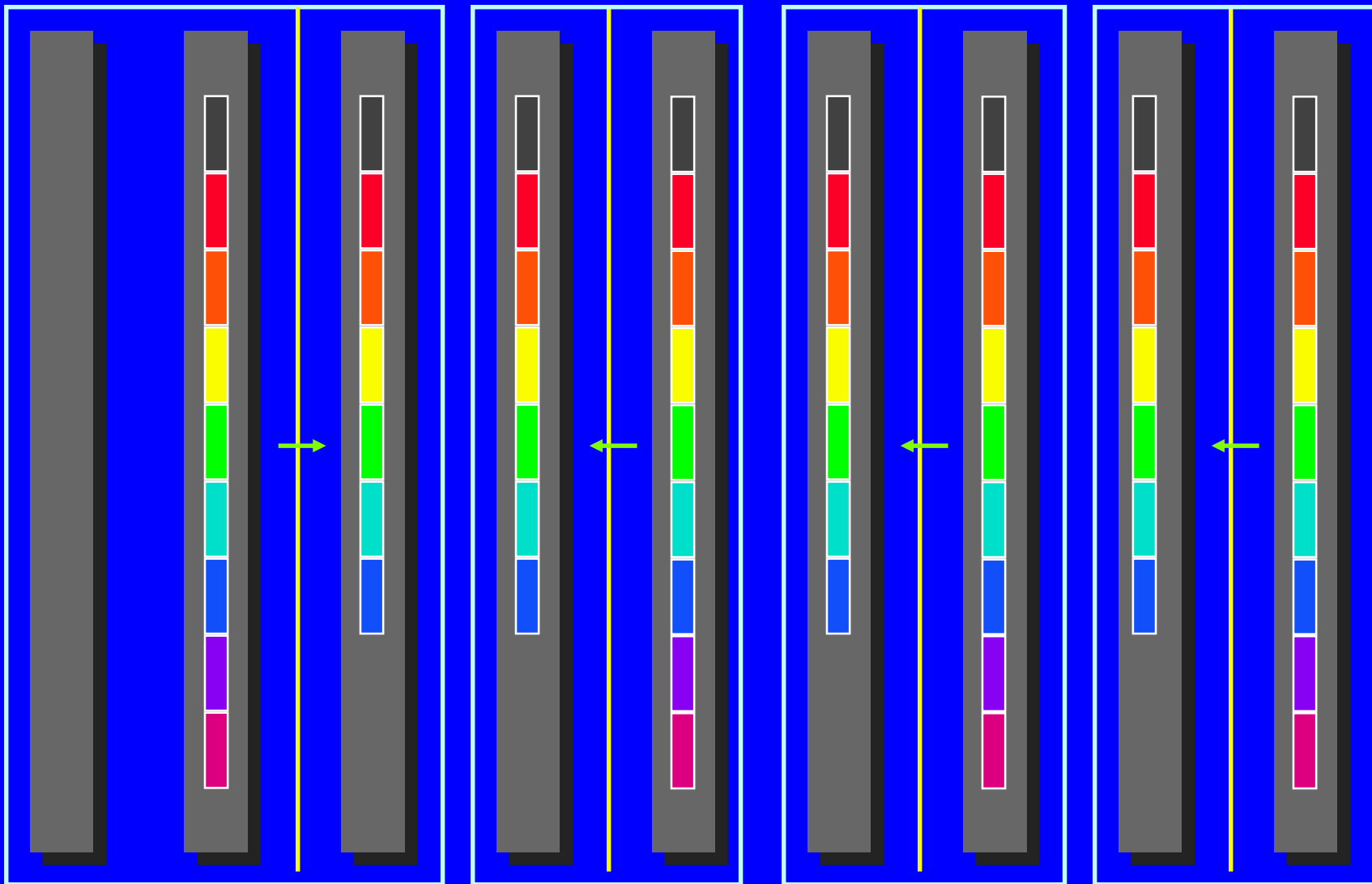


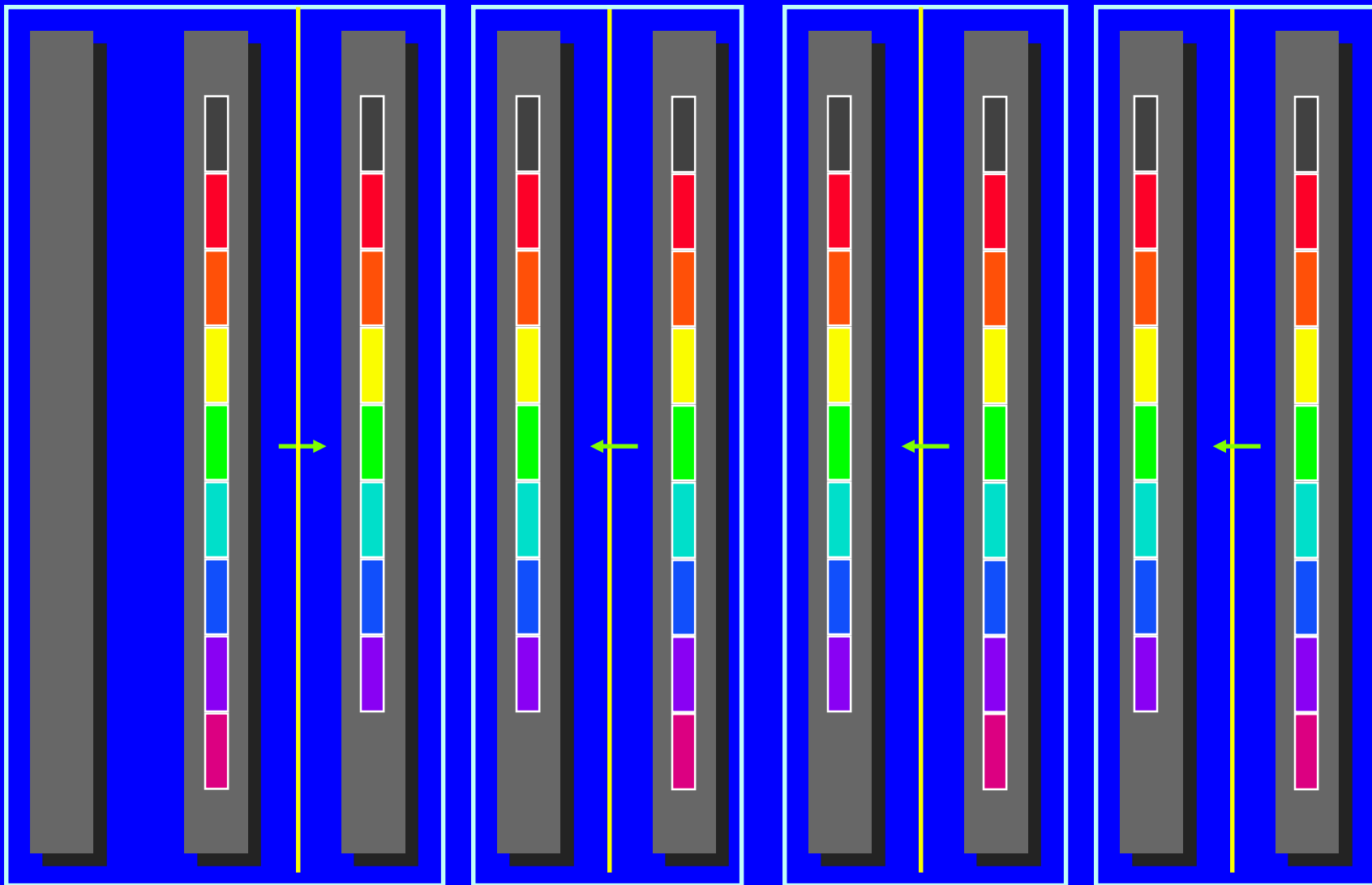


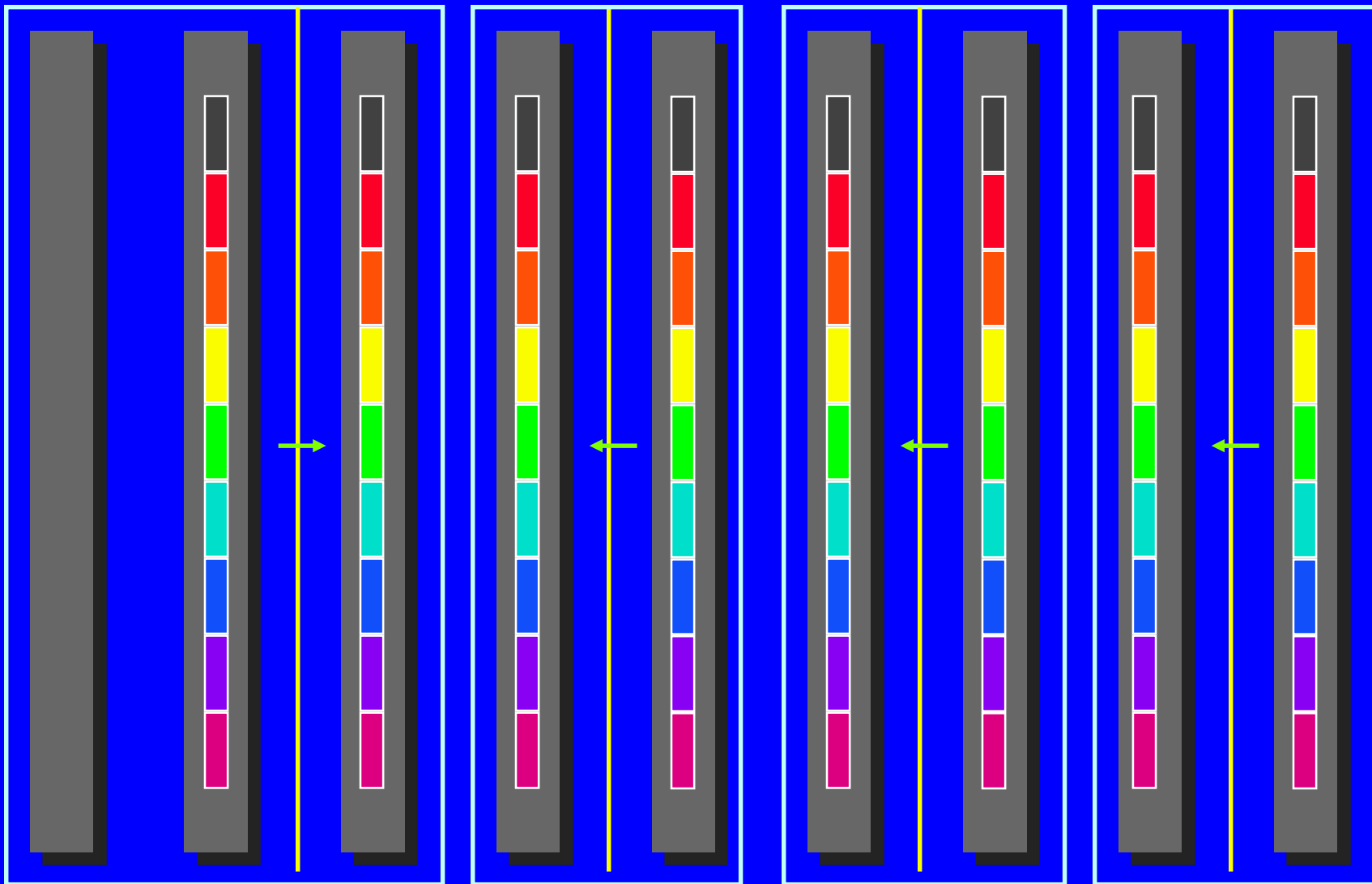


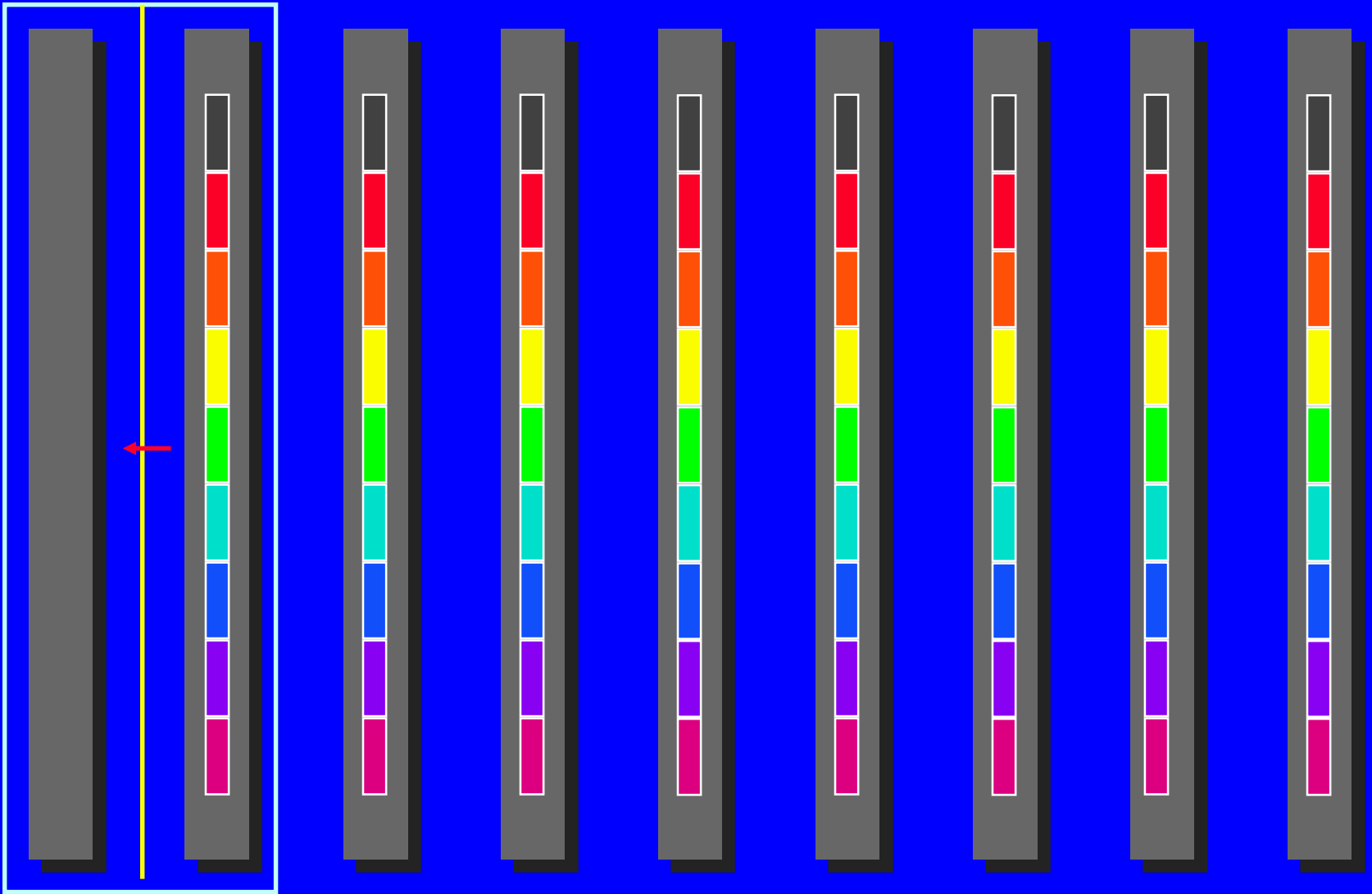


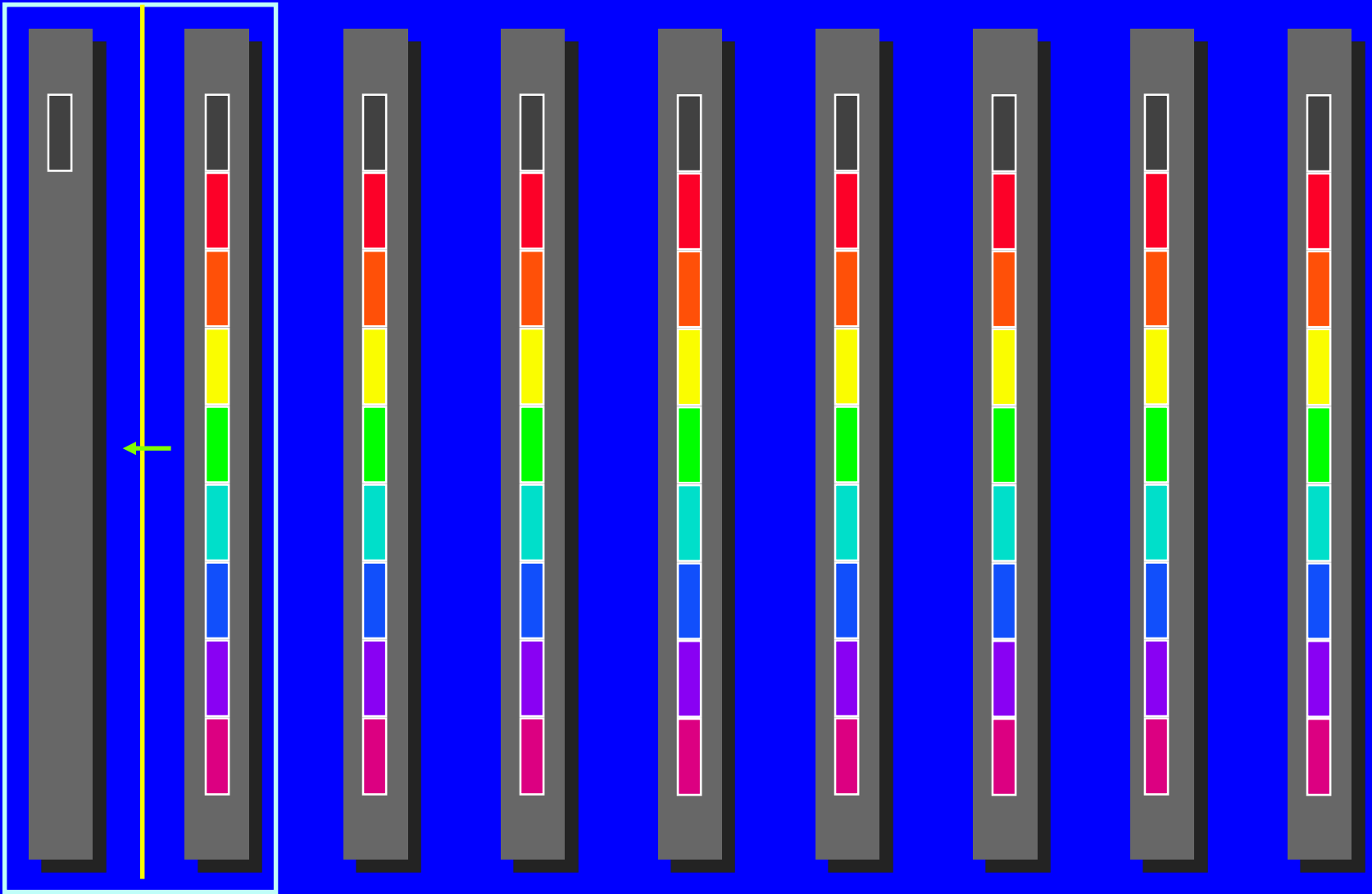


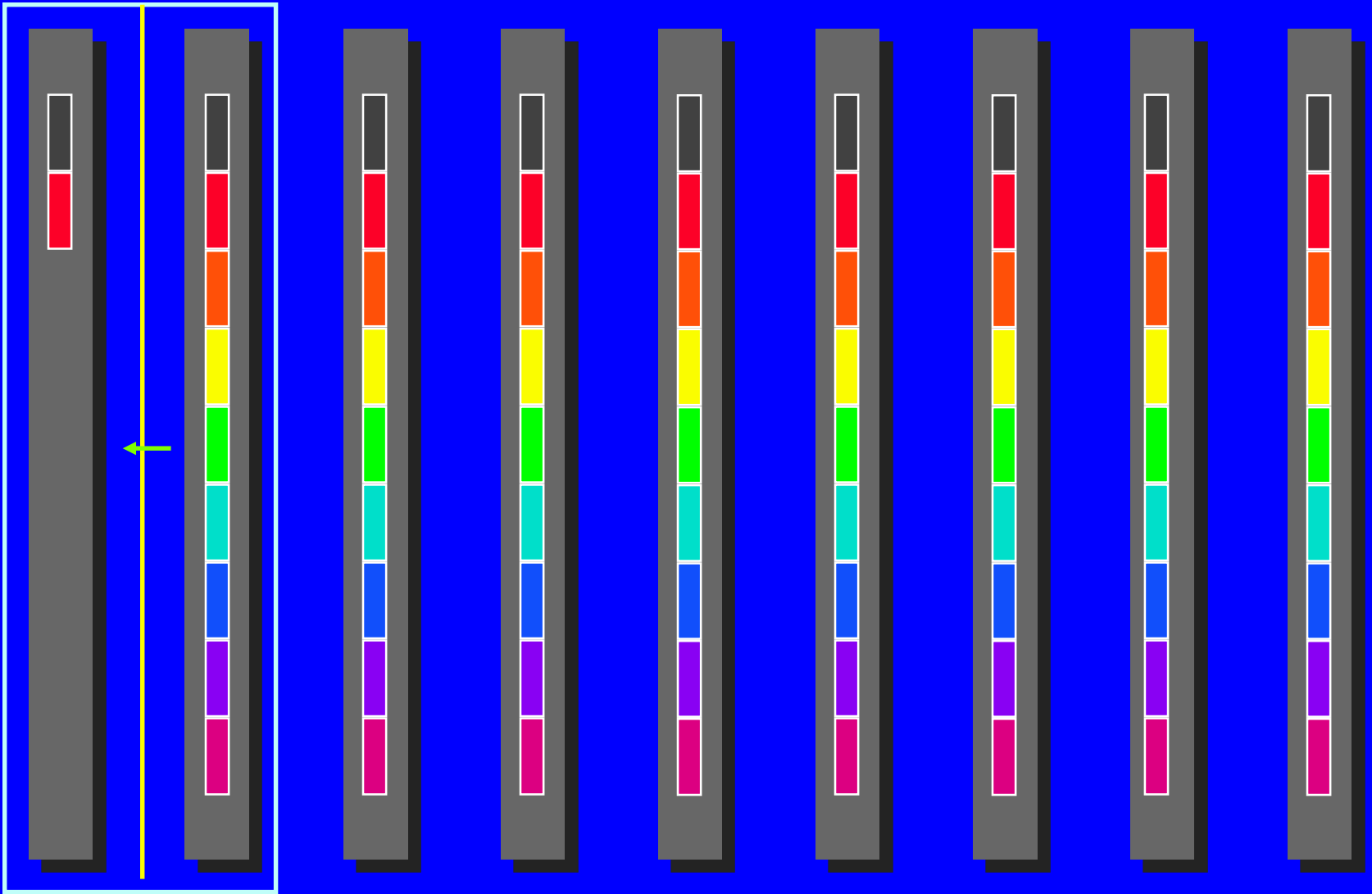


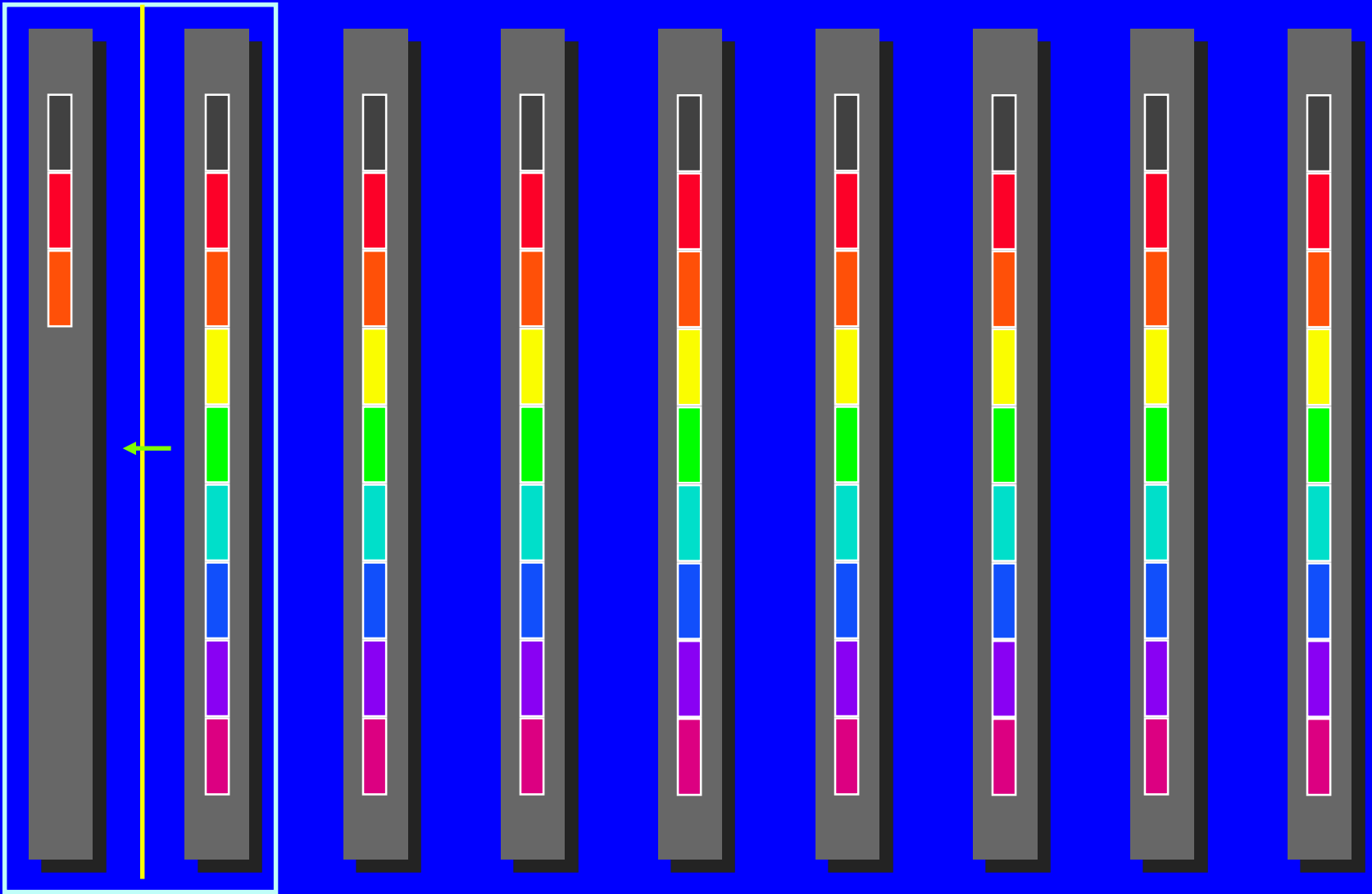


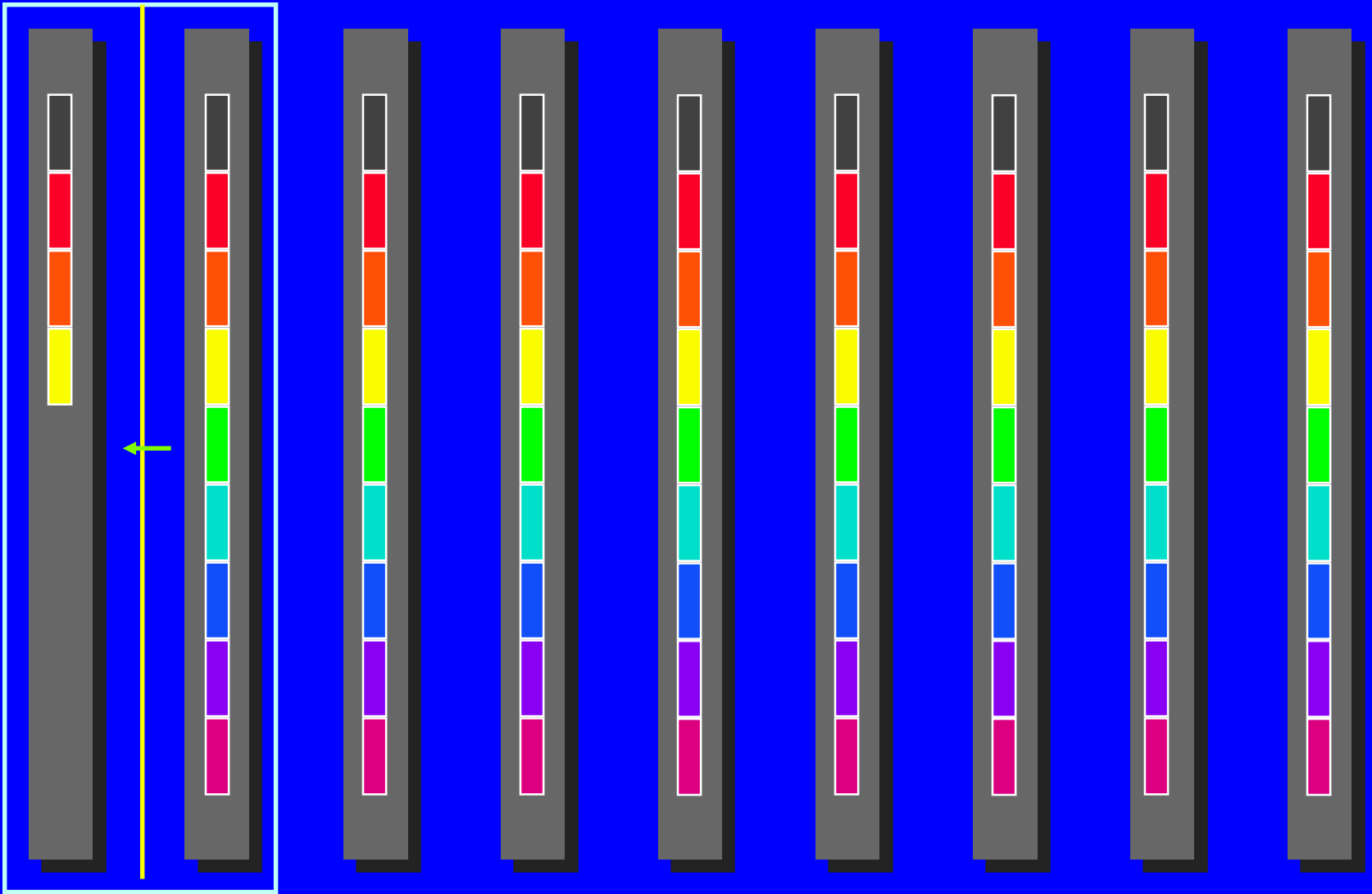


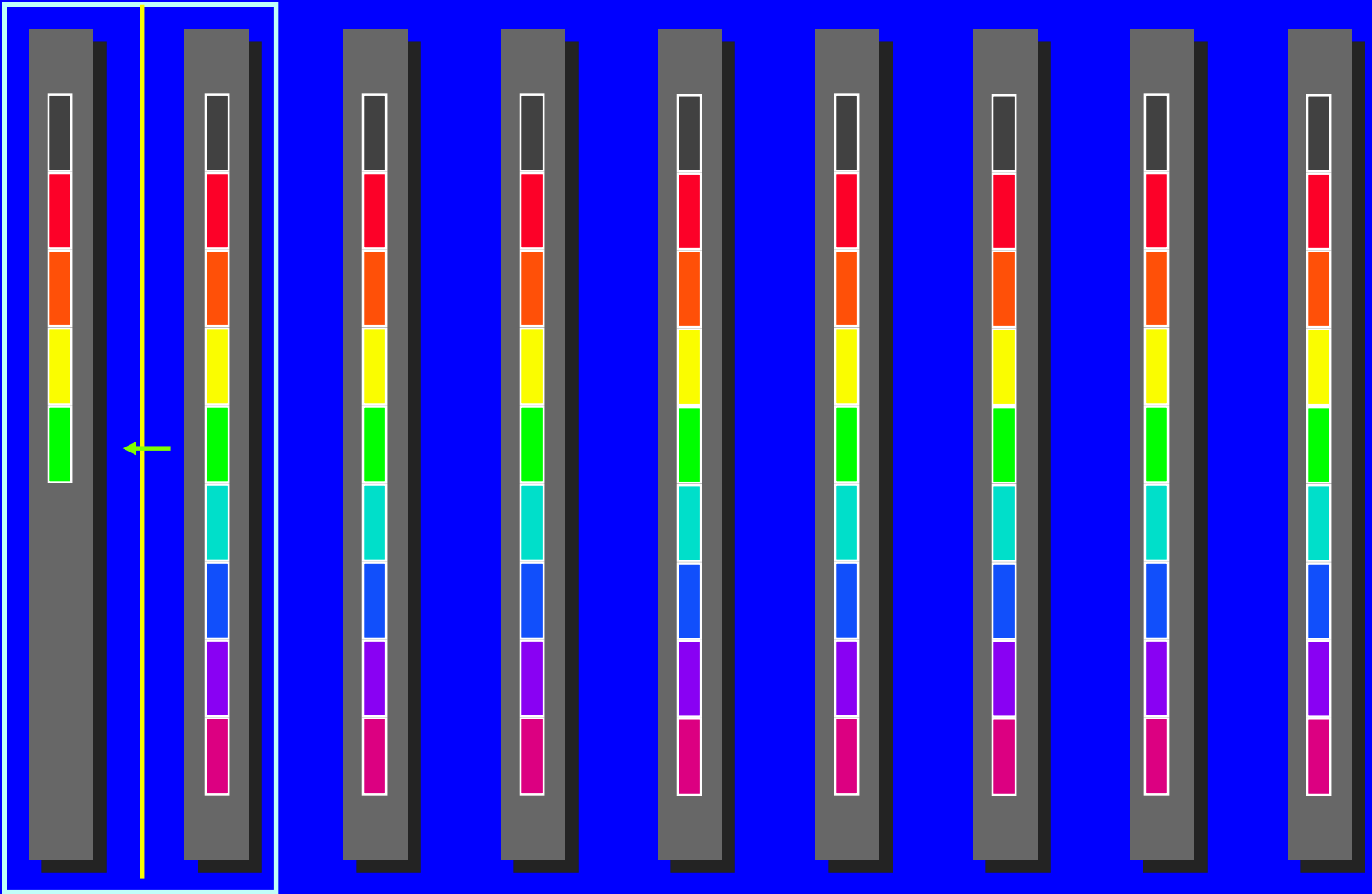


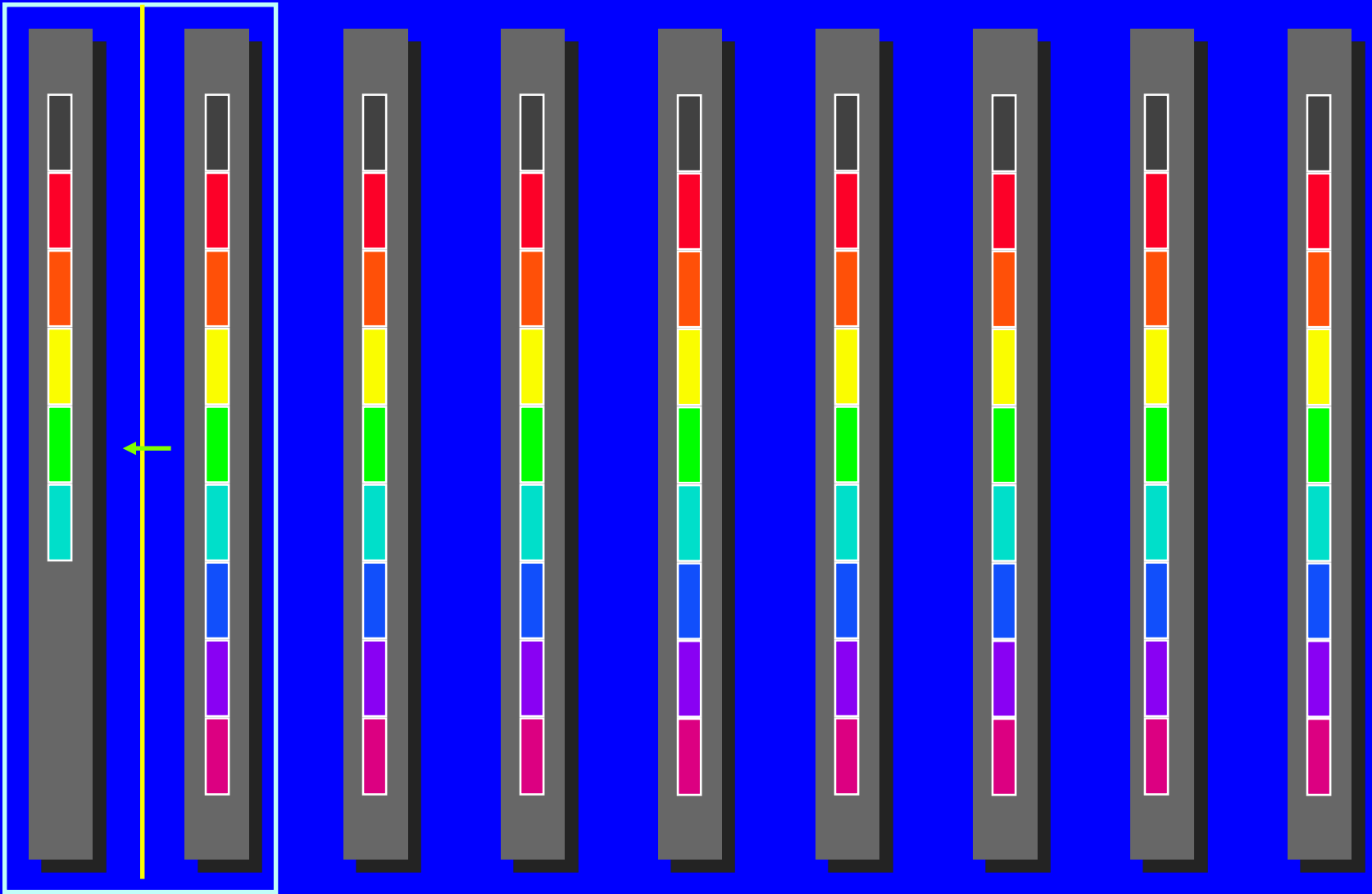


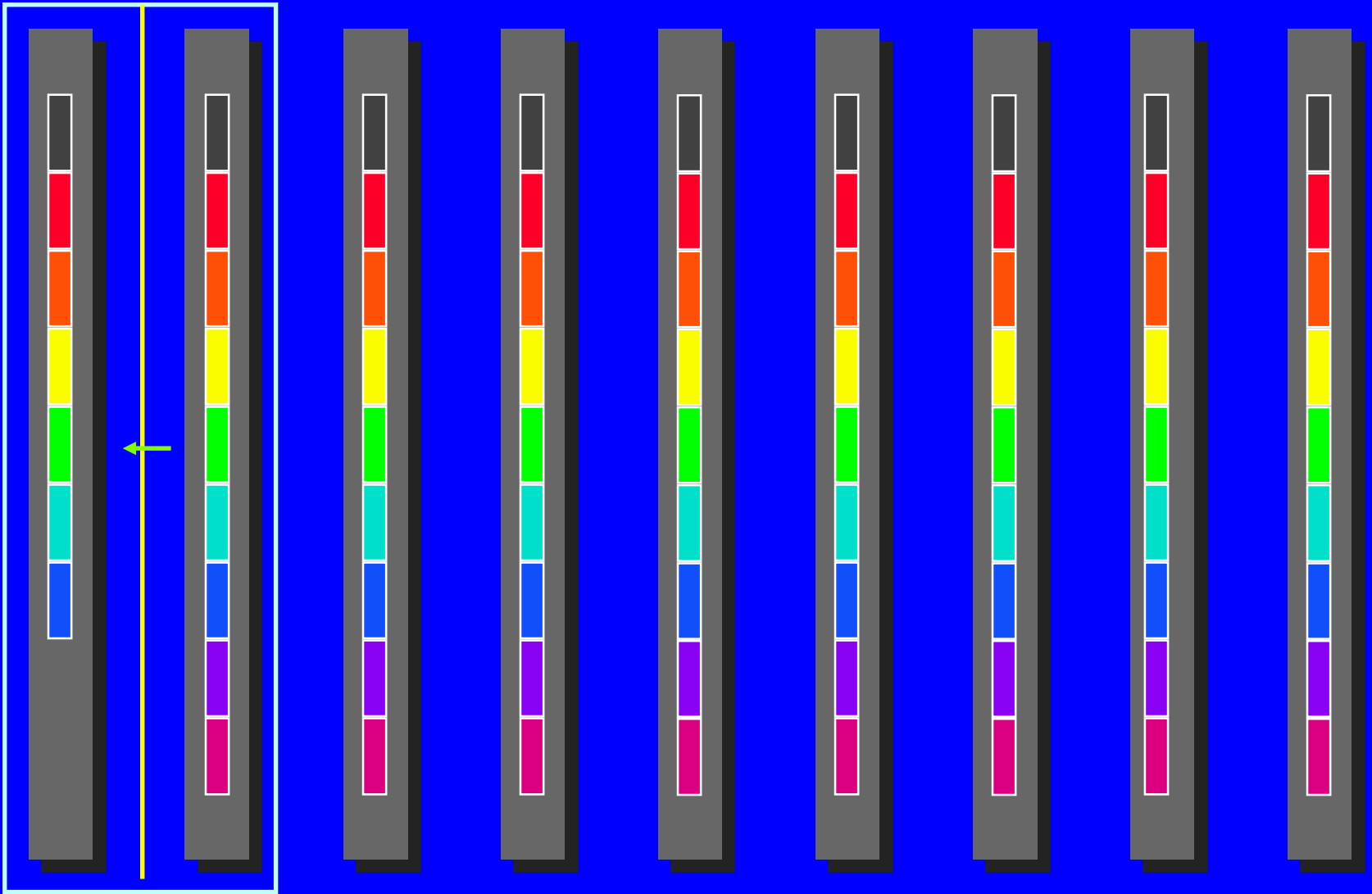


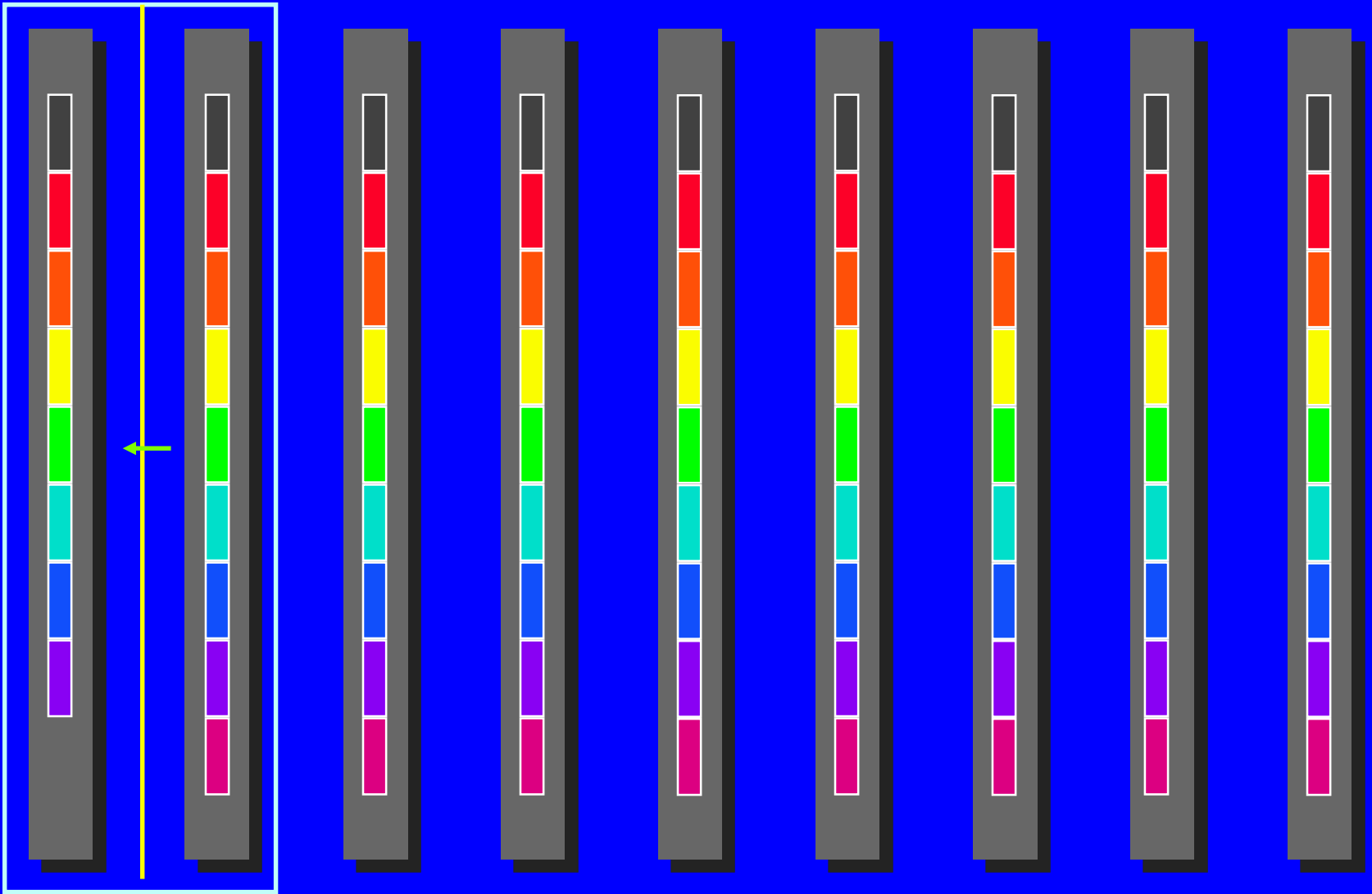


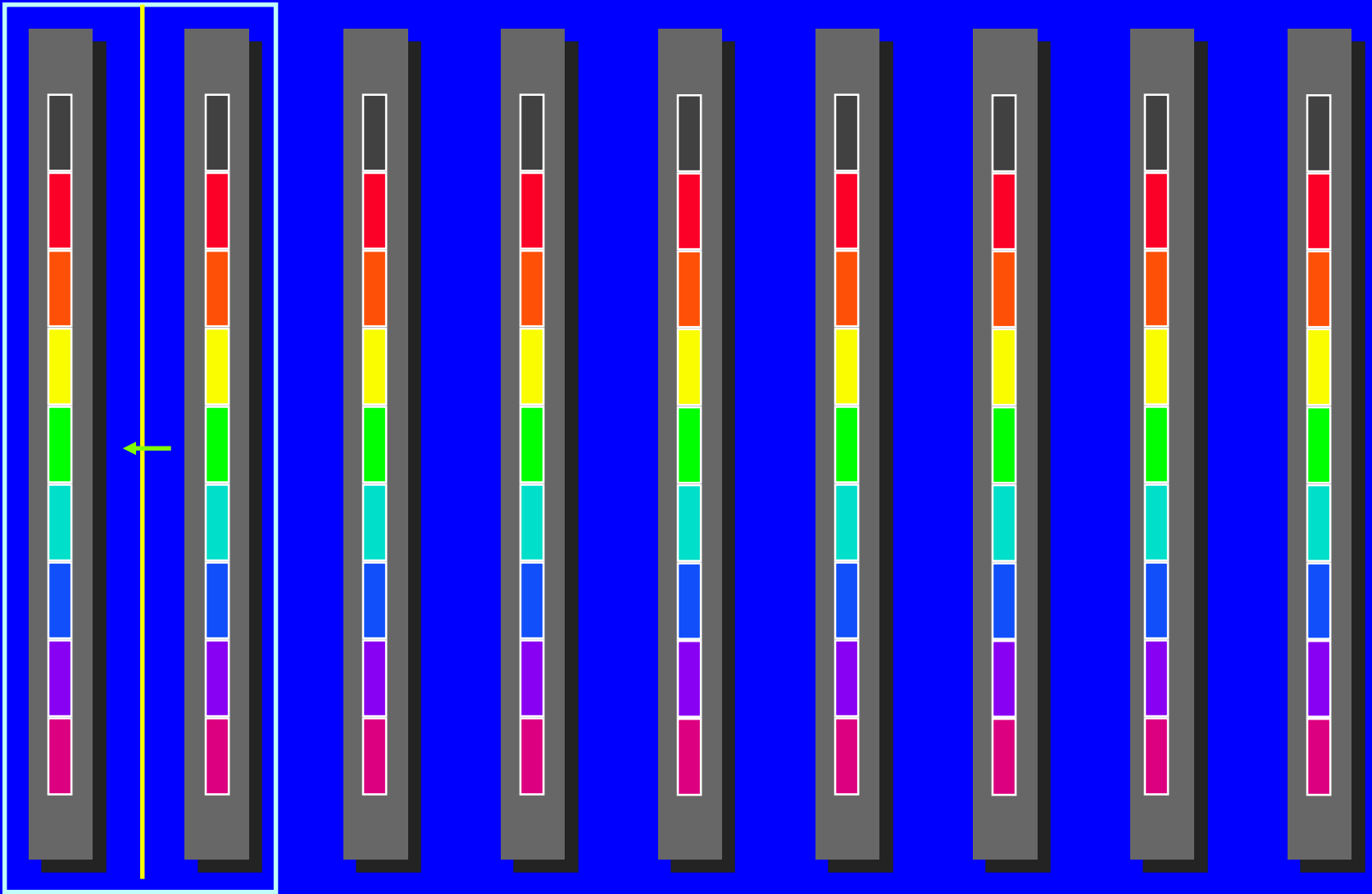


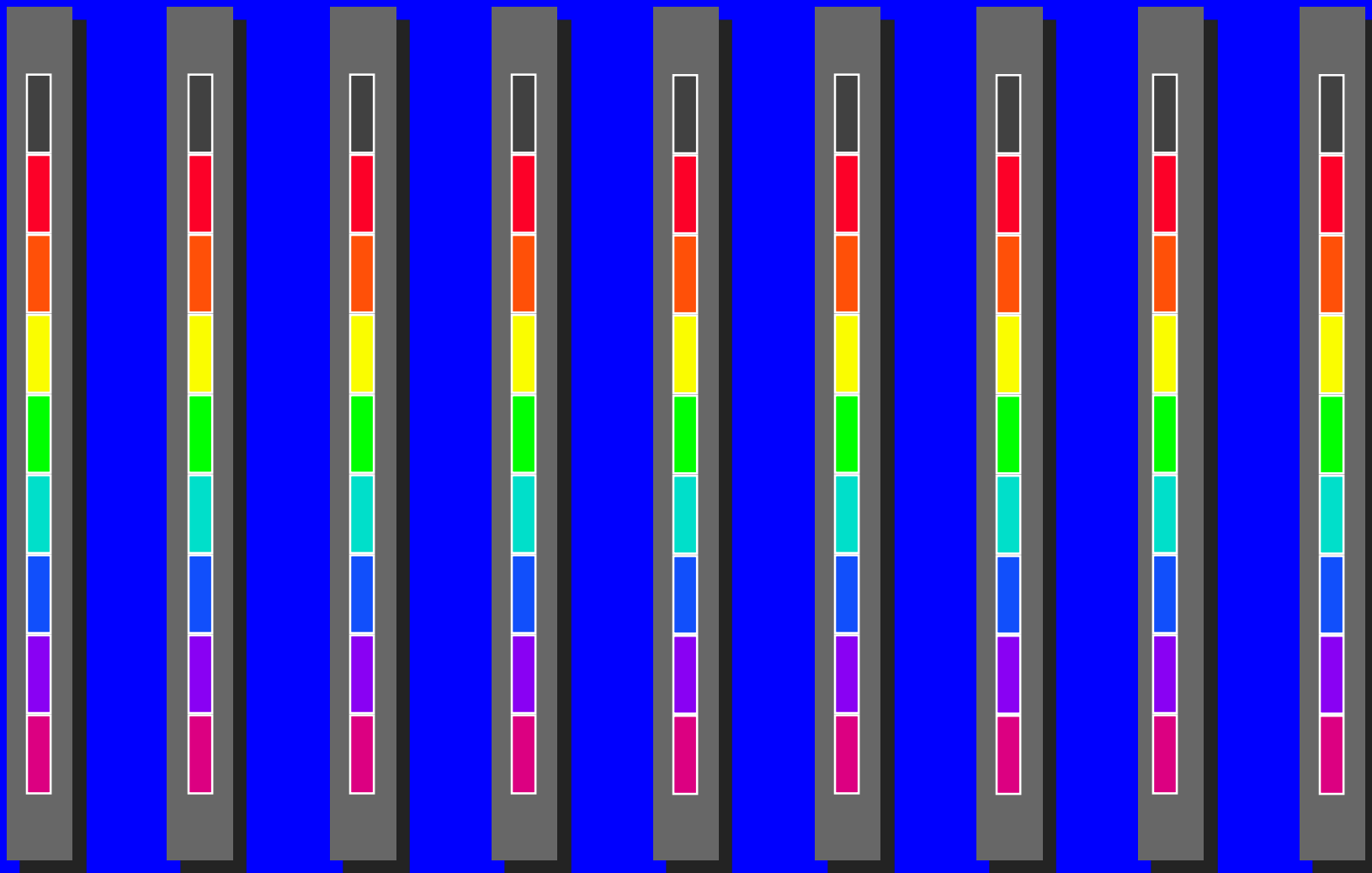












Cost of minimum spanning tree broadcast

The diagram illustrates the cost of minimum spanning tree broadcast as the product of two factors. The first factor, $\lceil \log(p) \rceil$, is enclosed in a blue box and labeled "number of steps" with a yellow arrow. The second factor, $(\alpha + n\beta)$, is enclosed in a red box and labeled "cost per steps" with a red arrow. The two boxes are placed side-by-side, representing the multiplication of the two terms.

$$\lceil \log(p) \rceil (\alpha + n\beta)$$

number of steps

cost per steps

Cost of minimum spanning tree broadcast

The diagram illustrates the cost of minimum spanning tree broadcast as the product of two components. The first component, $\lceil \log(p) \rceil$, is enclosed in a yellow box and labeled "number of steps" with a yellow arrow. The second component, $(\alpha + n\beta)$, is enclosed in a red box and labeled "cost per steps" with a red arrow. The two boxes are placed side-by-side, representing the multiplication of these two factors.

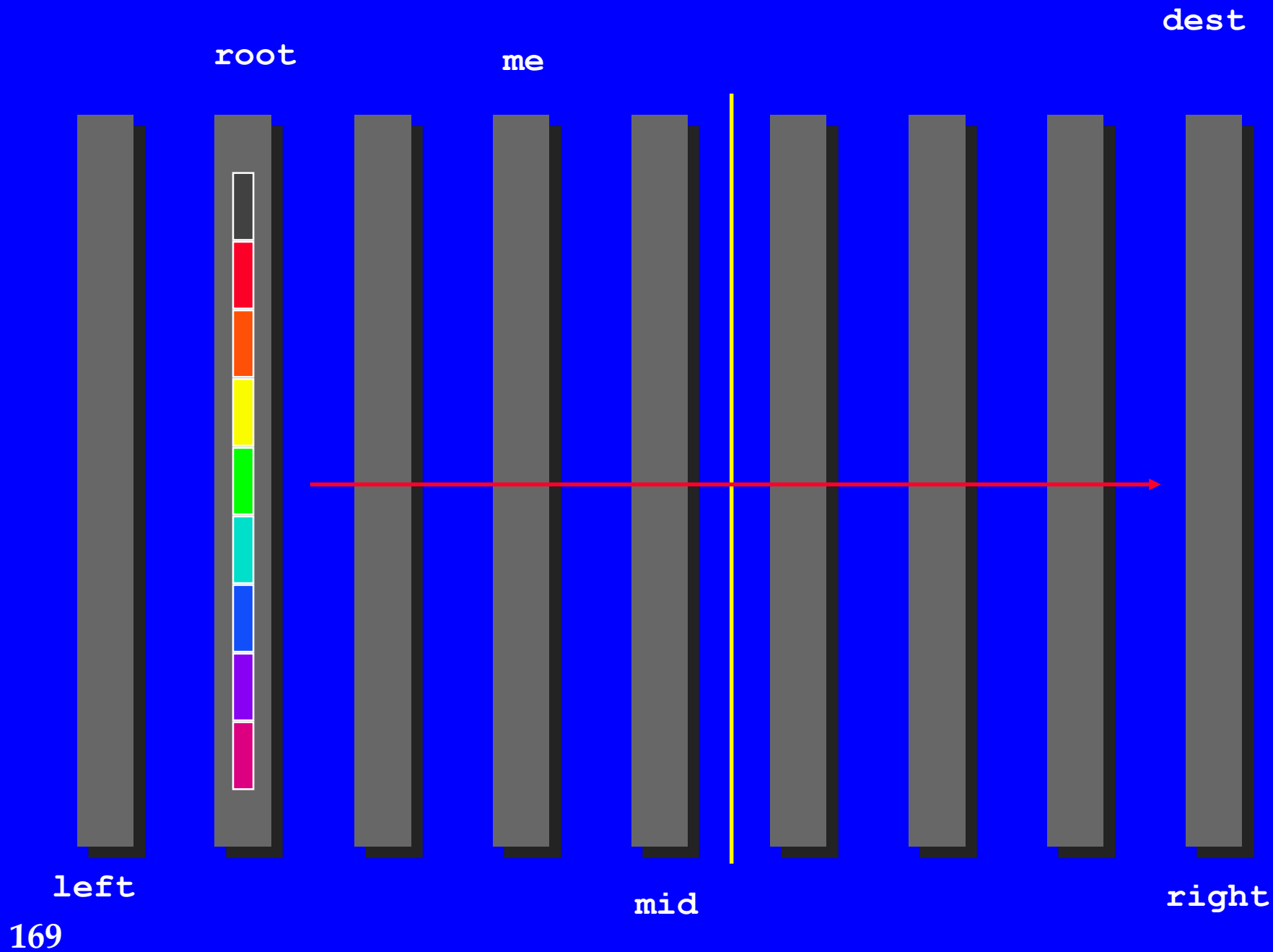
$$\lceil \log(p) \rceil (\alpha + n\beta)$$

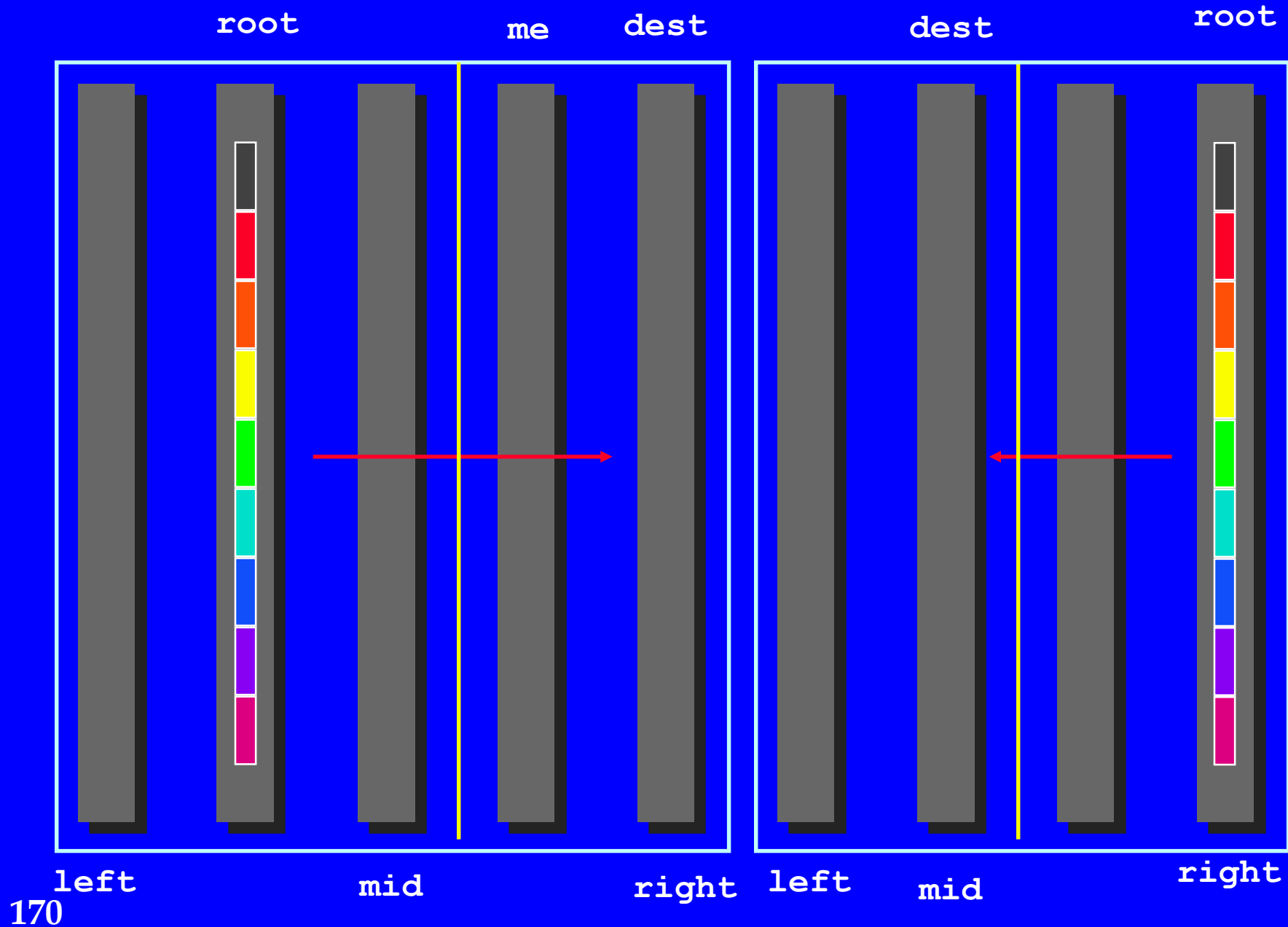
number of steps

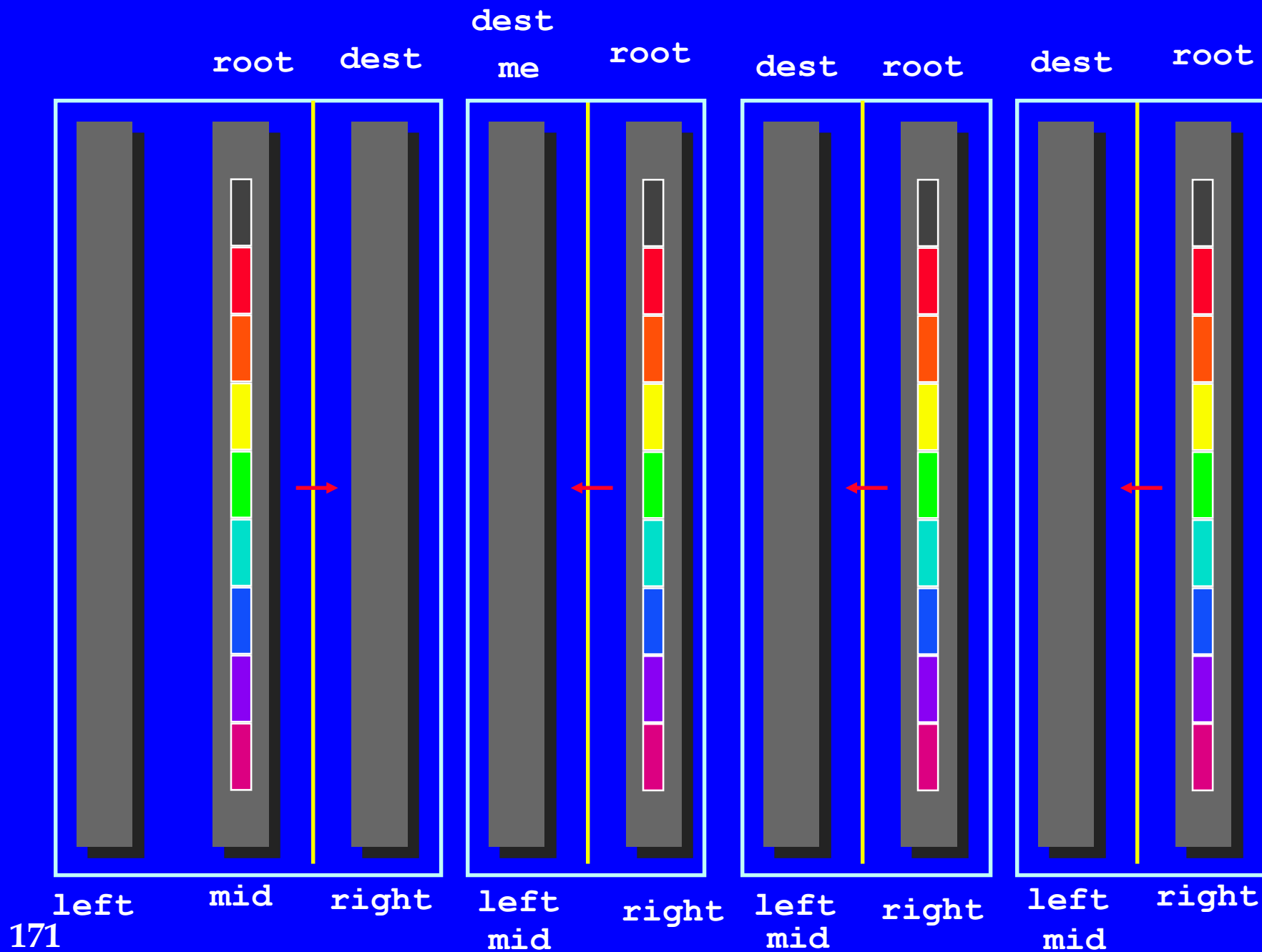
cost per steps

Notice: attains lower bound for latency component

QuickTime™ and a
decompressor
are needed to see this picture.

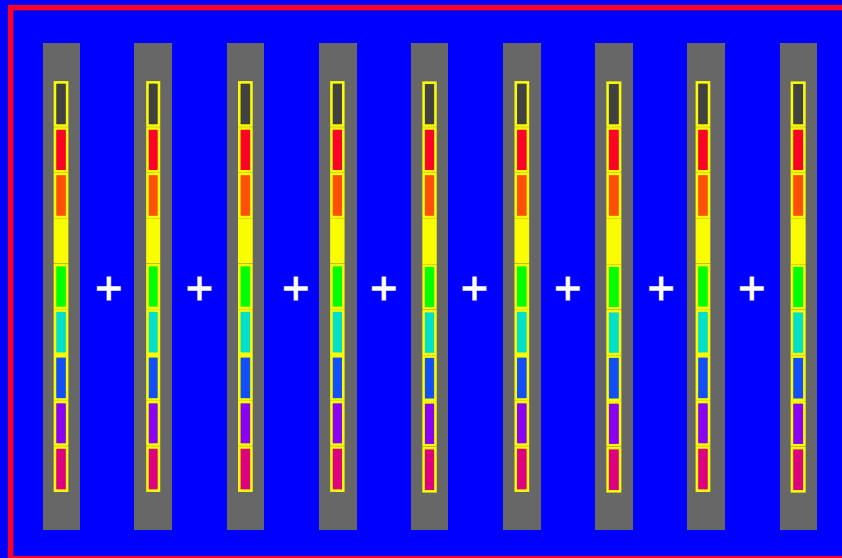




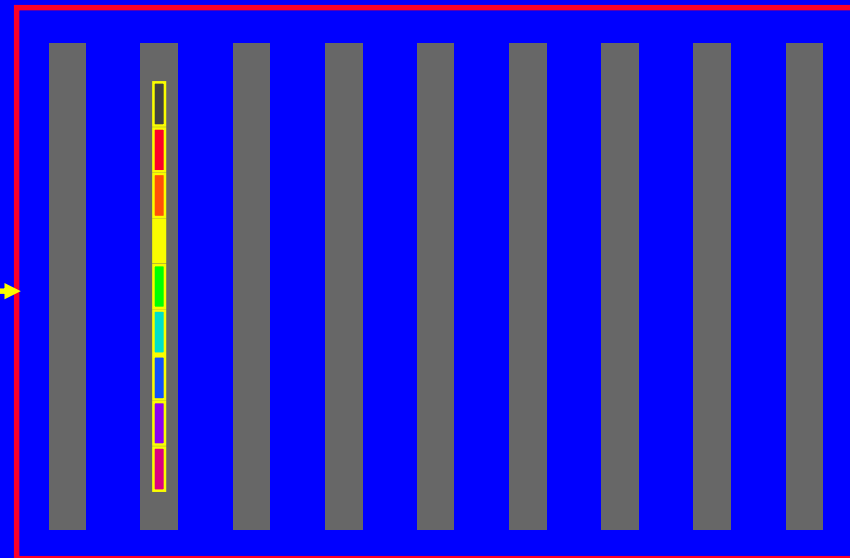


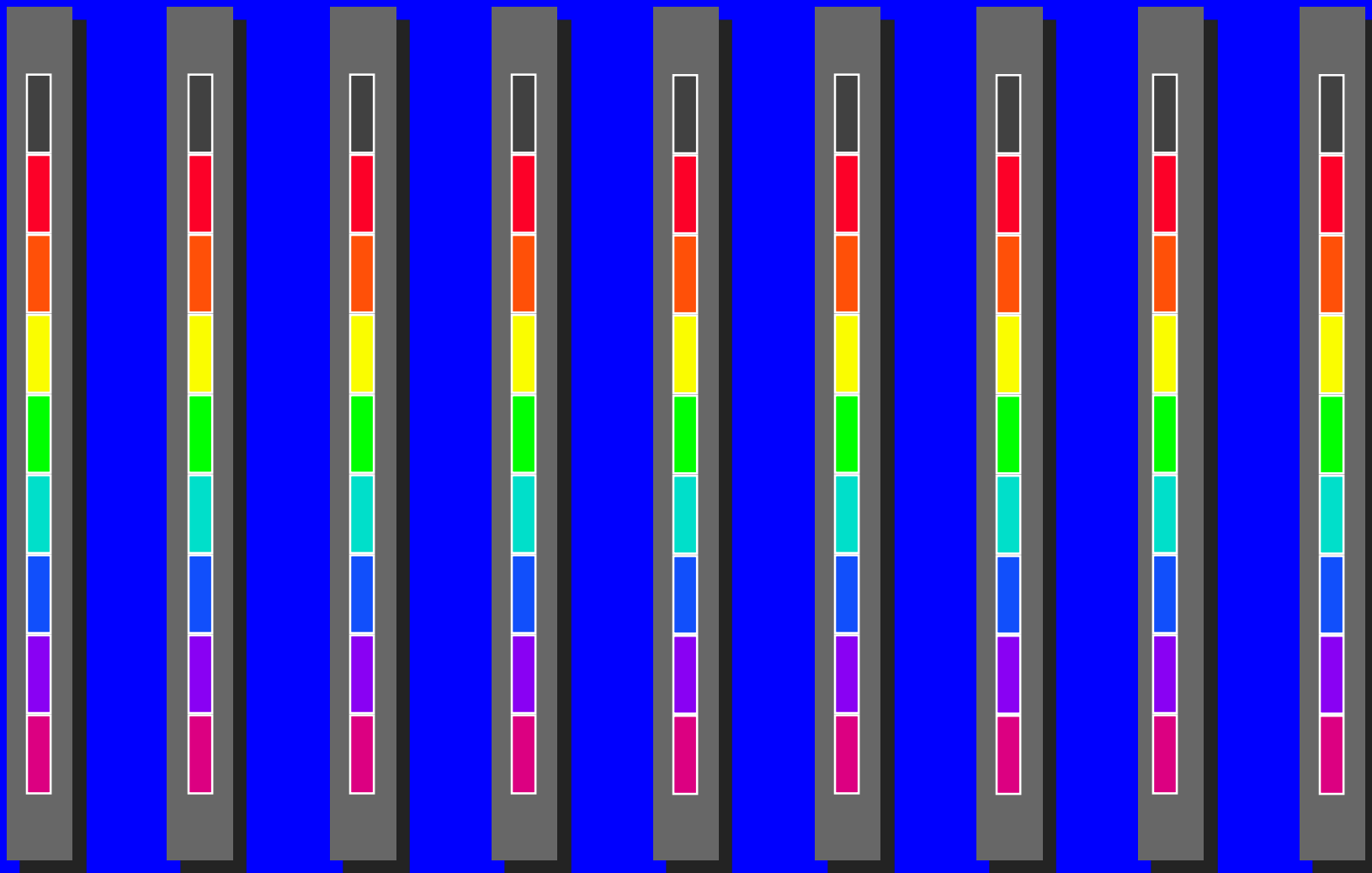
Reduce(-to-one)

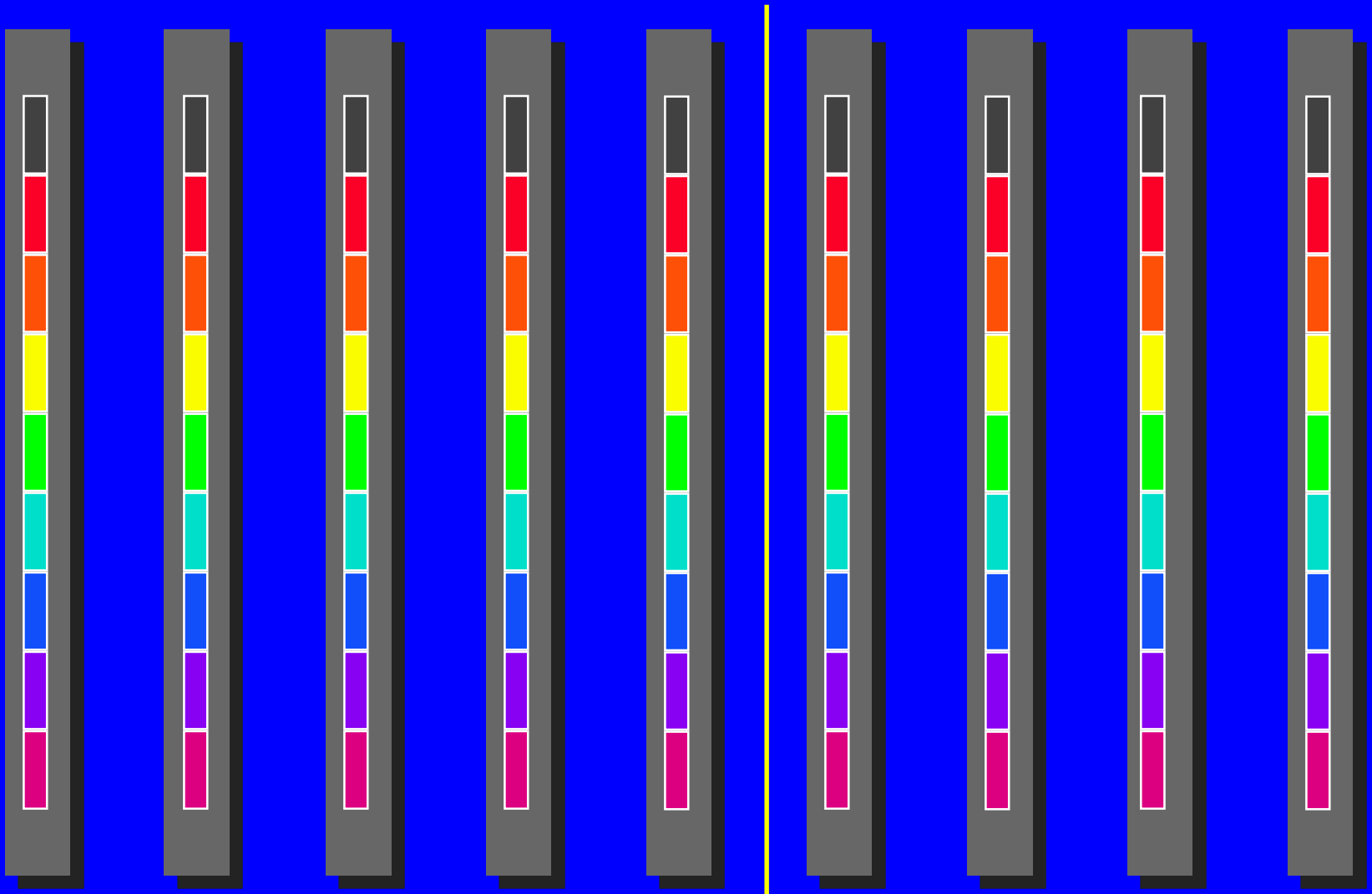
Before

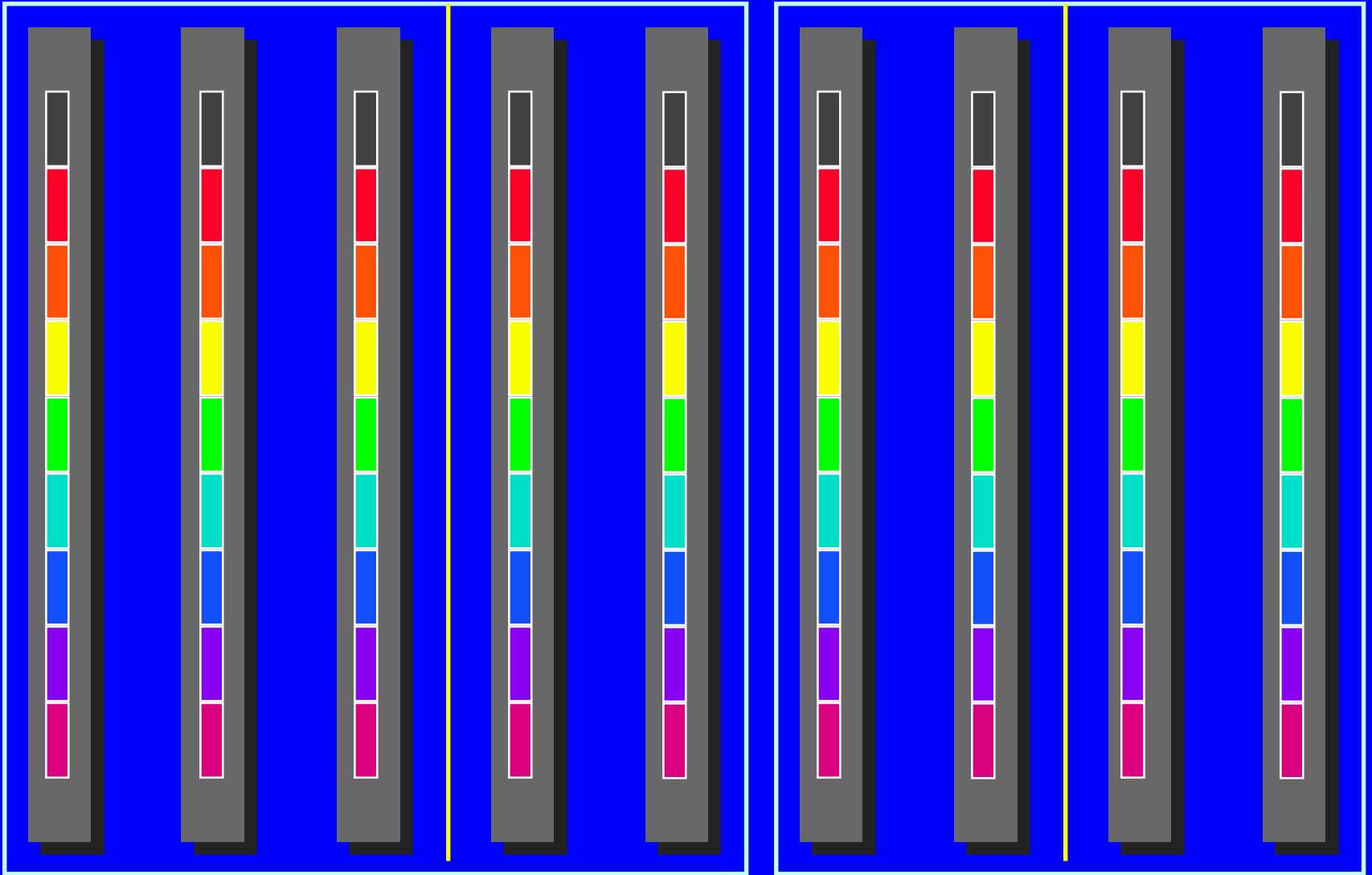


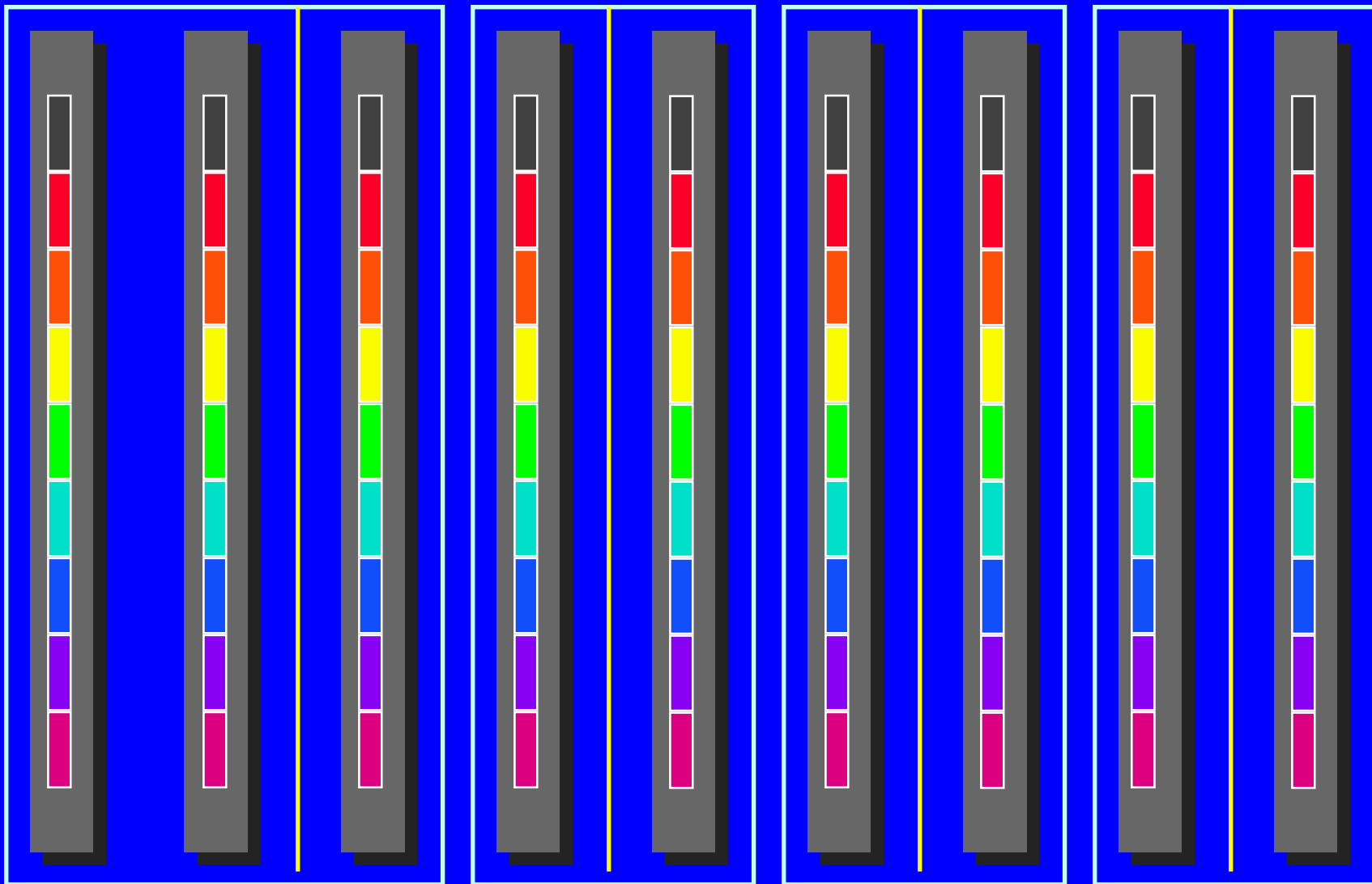
After

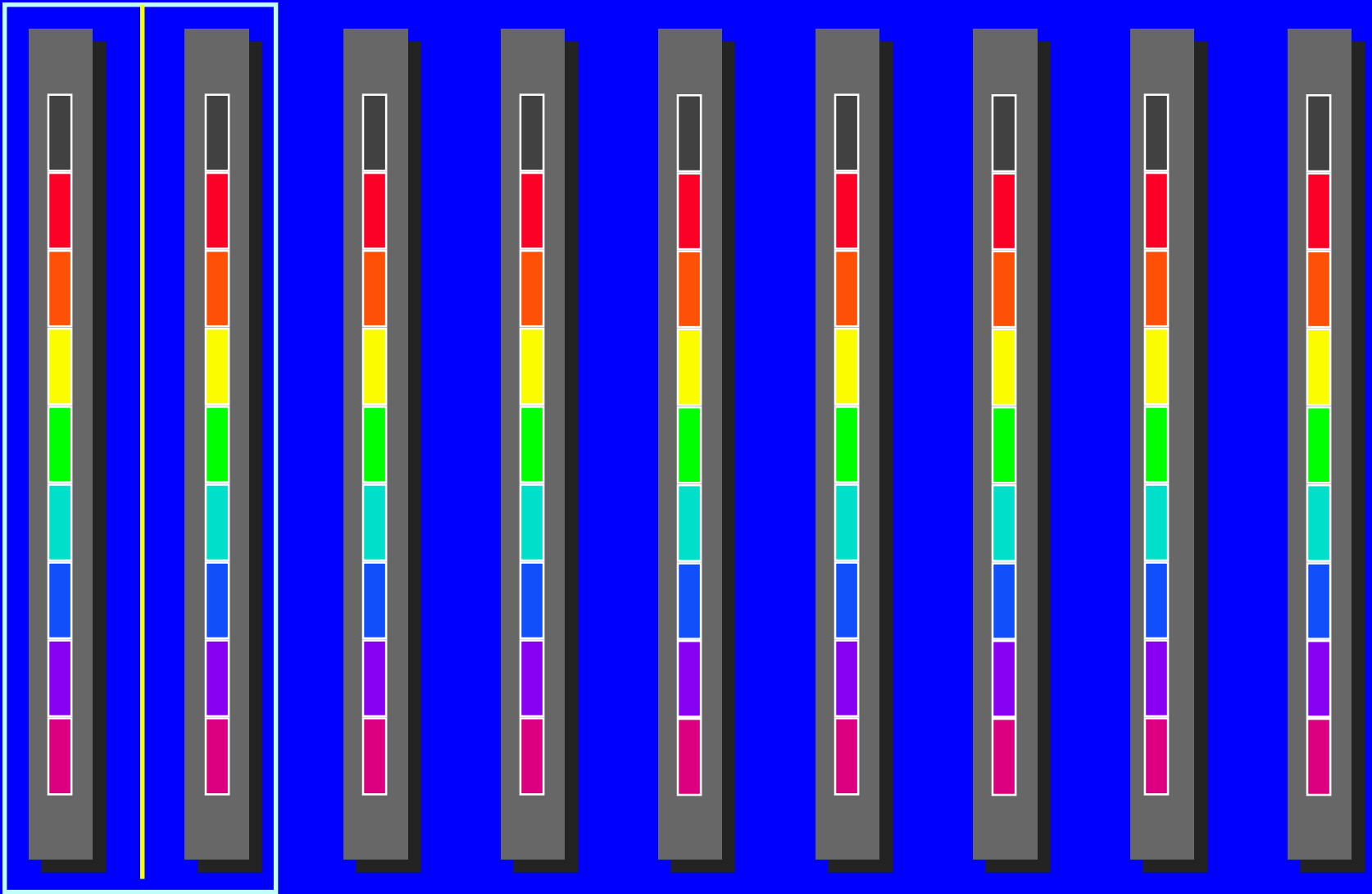


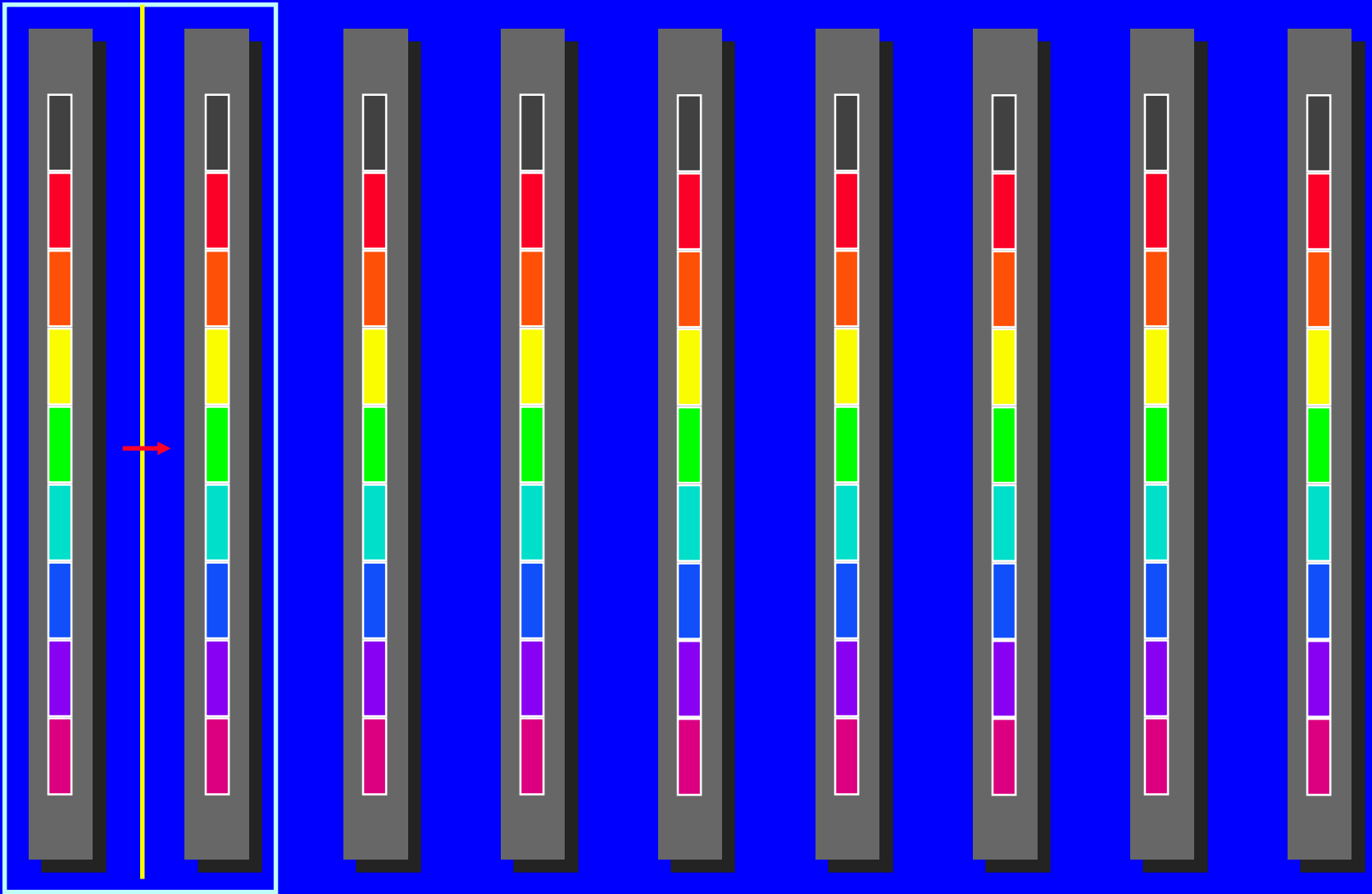


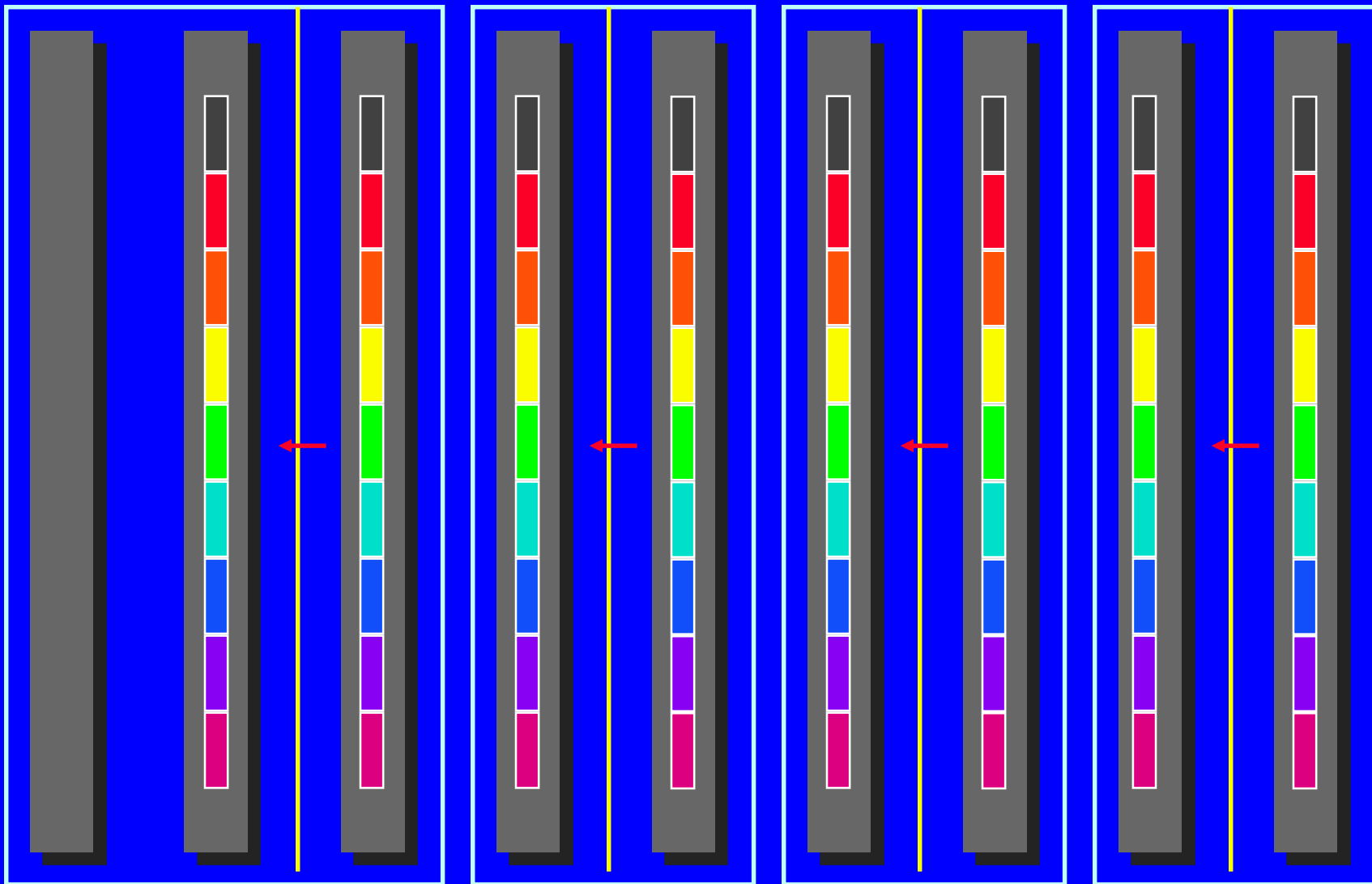


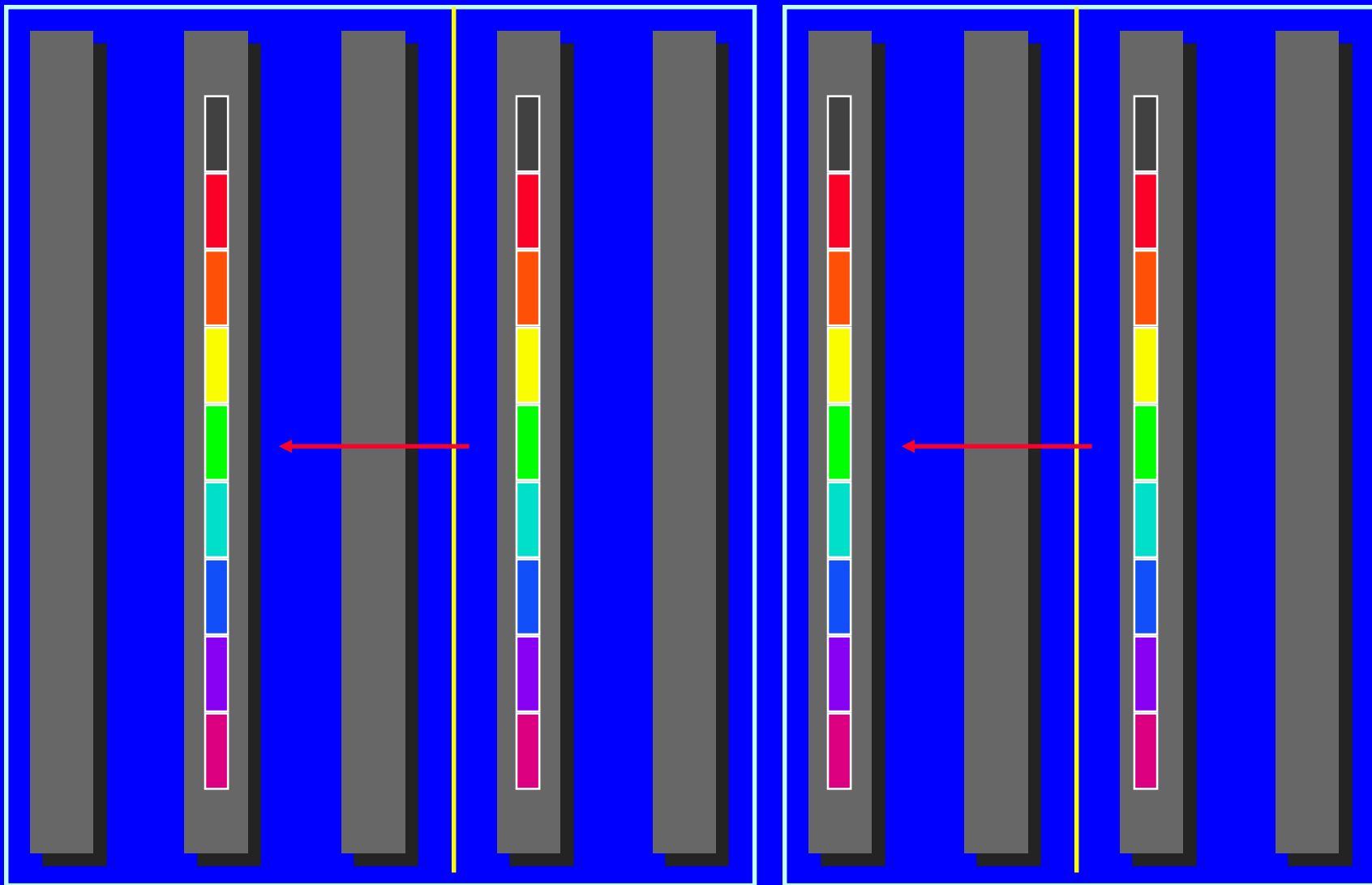


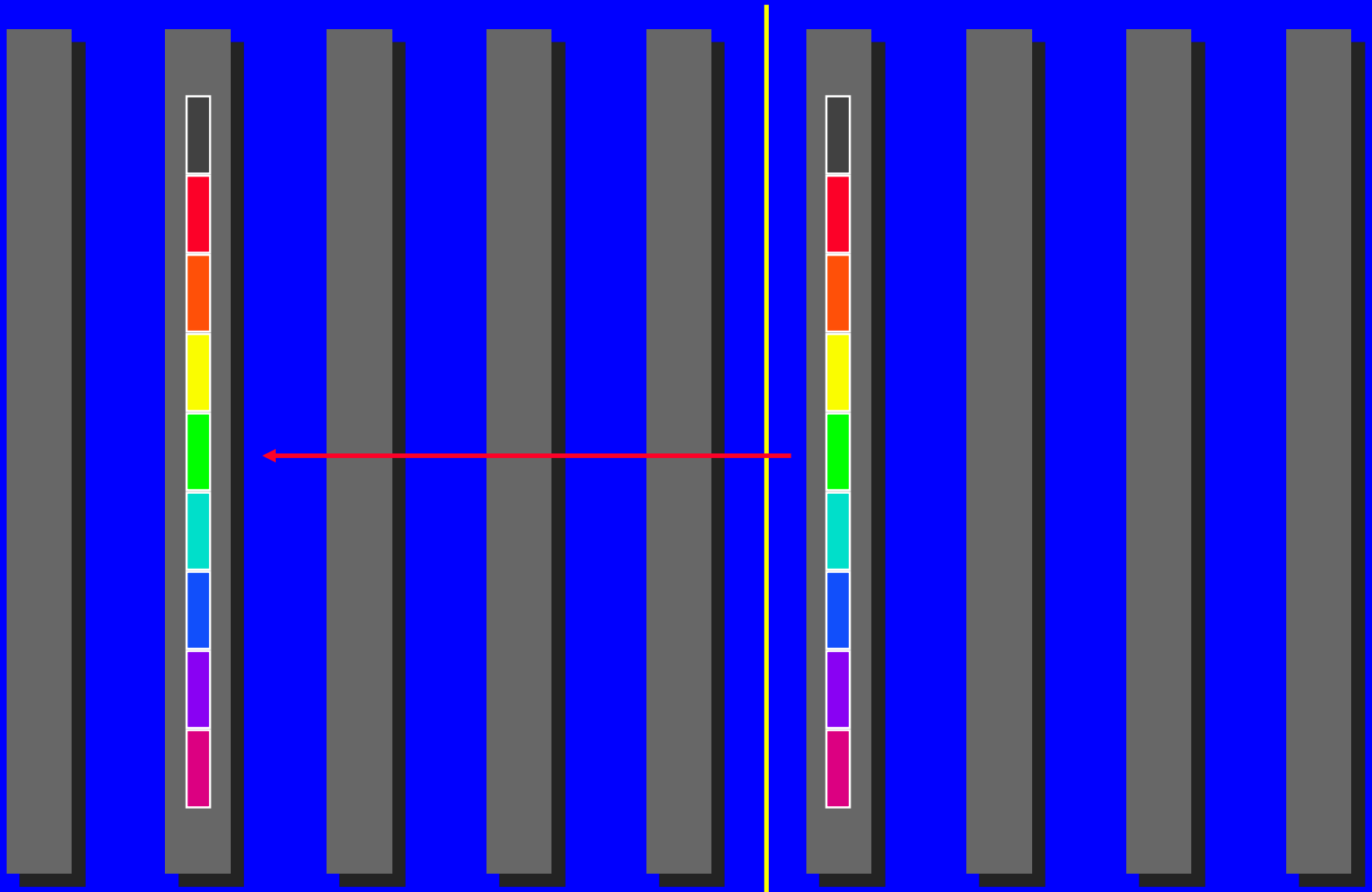












Cost of minimum spanning tree reduce(-to-one)

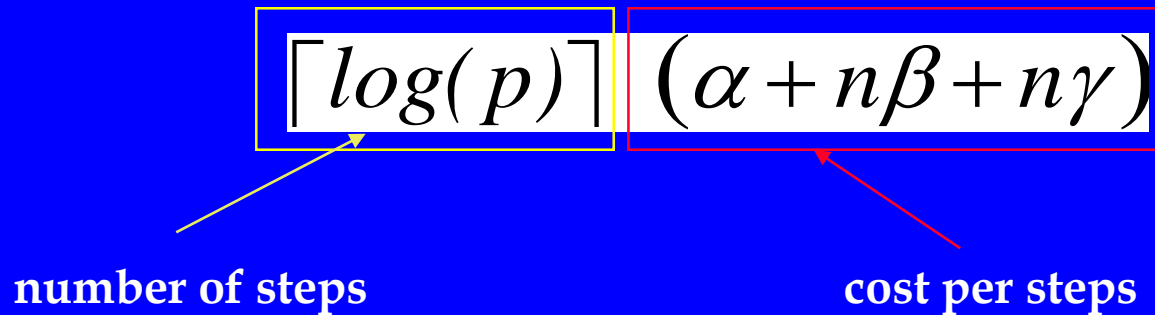
The diagram illustrates the cost formula for a minimum spanning tree reduce(-to-one) operation. The formula is presented as two adjacent rectangular boxes. The first box, outlined in blue, contains the expression $\lceil \log(p) \rceil$. A yellow arrow points from the text 'number of steps' below to this box. The second box, outlined in red, contains the expression $(\alpha + n\beta + n\gamma)$. A red arrow points from the text 'cost per steps' below to this box.

$$\lceil \log(p) \rceil \quad (\alpha + n\beta + n\gamma)$$

number of steps

cost per steps

Cost of minimum spanning tree reduce(-to-one)



The diagram shows the formula $\lceil \log(p) \rceil (\alpha + n\beta + n\gamma)$ with two boxes and arrows. A blue box highlights $\lceil \log(p) \rceil$ with a yellow arrow pointing to the text 'number of steps' below it. A red box highlights $(\alpha + n\beta + n\gamma)$ with a red arrow pointing to the text 'cost per steps' below it.

$$\lceil \log(p) \rceil (\alpha + n\beta + n\gamma)$$

number of steps

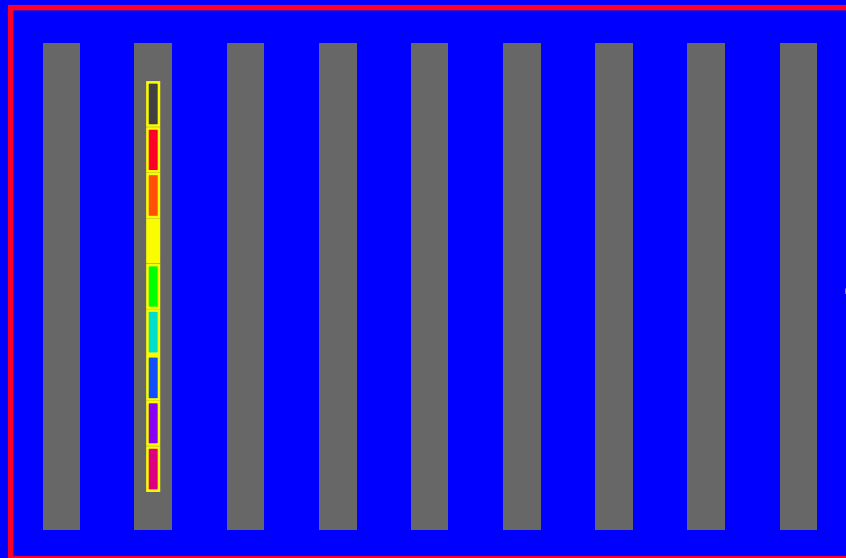
cost per steps

Notice: attains lower bound for latency component

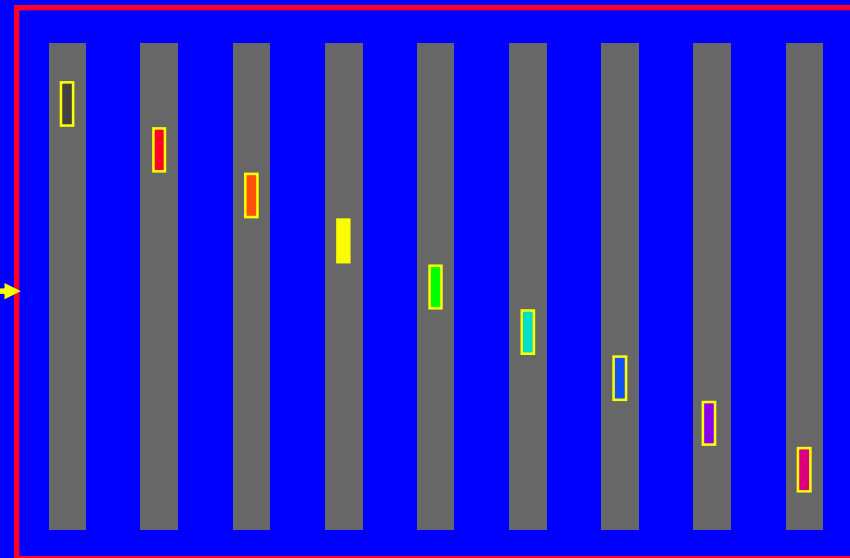
QuickTime™ and a
decompressor
are needed to see this picture.

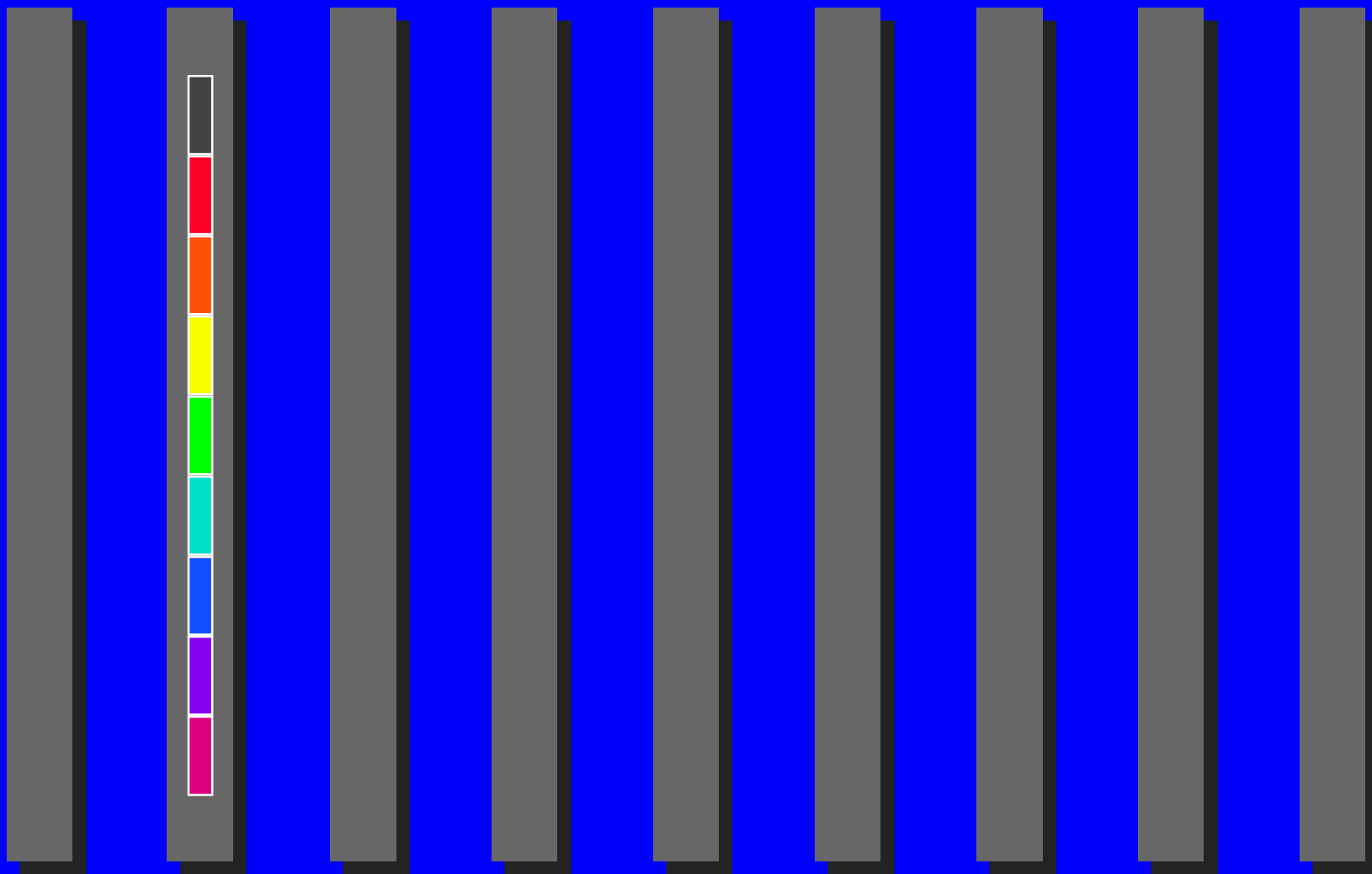
Scatter

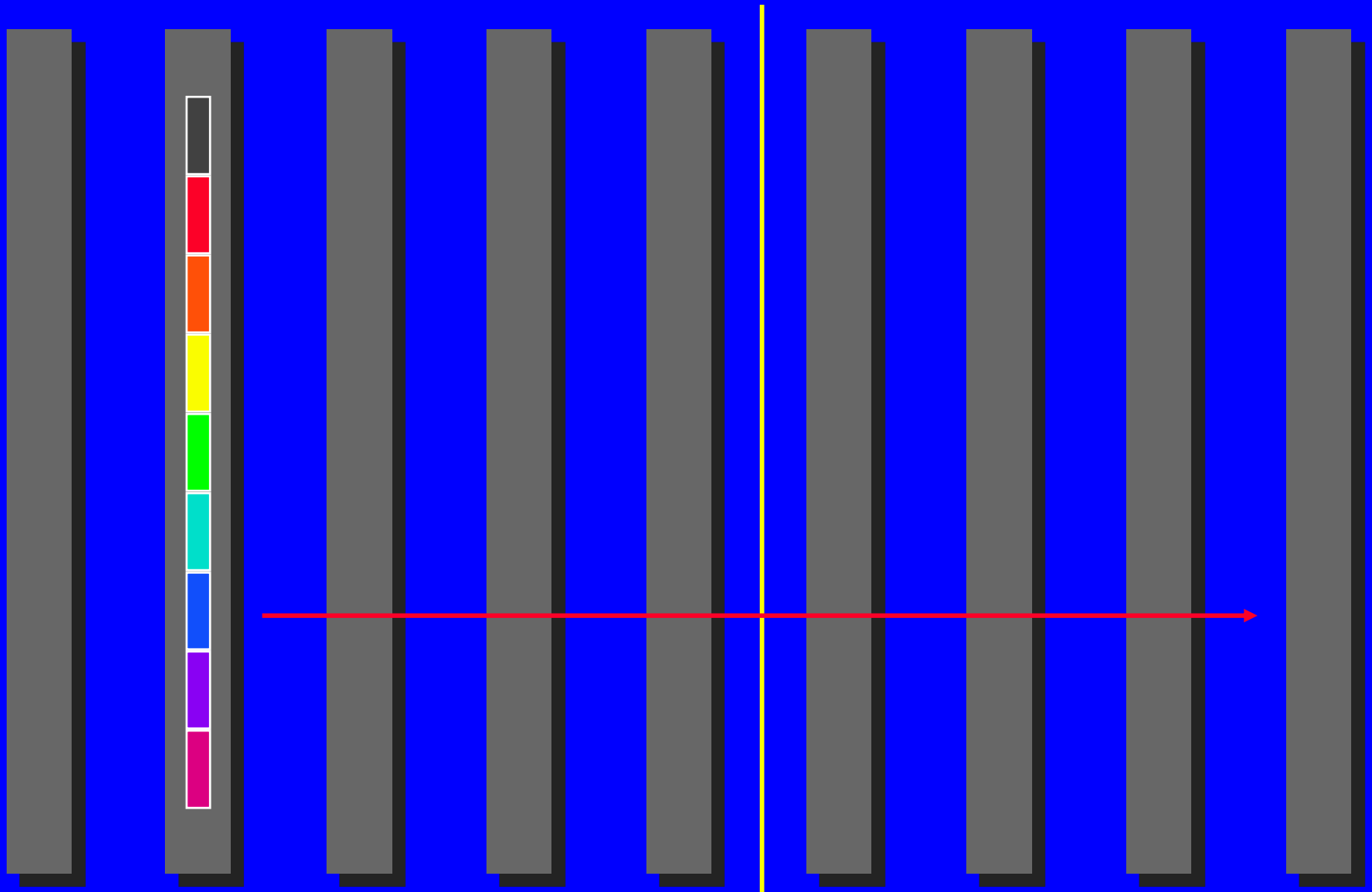
Before

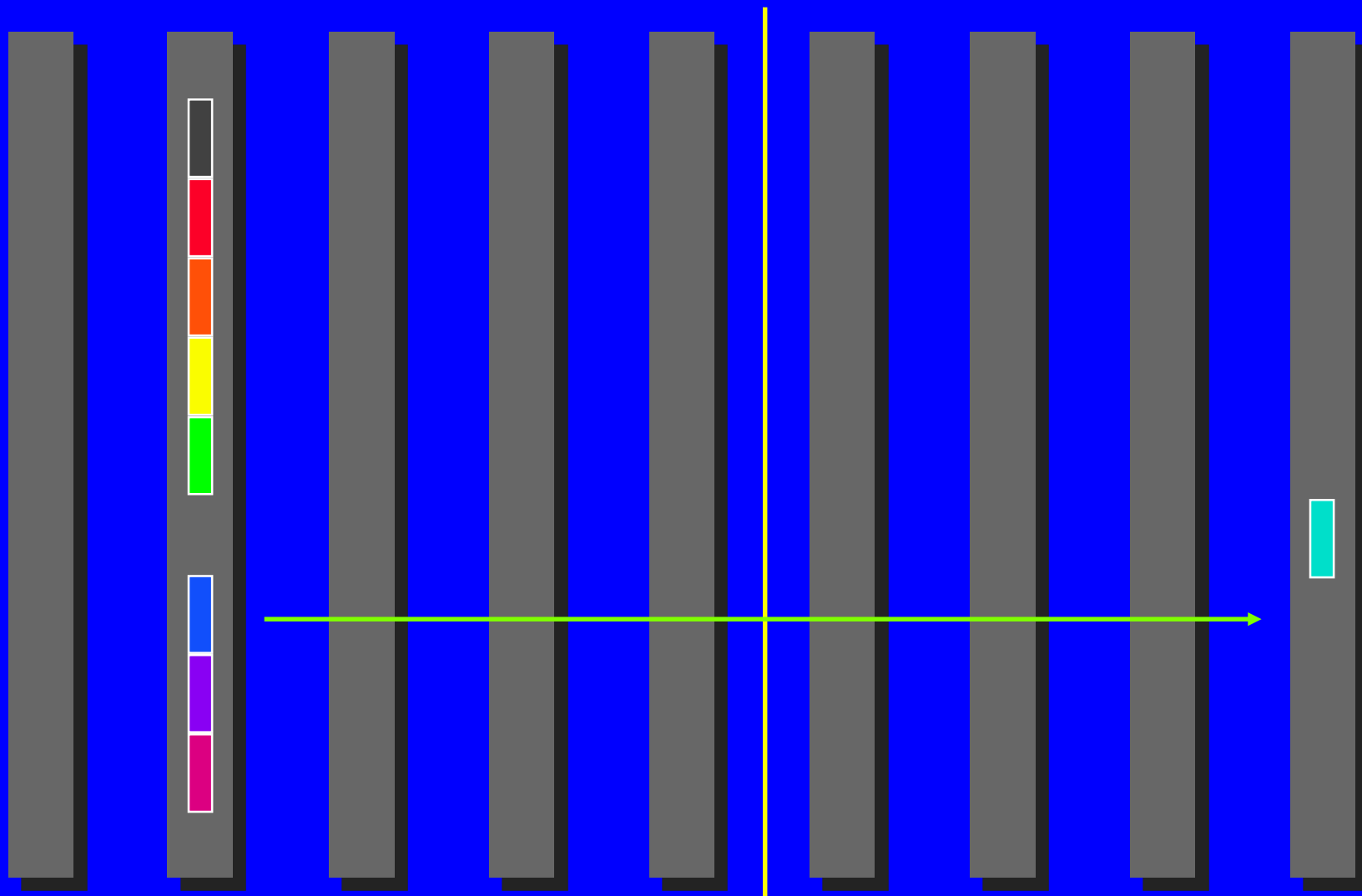


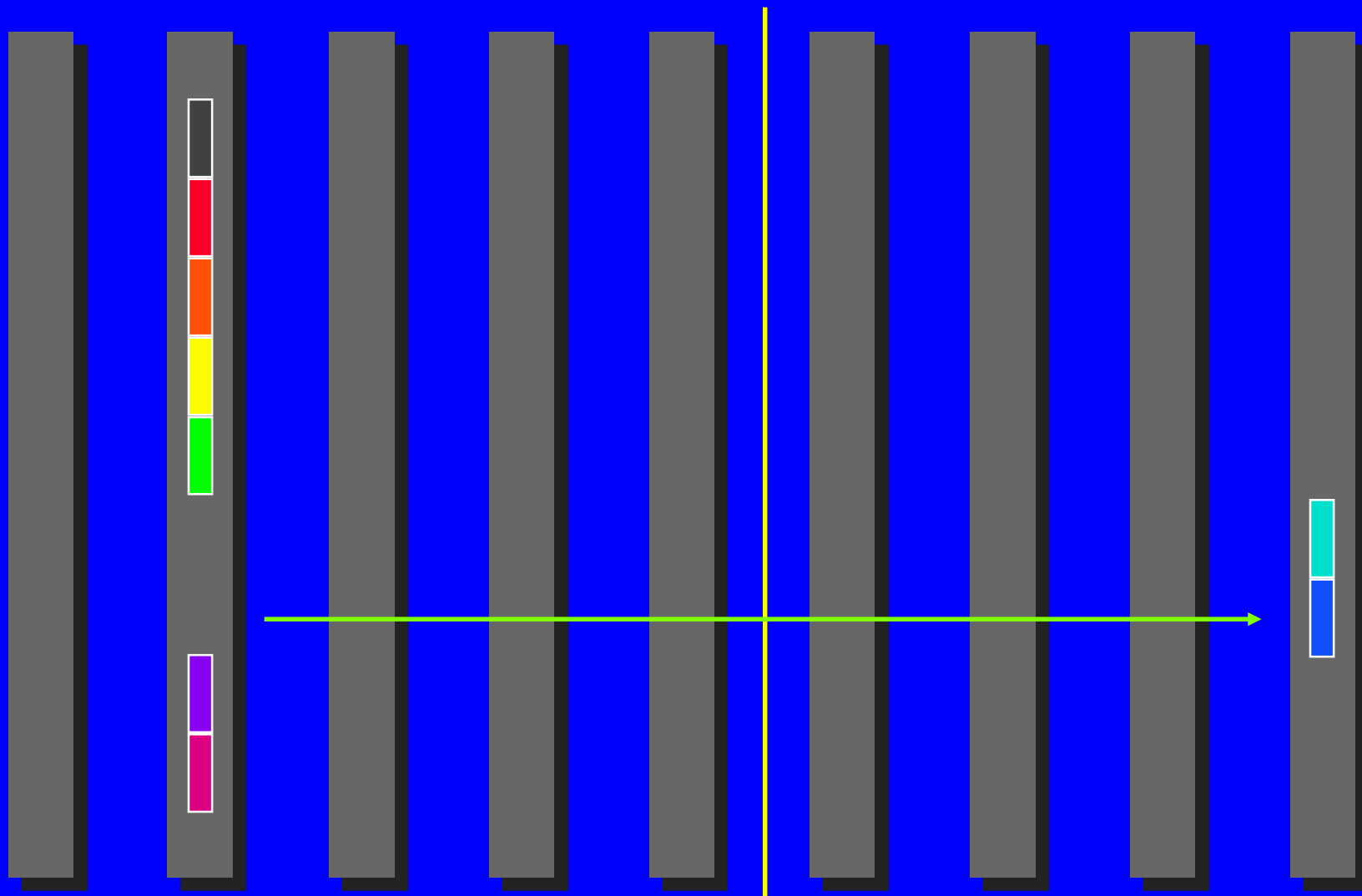
After

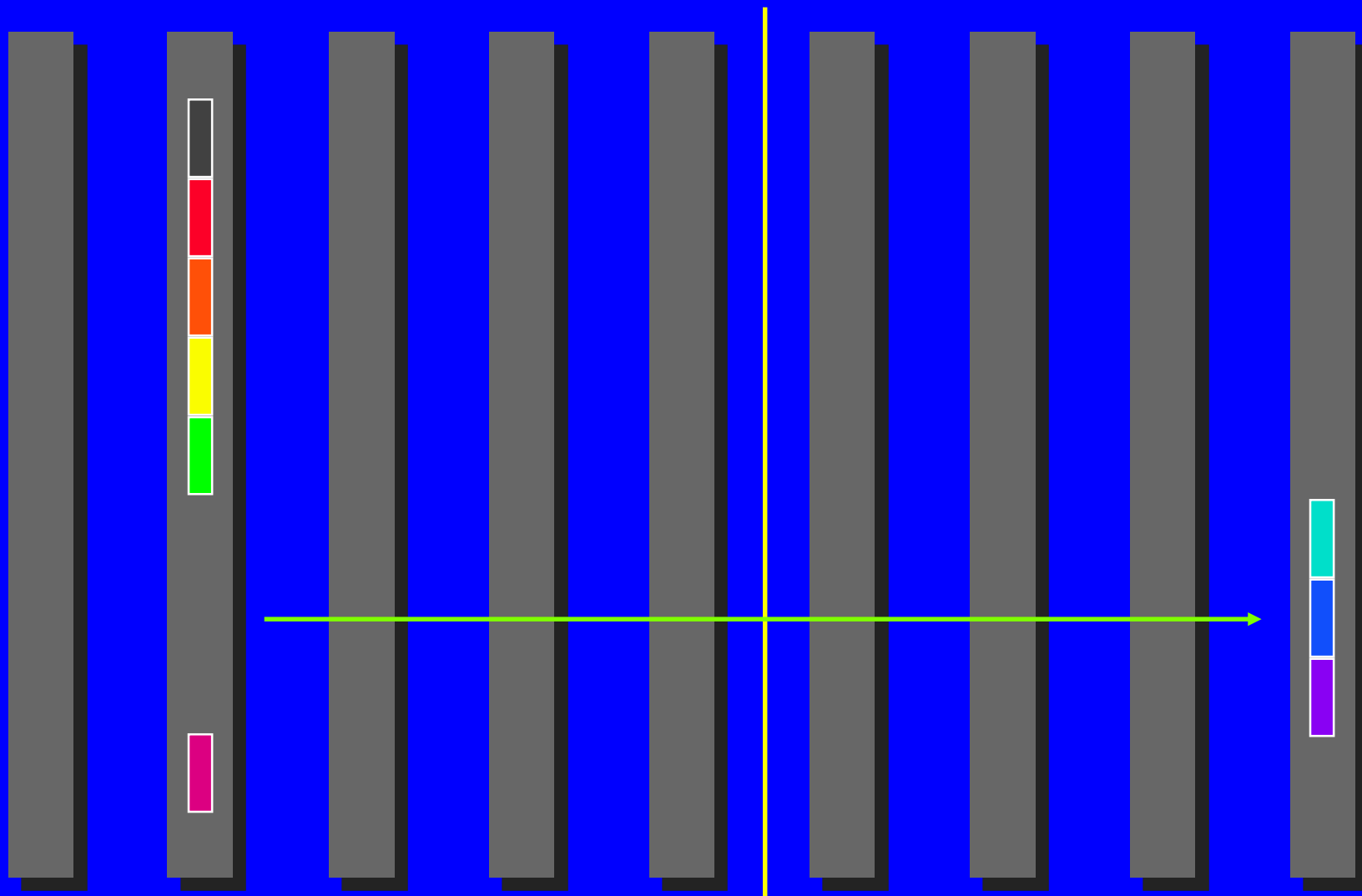


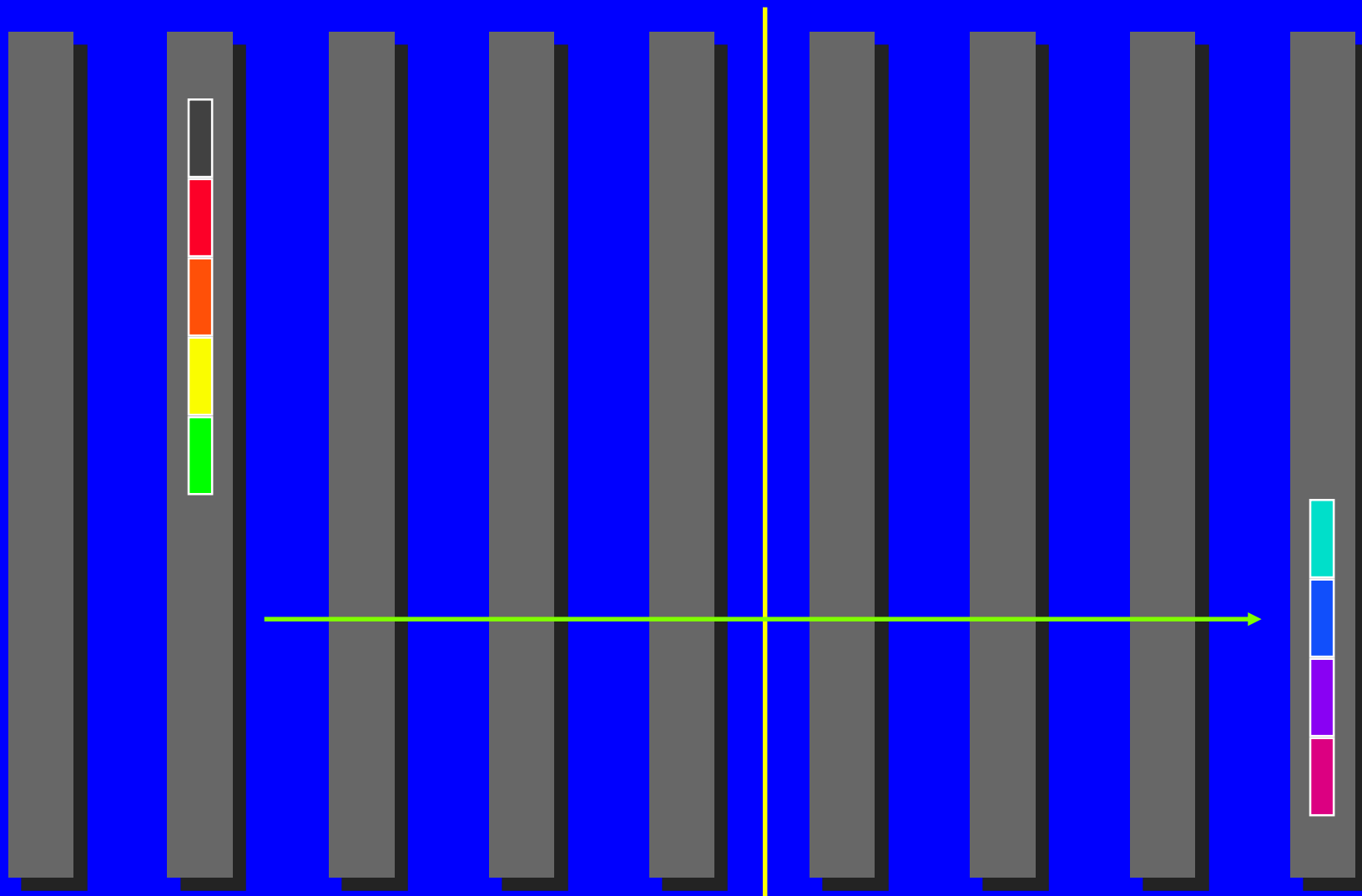


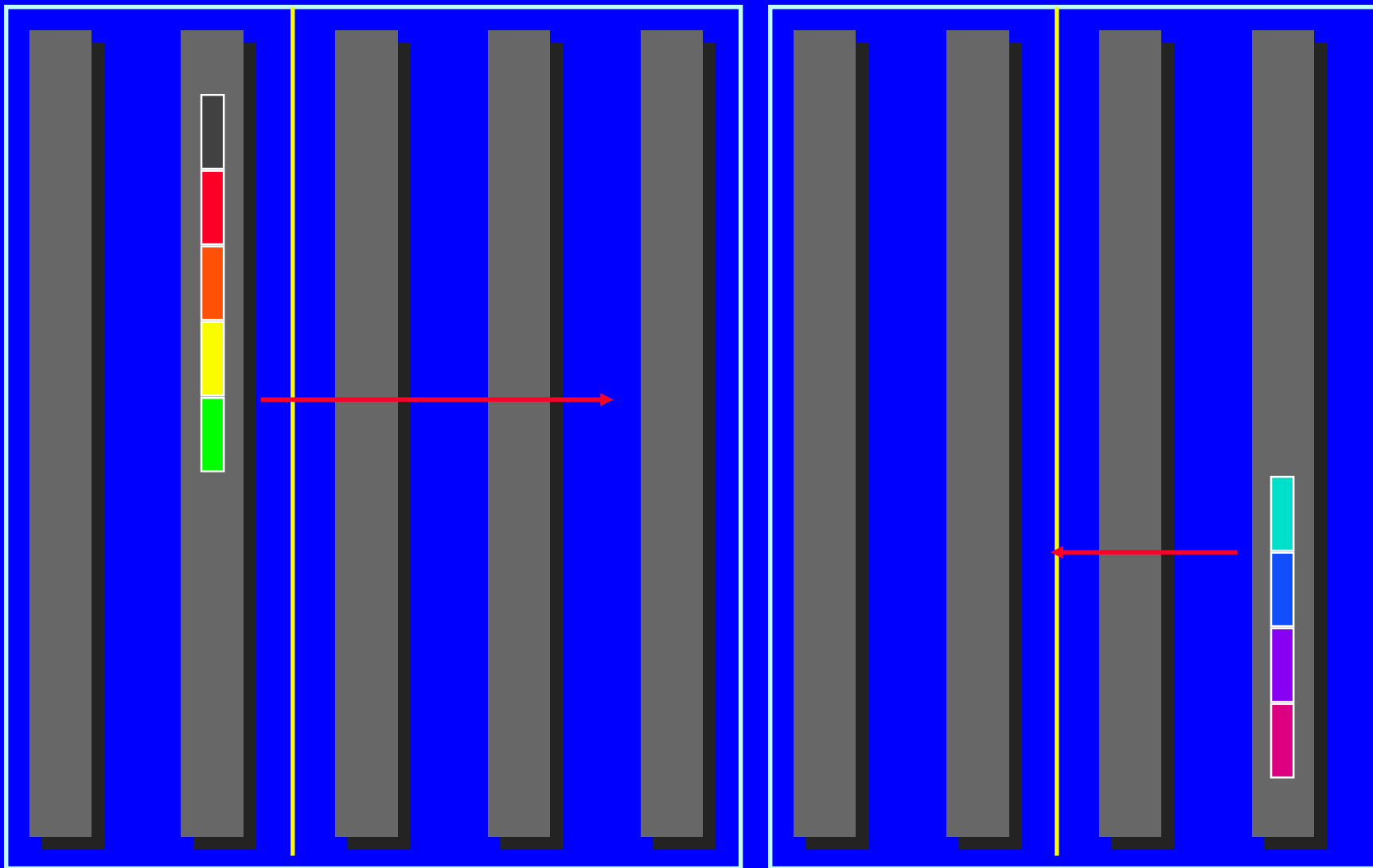


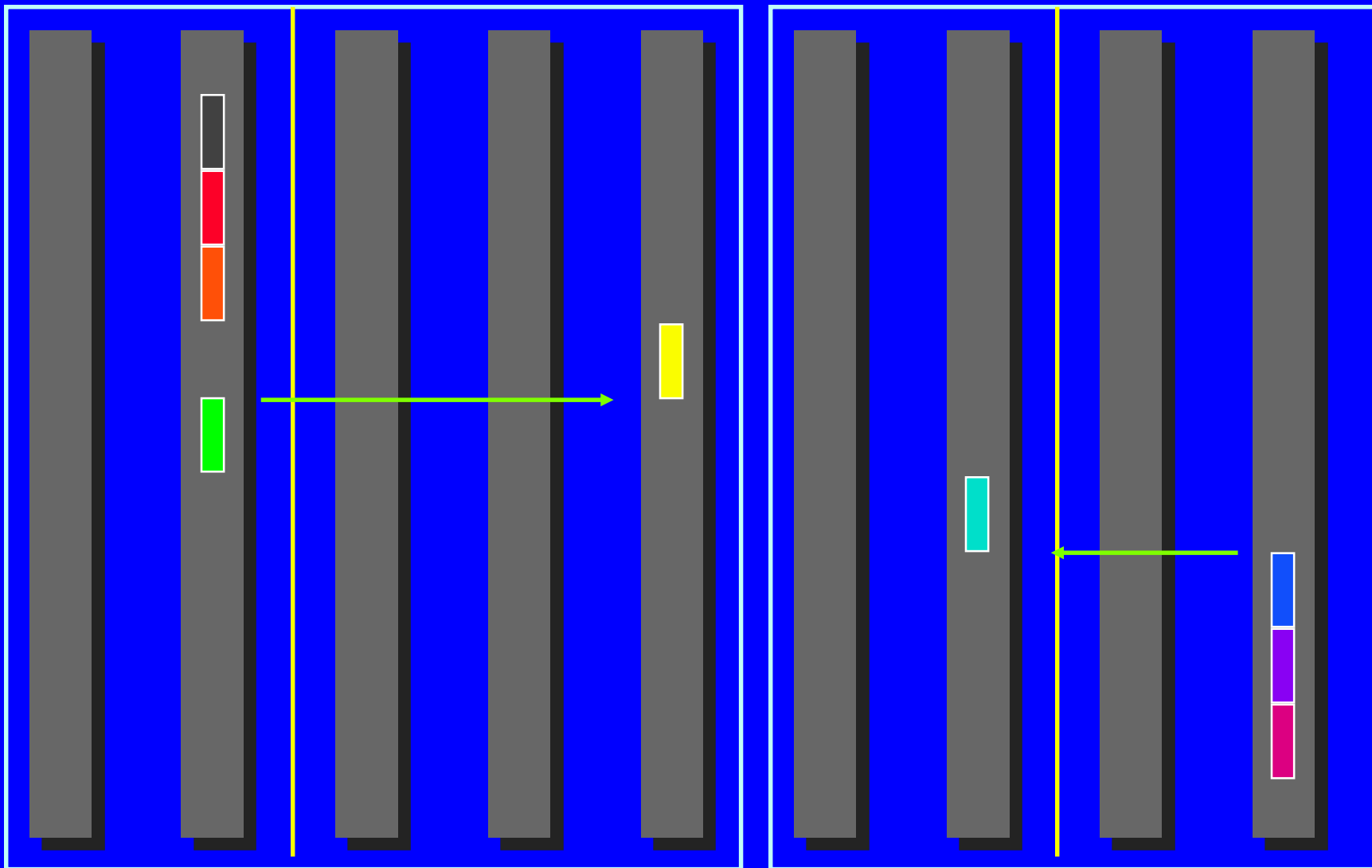


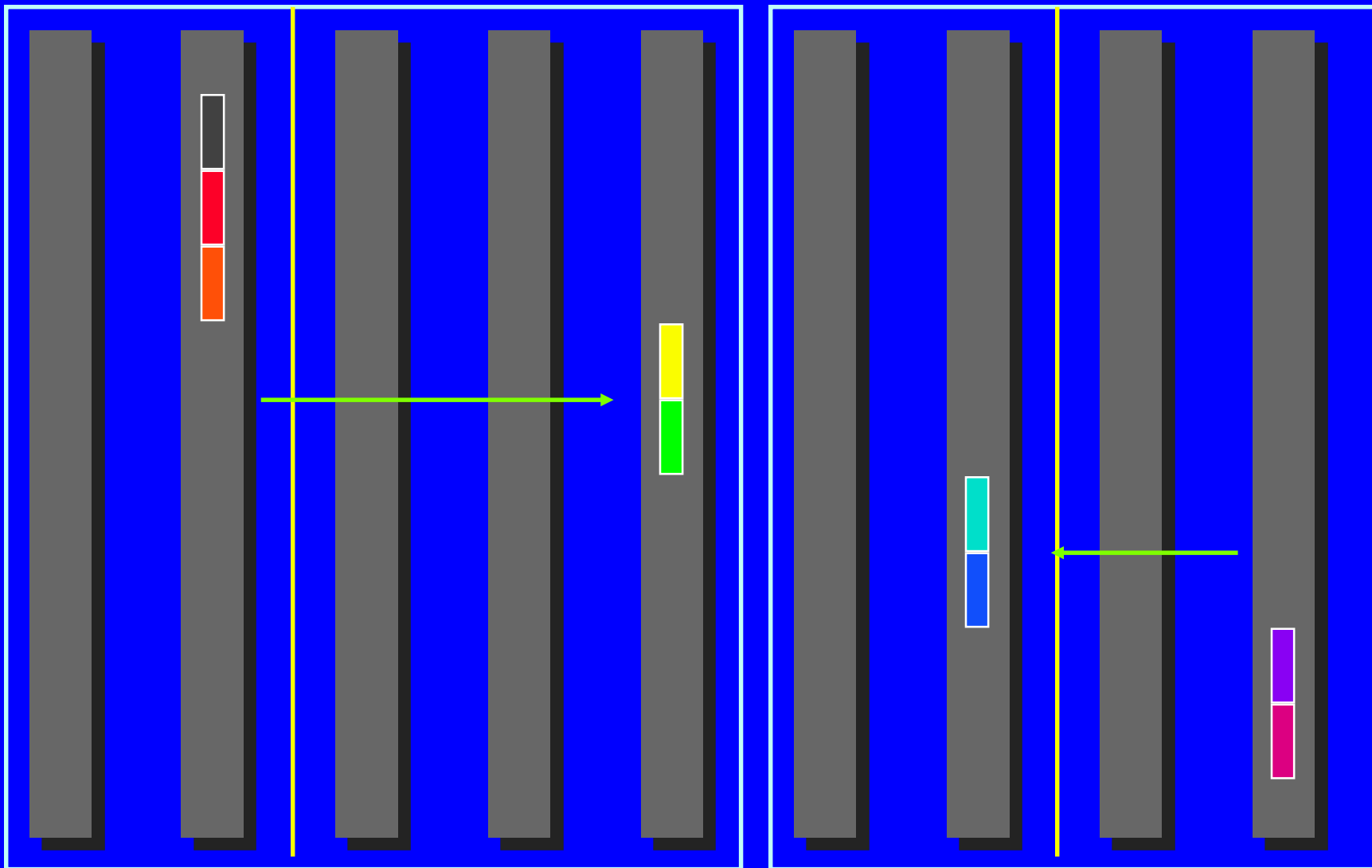


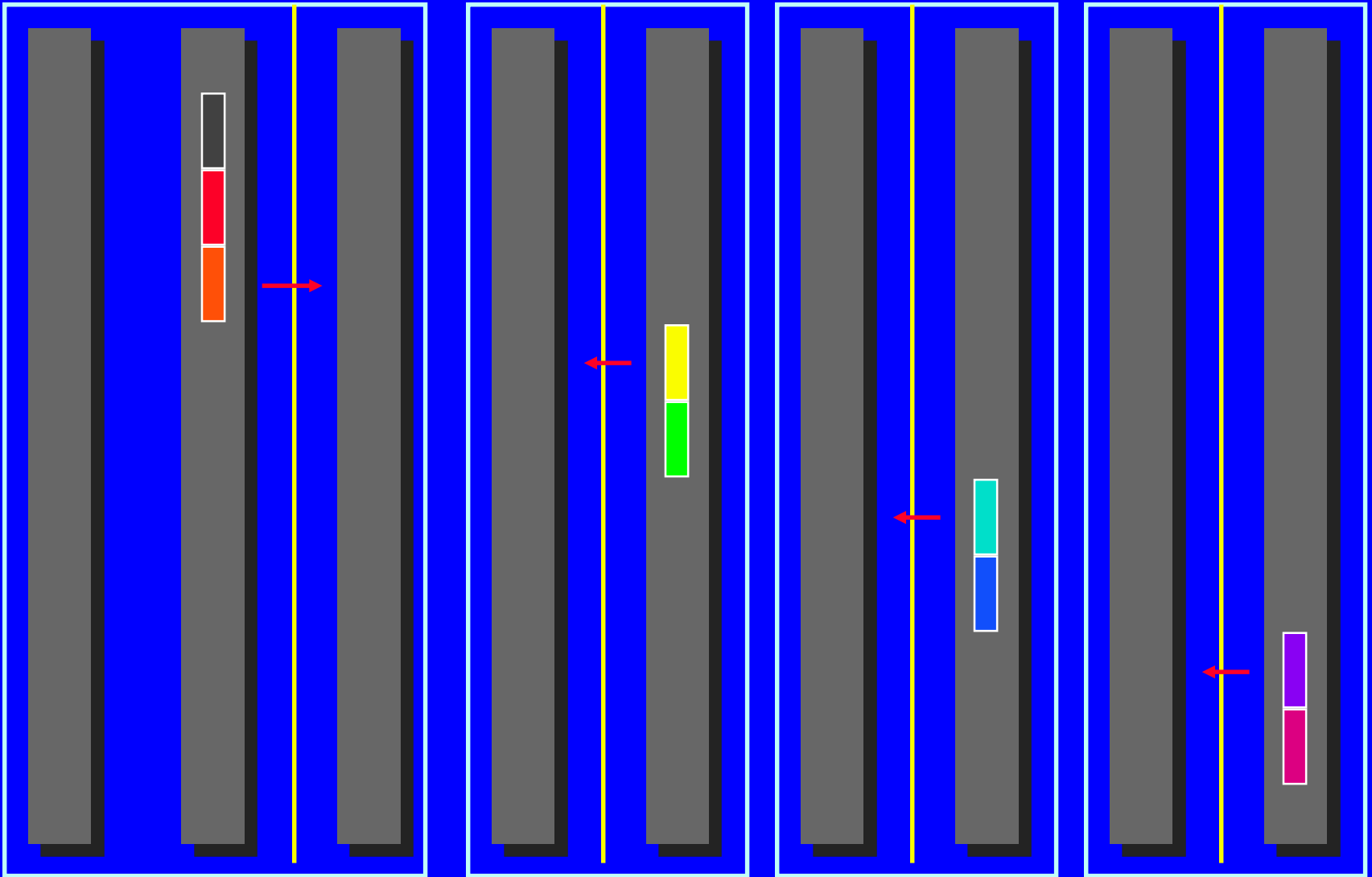


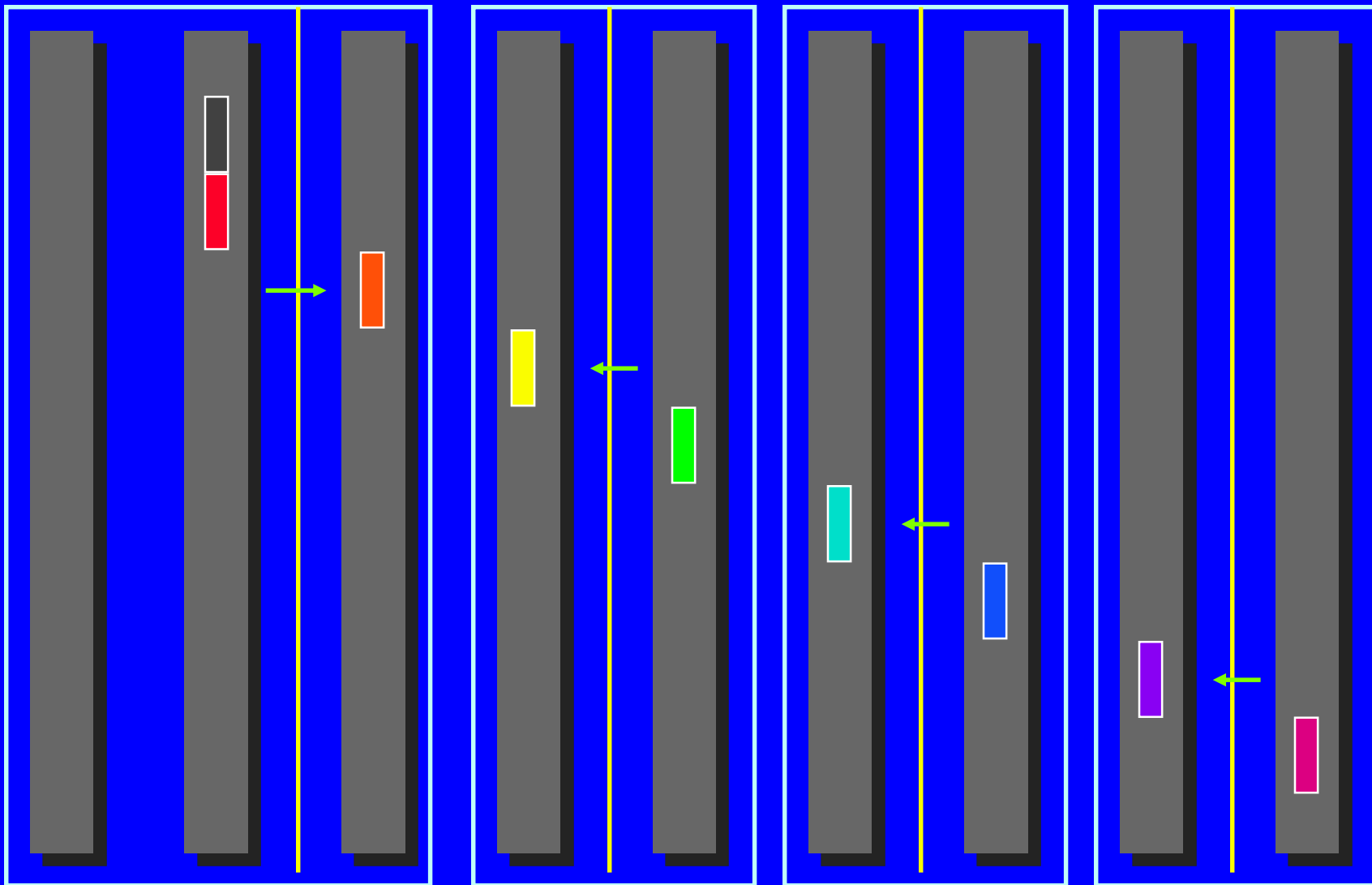


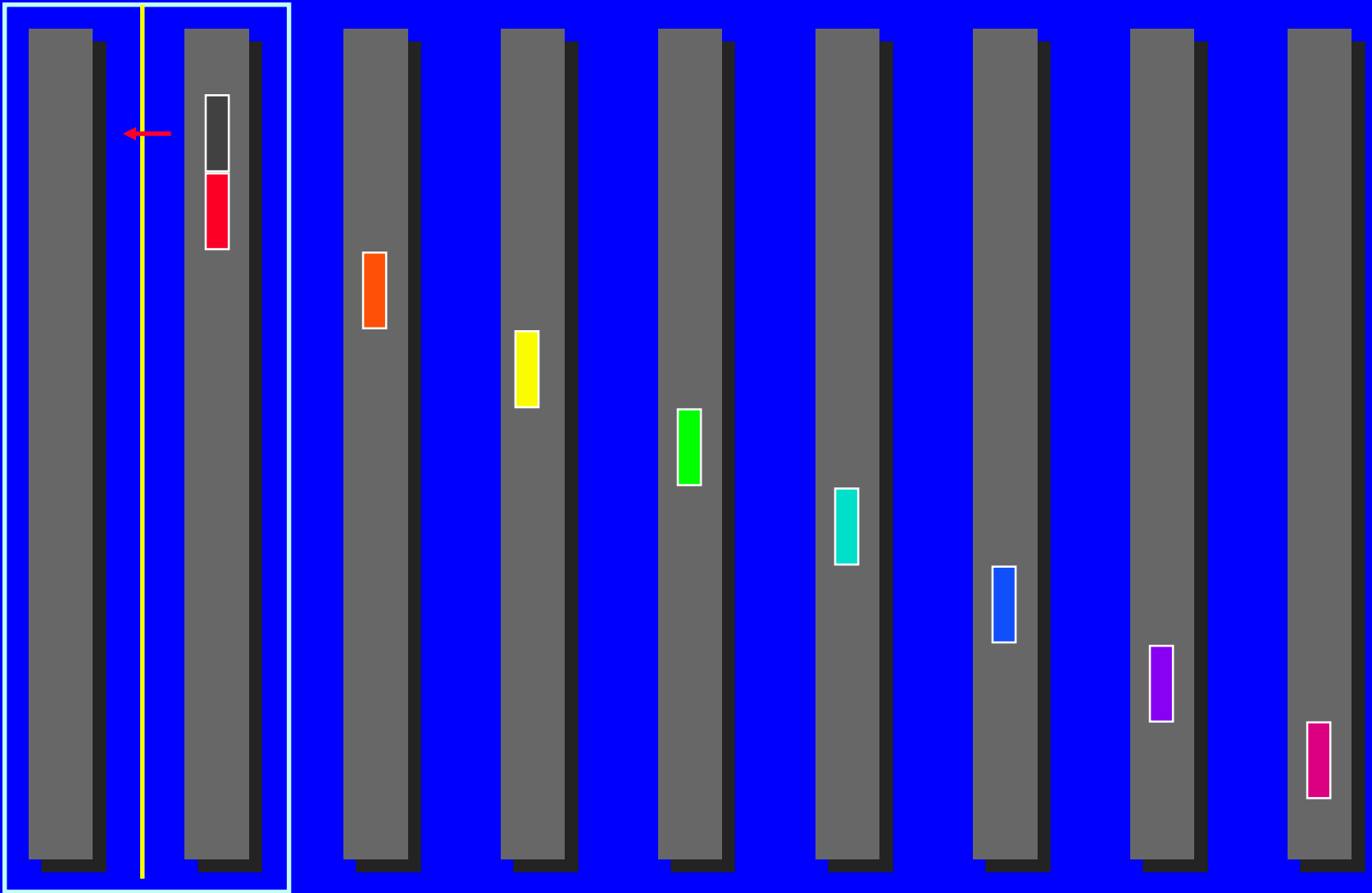


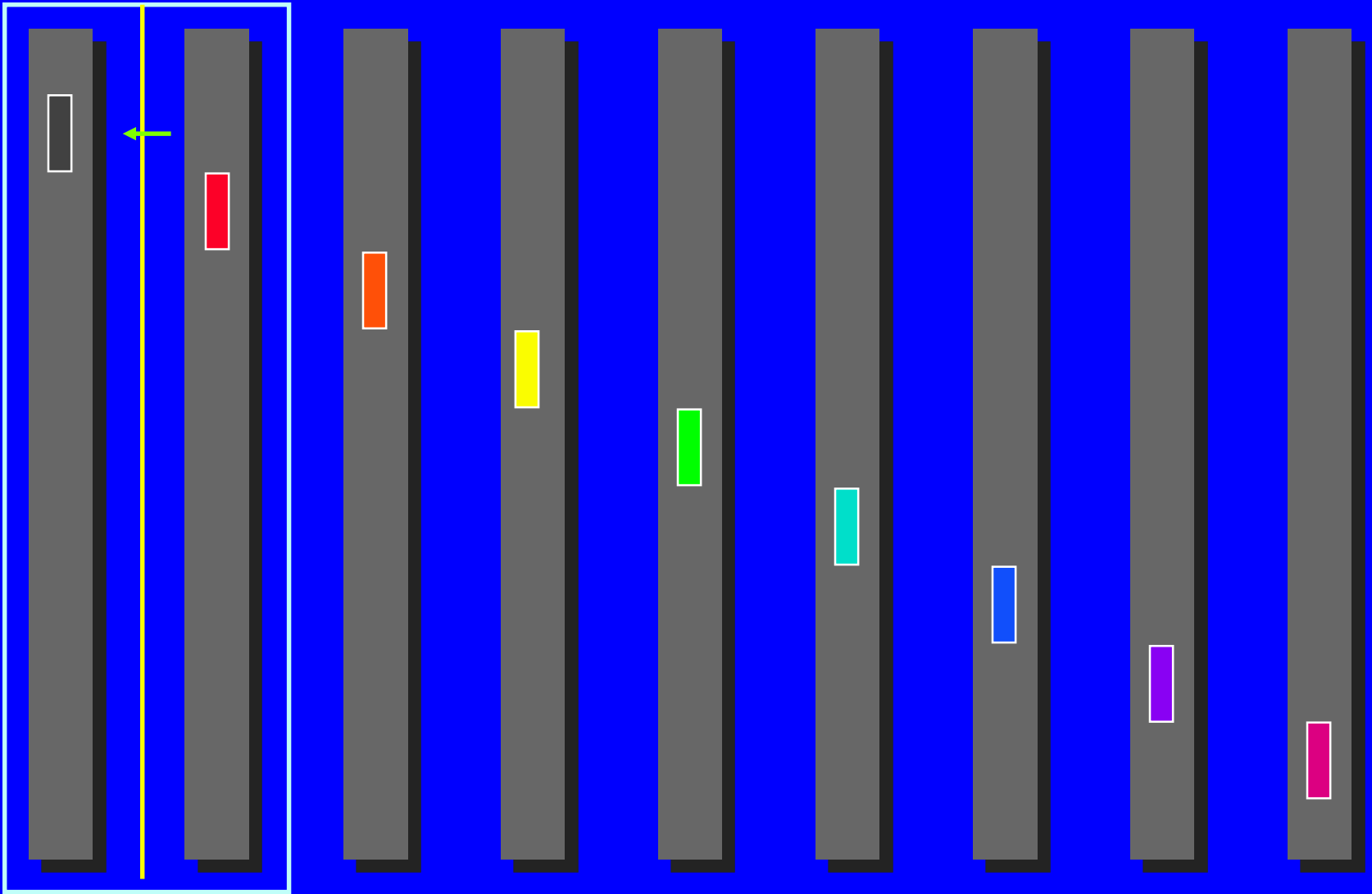


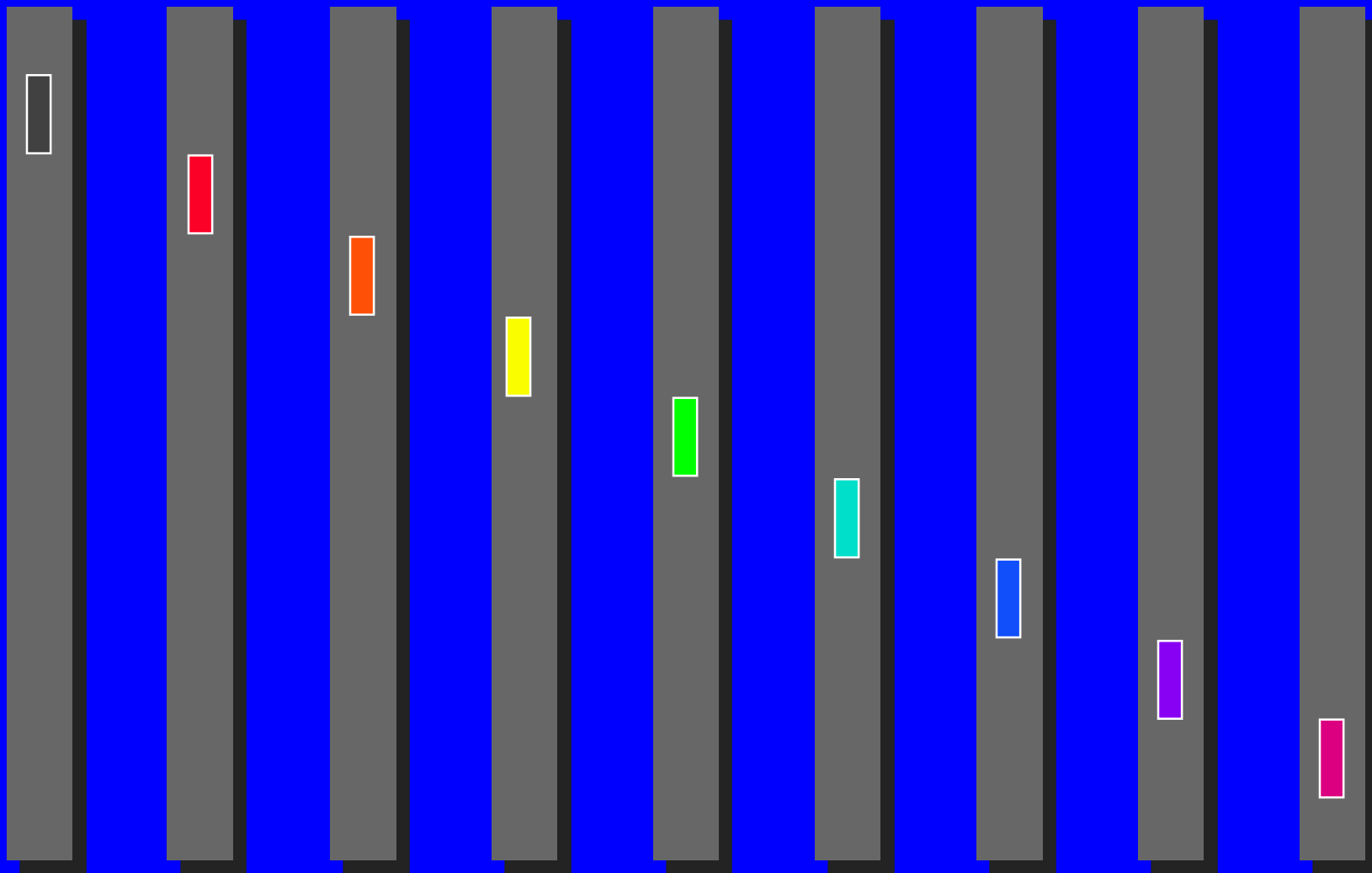












Cost of minimum spanning tree scatter

- Assumption: power of two number of nodes

$$\sum_{k=1}^{\log(p)} \left(\alpha + \frac{n}{2^k} \beta \right) = \log(p) \alpha + \frac{p-1}{p} n \beta$$

Cost of minimum spanning tree scatter

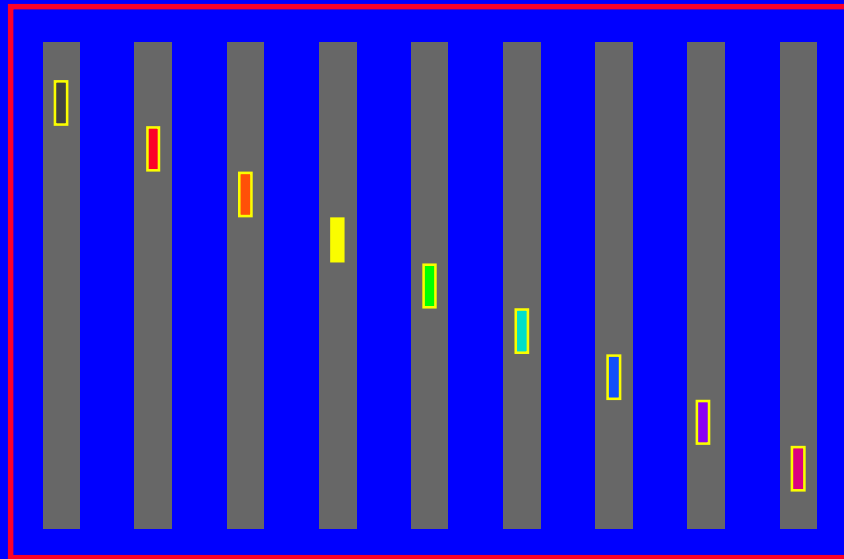
- Assumption: power of two number of nodes

$$\sum_{k=1}^{\log(p)} \left(\alpha + \frac{n}{2^k} \beta \right) = \log(p) \alpha + \frac{p-1}{p} n \beta$$

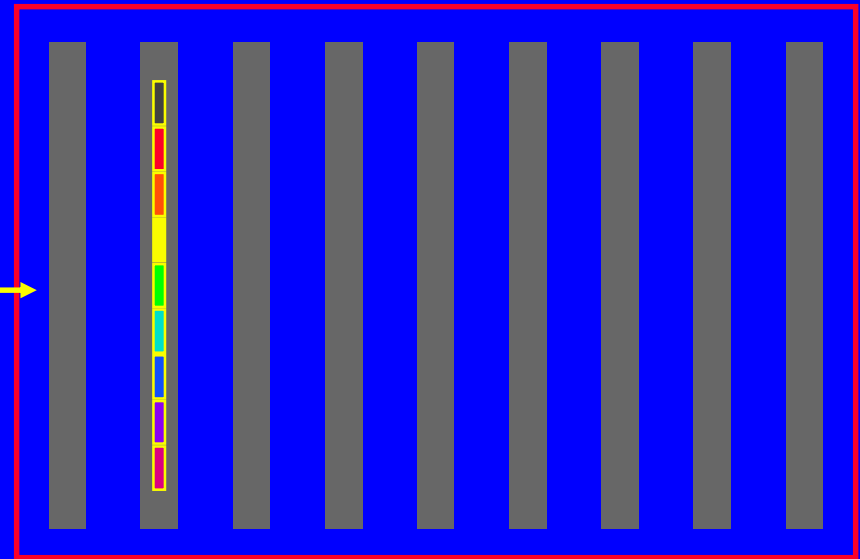
Notice: attains lower bound for latency and bandwidth components

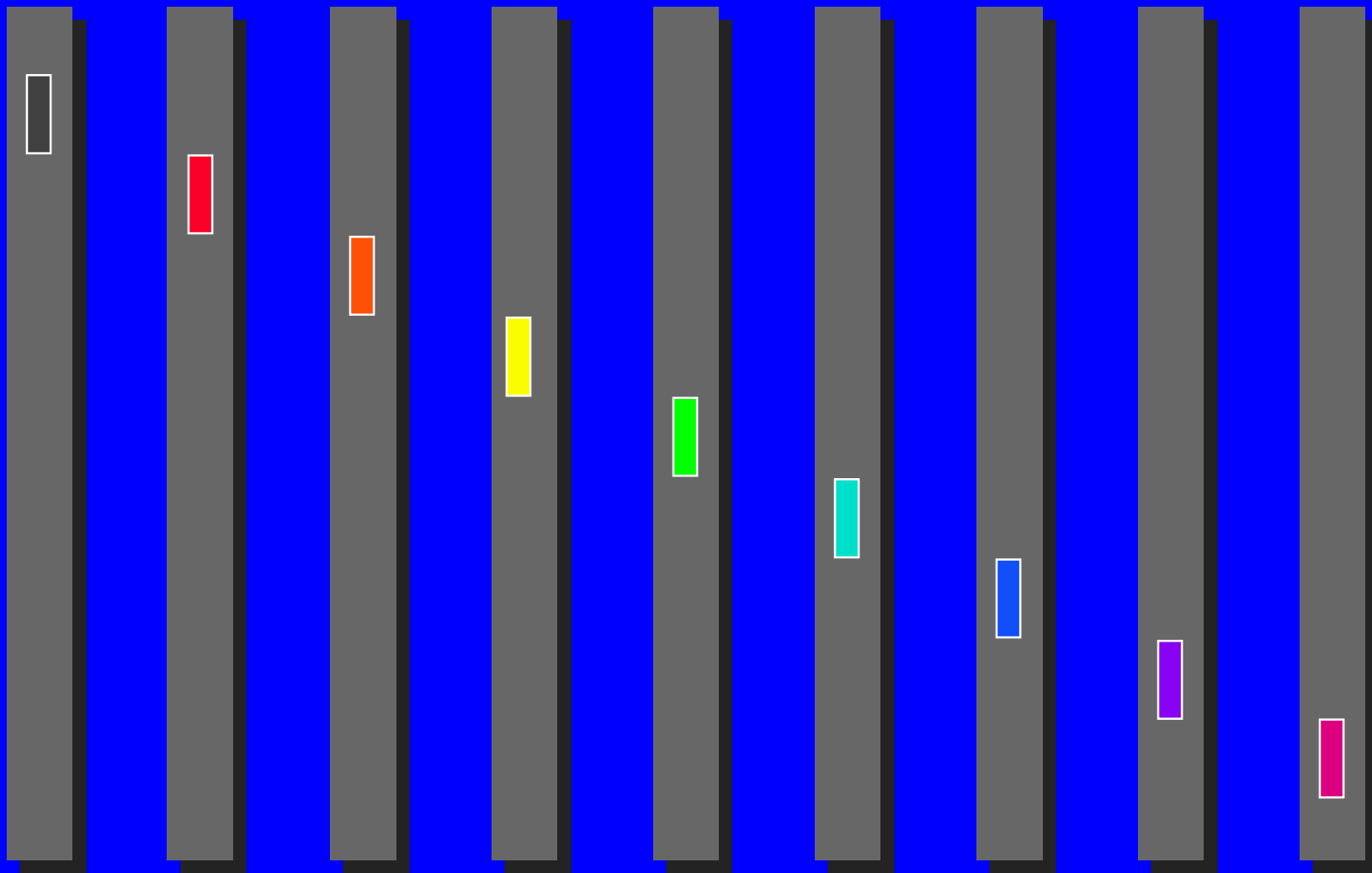
Gather

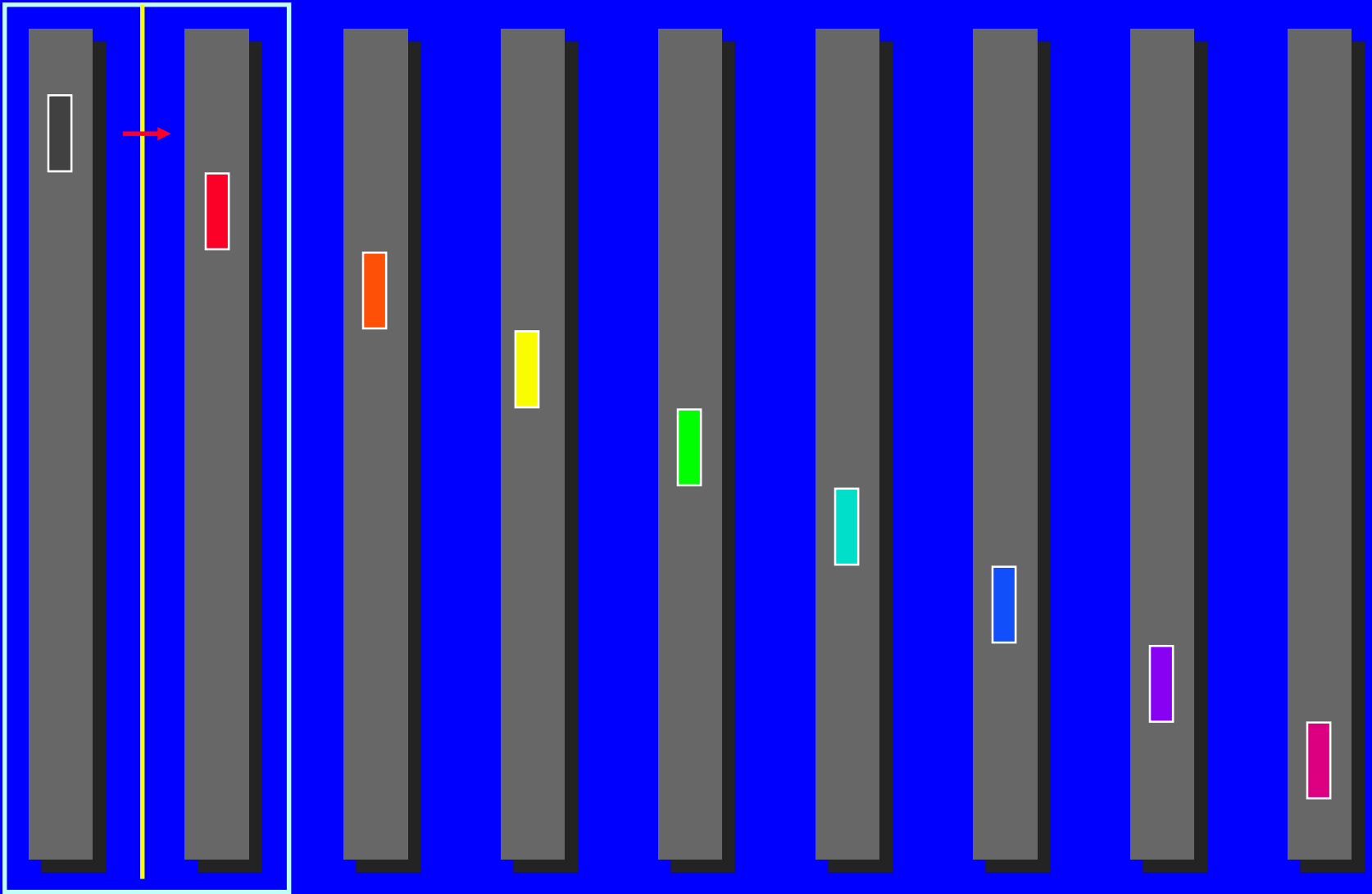
Before

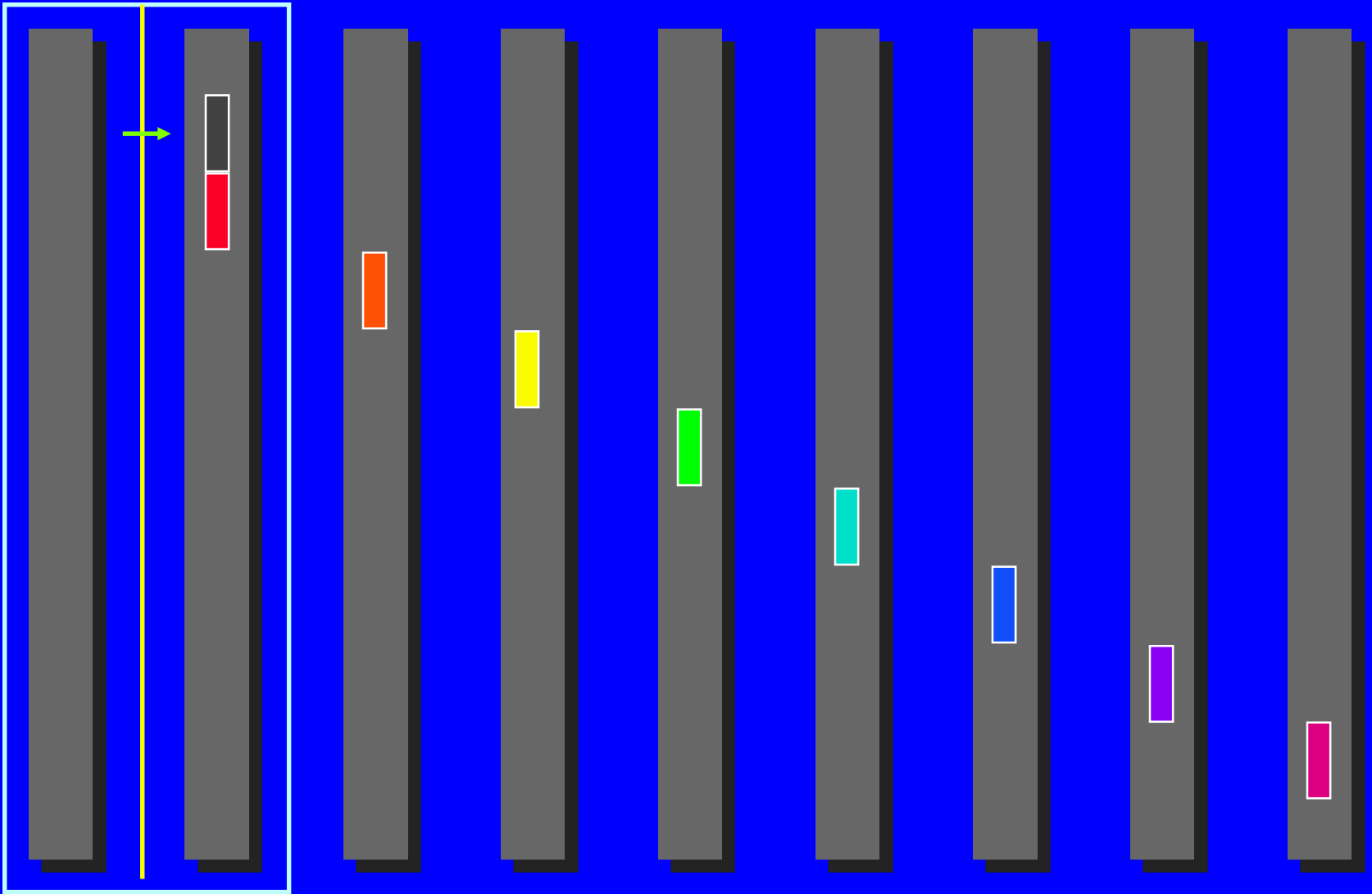


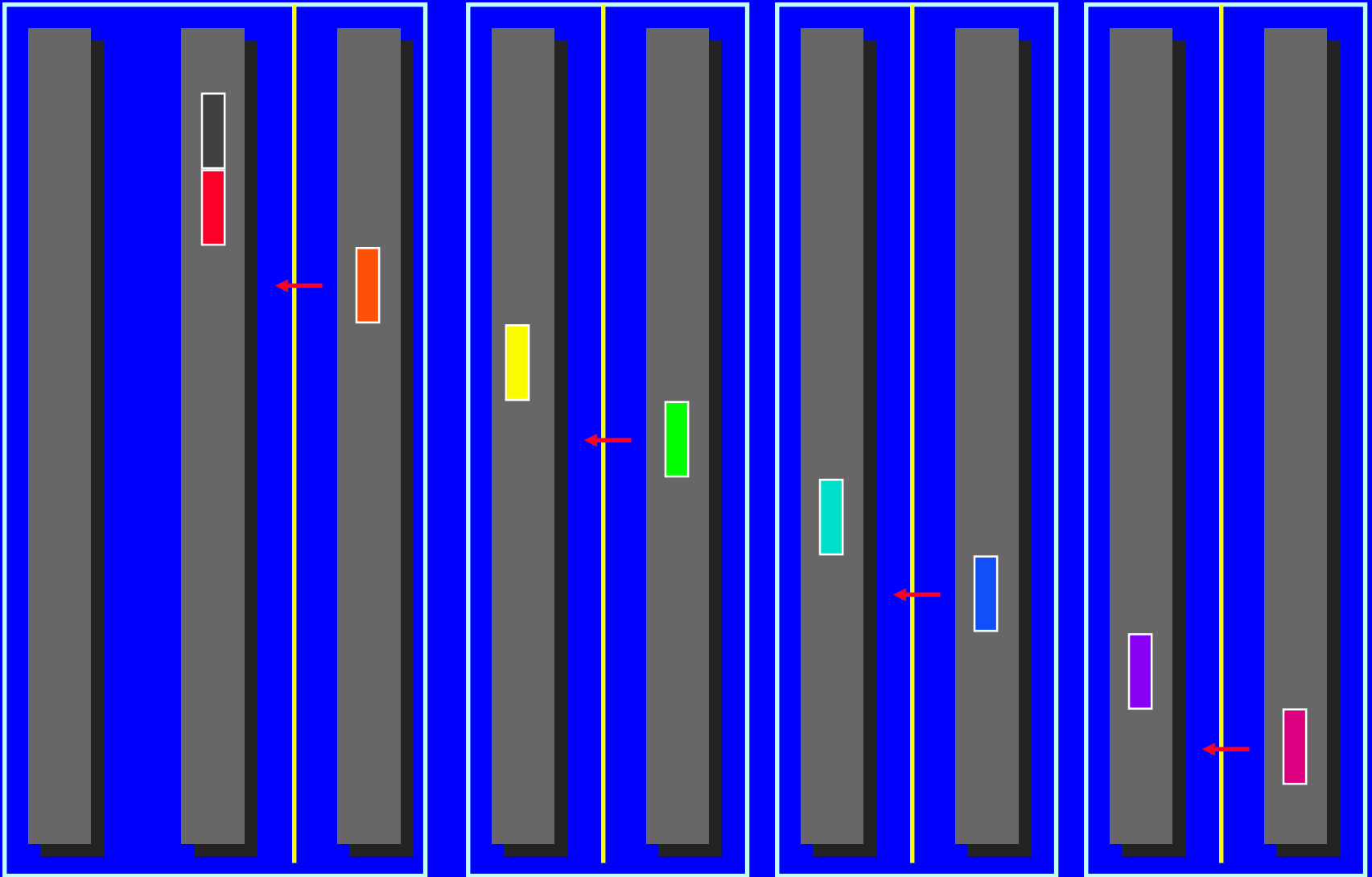
After

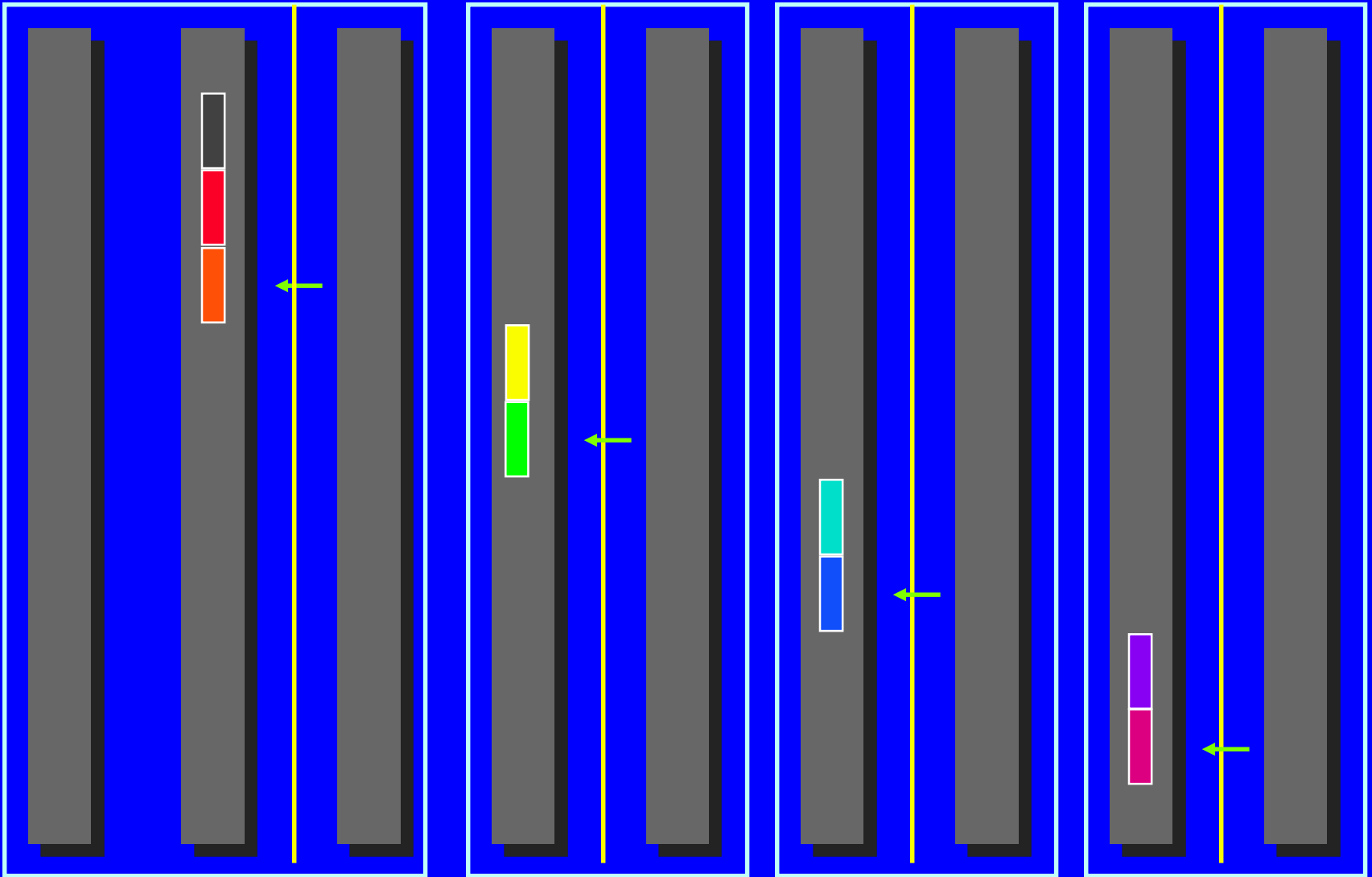


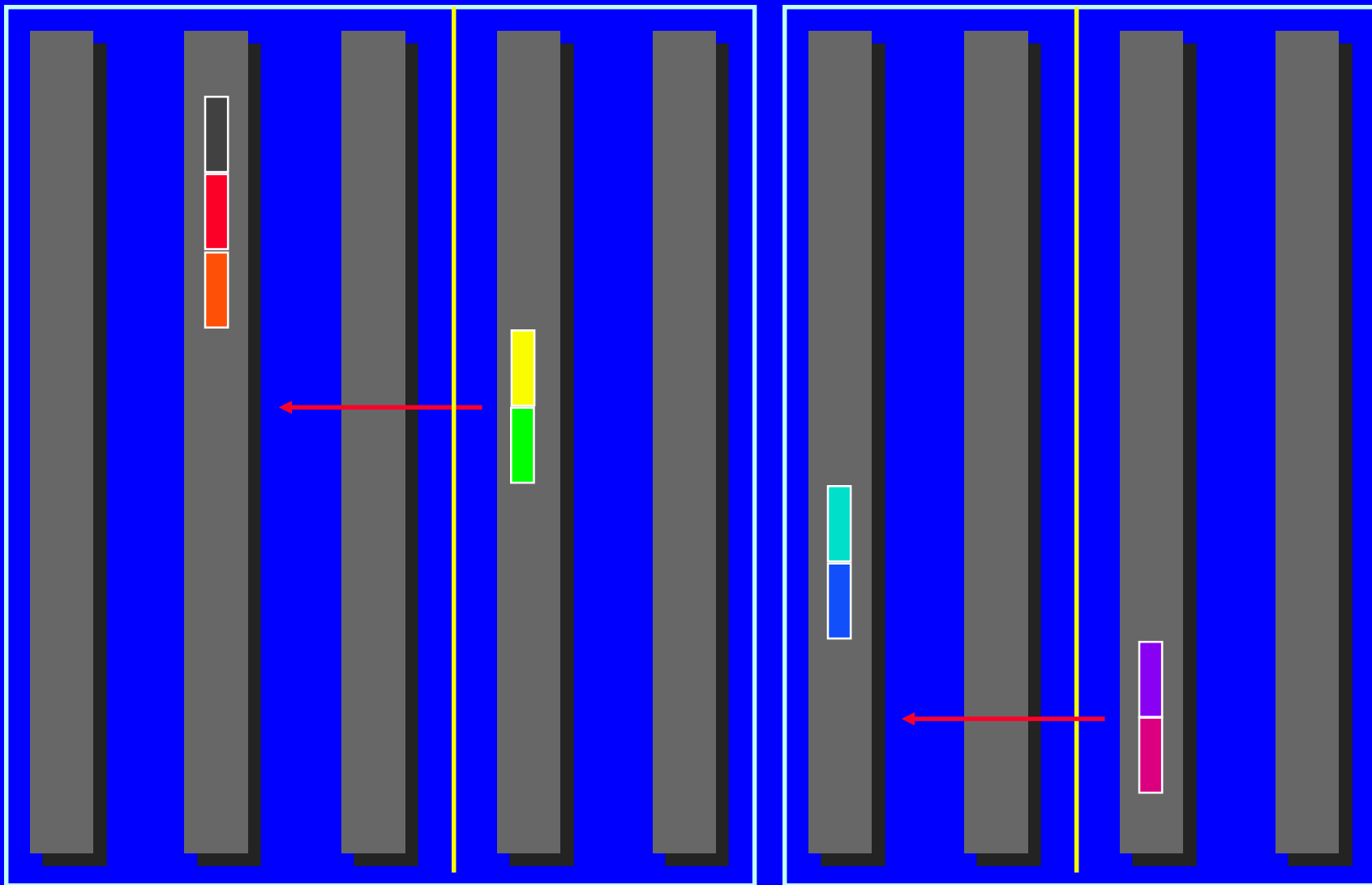


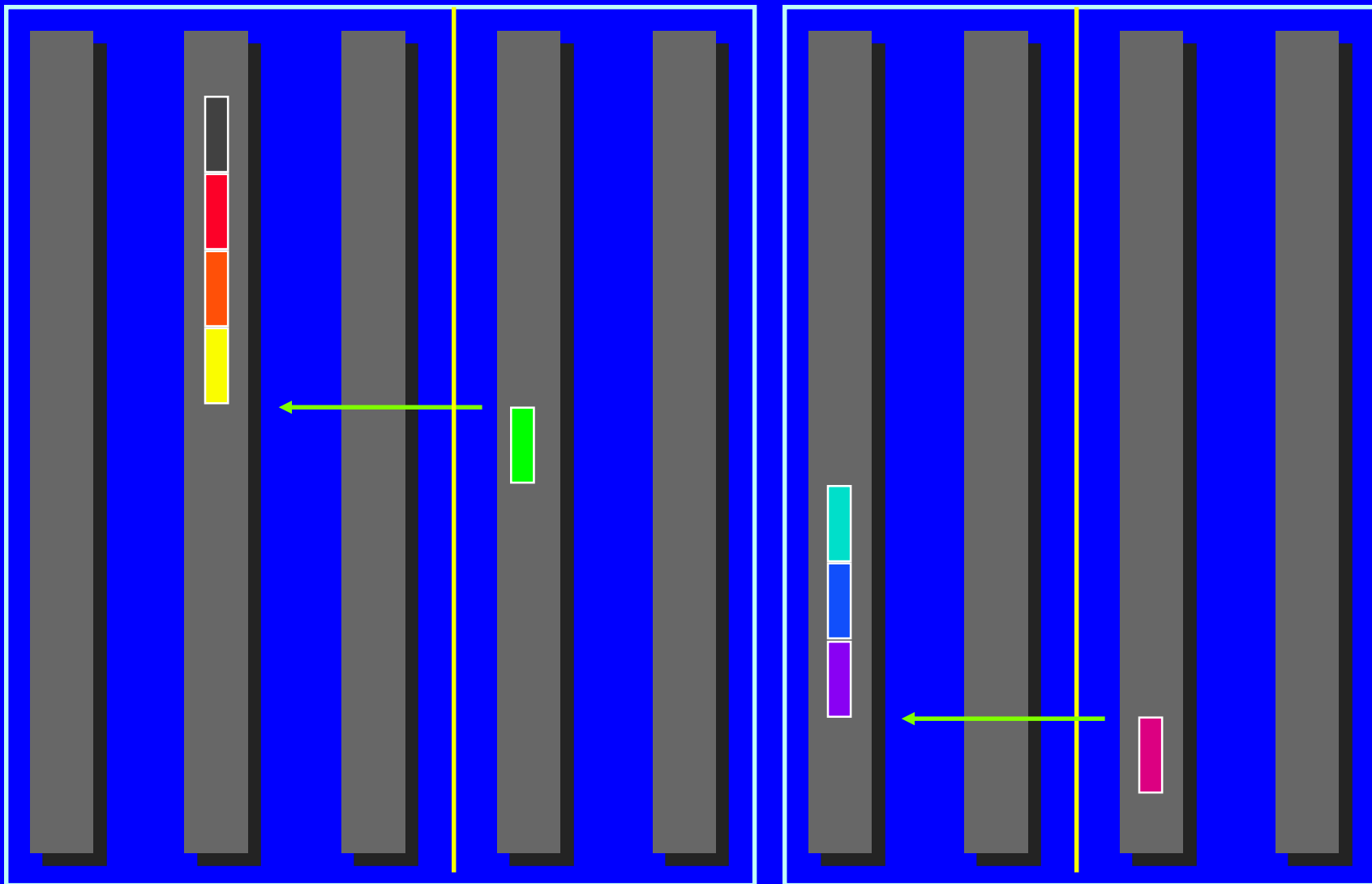


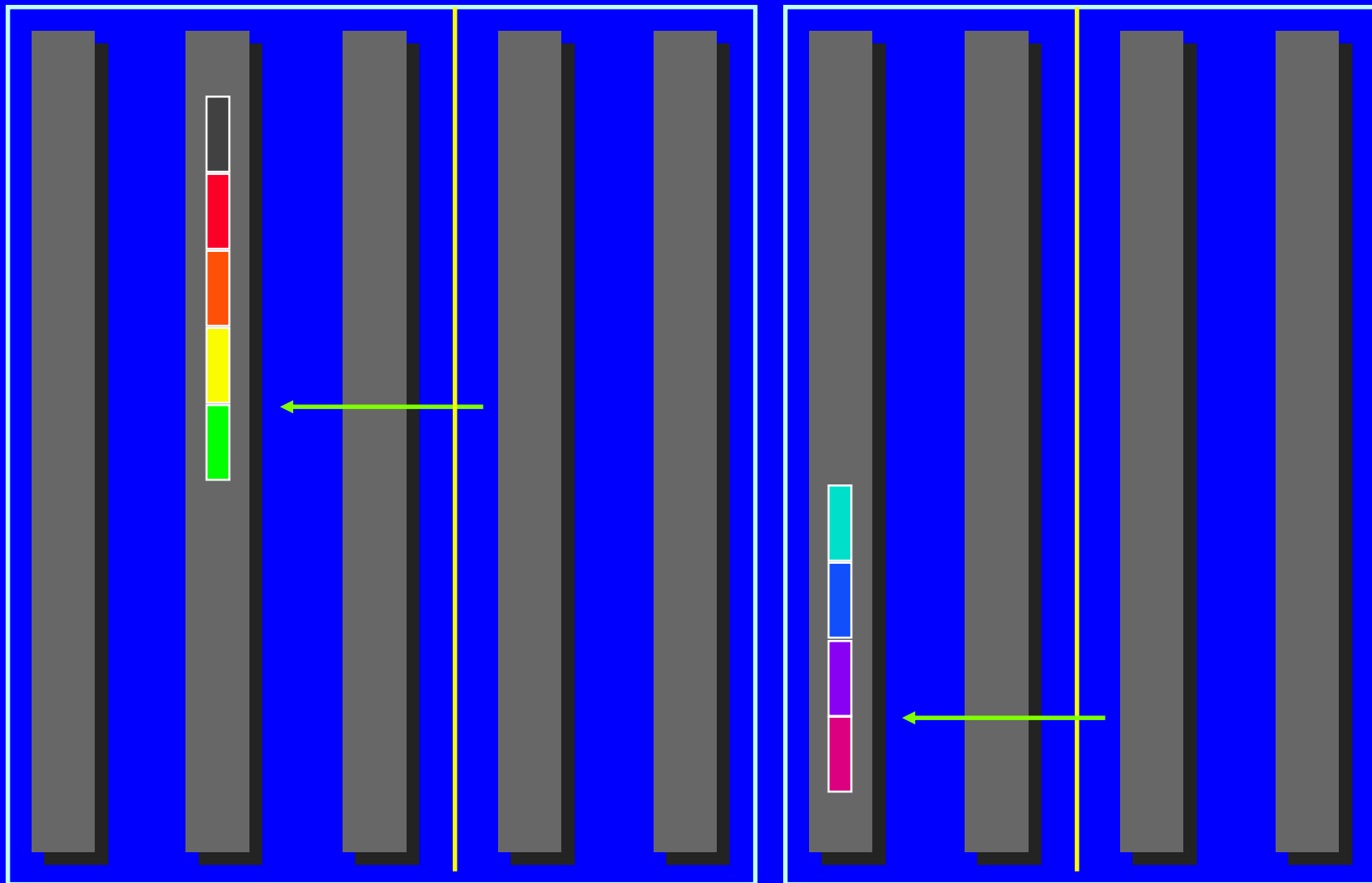


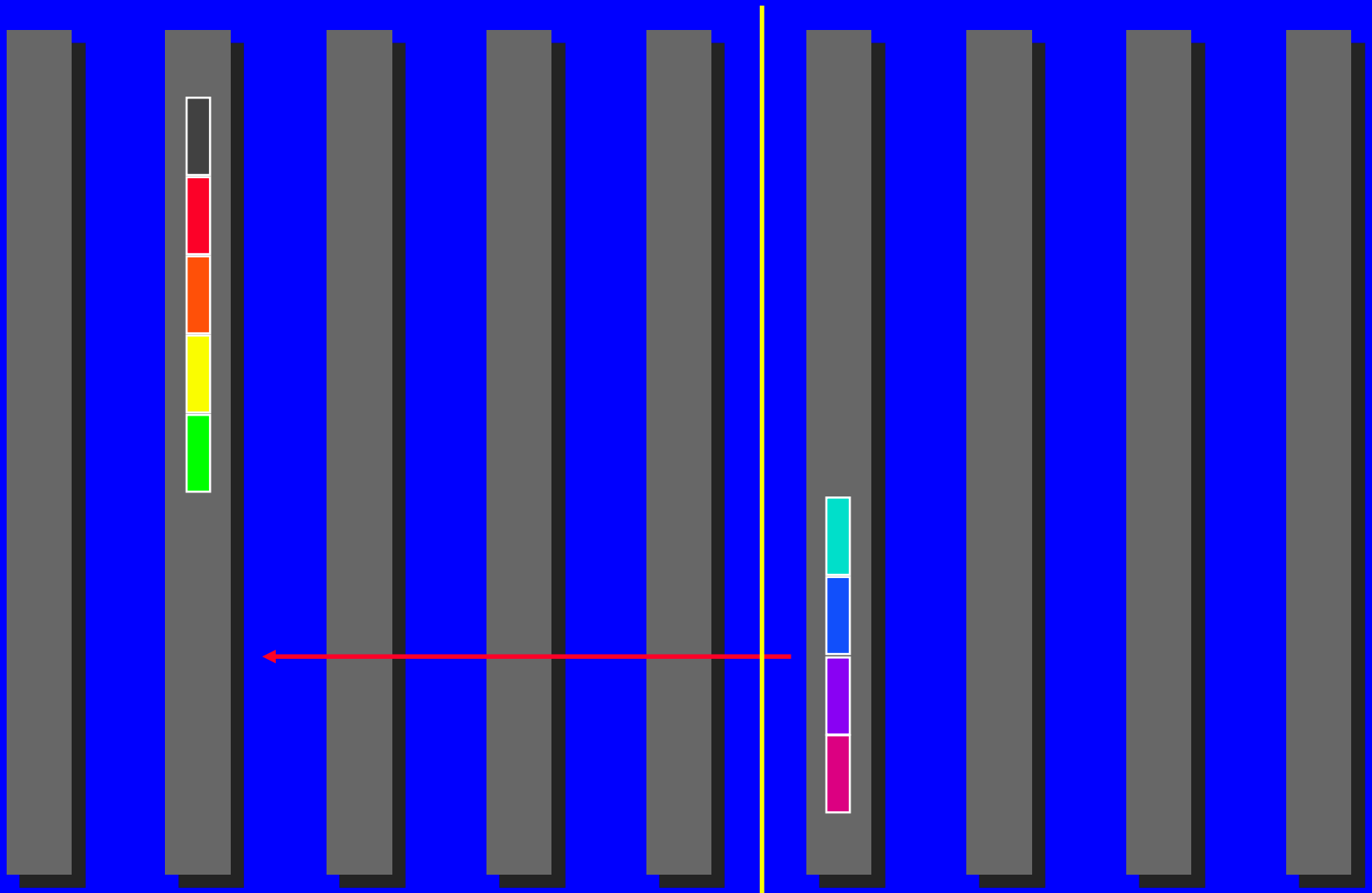


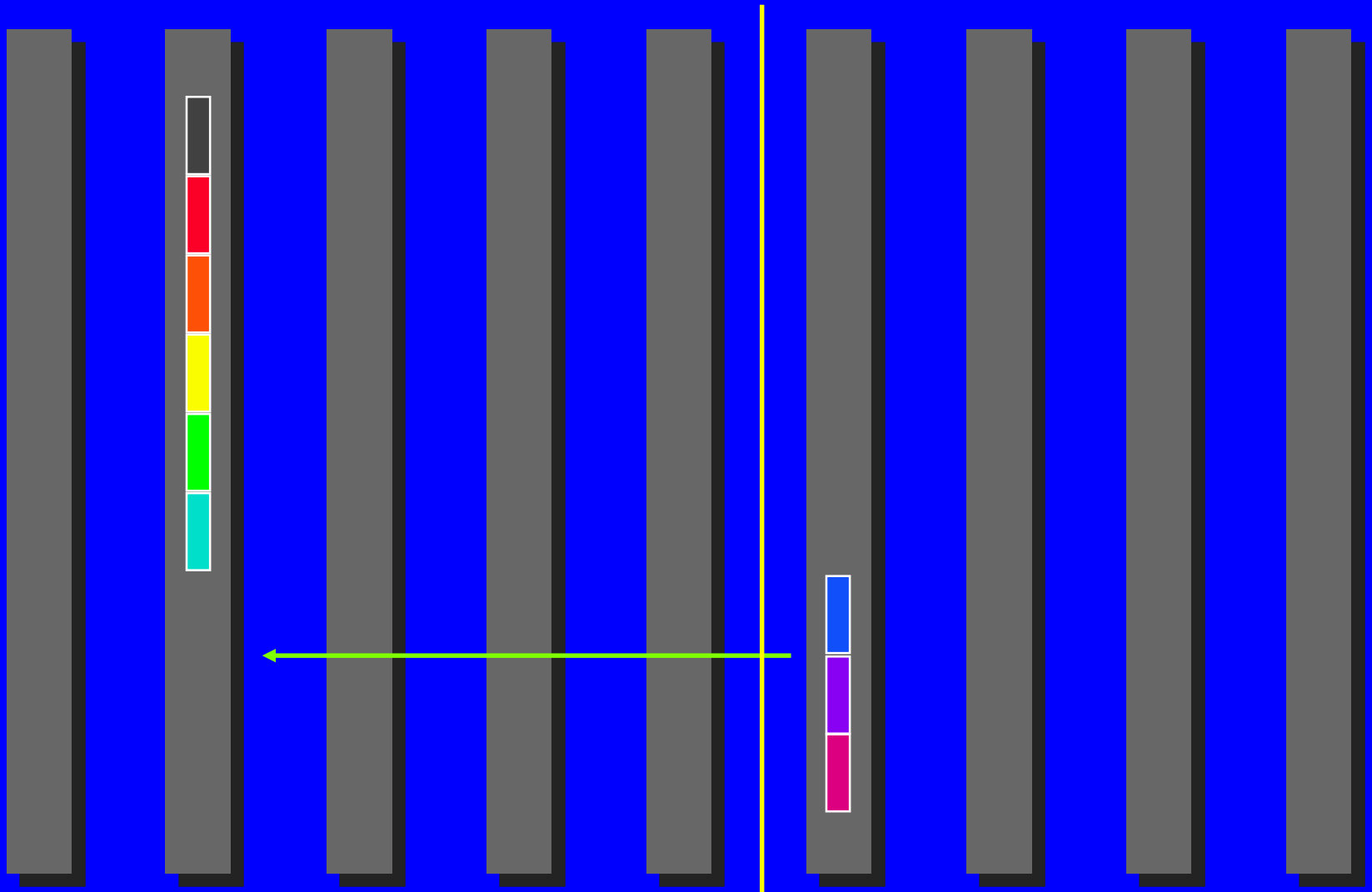


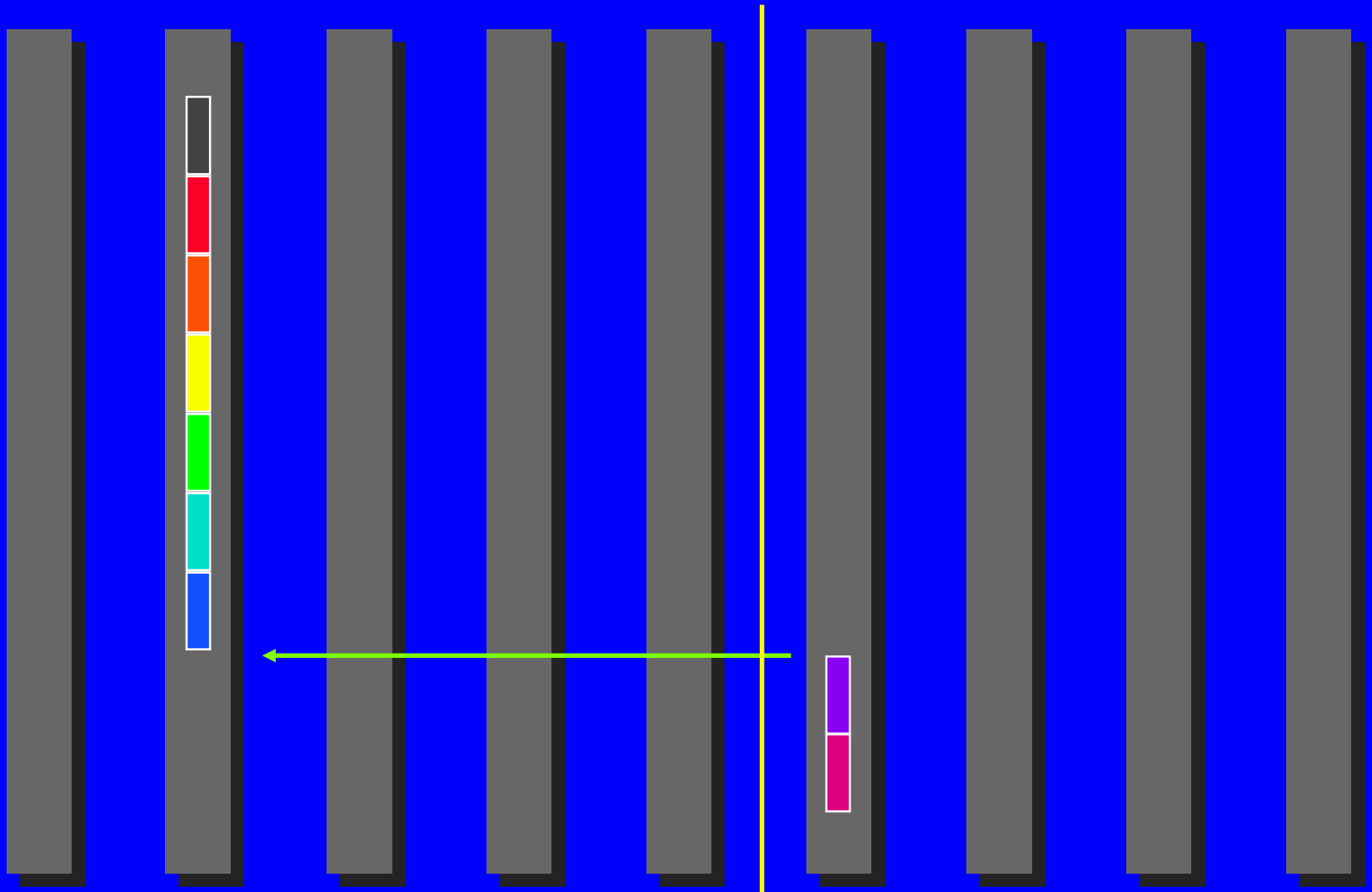


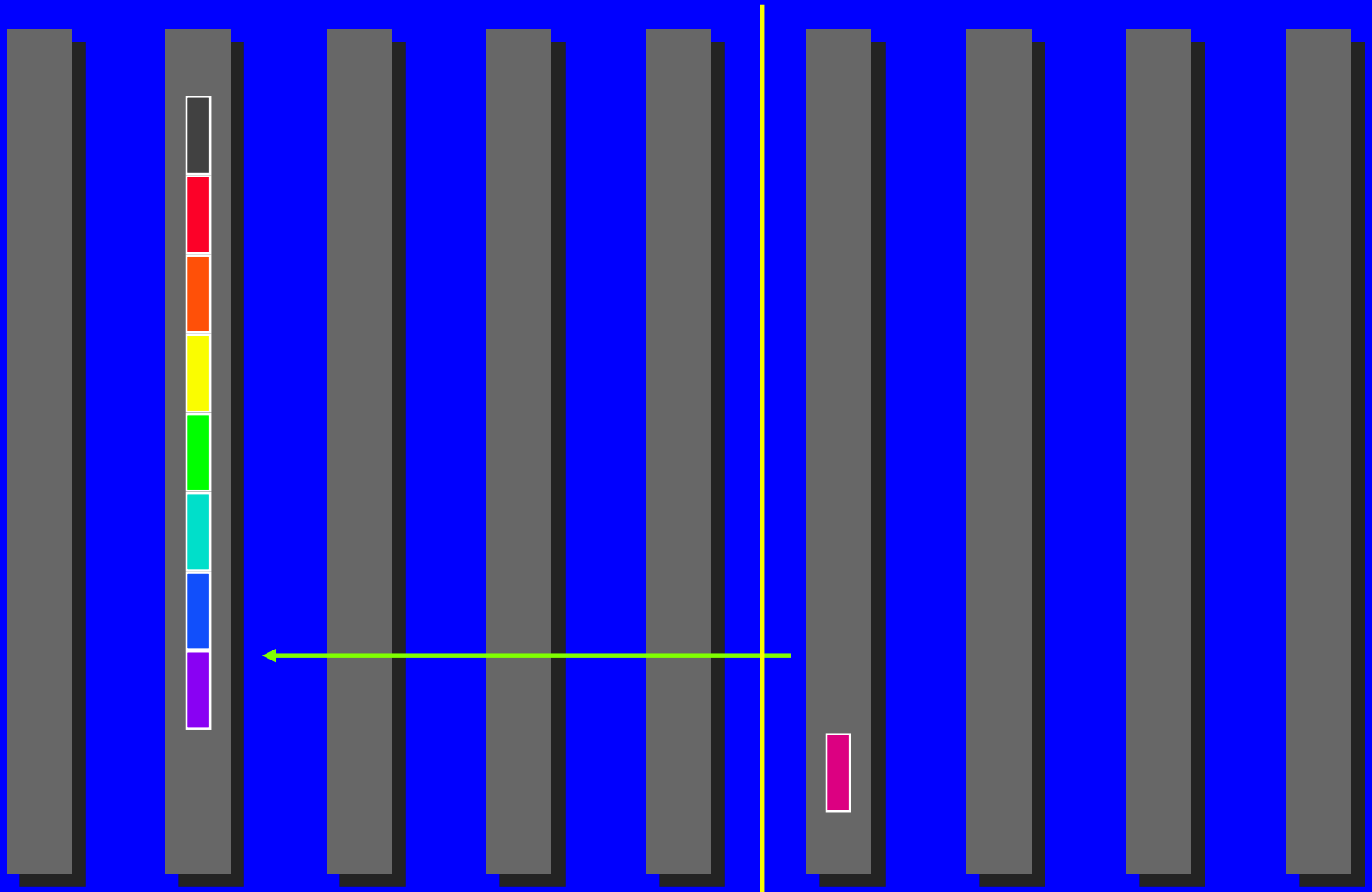


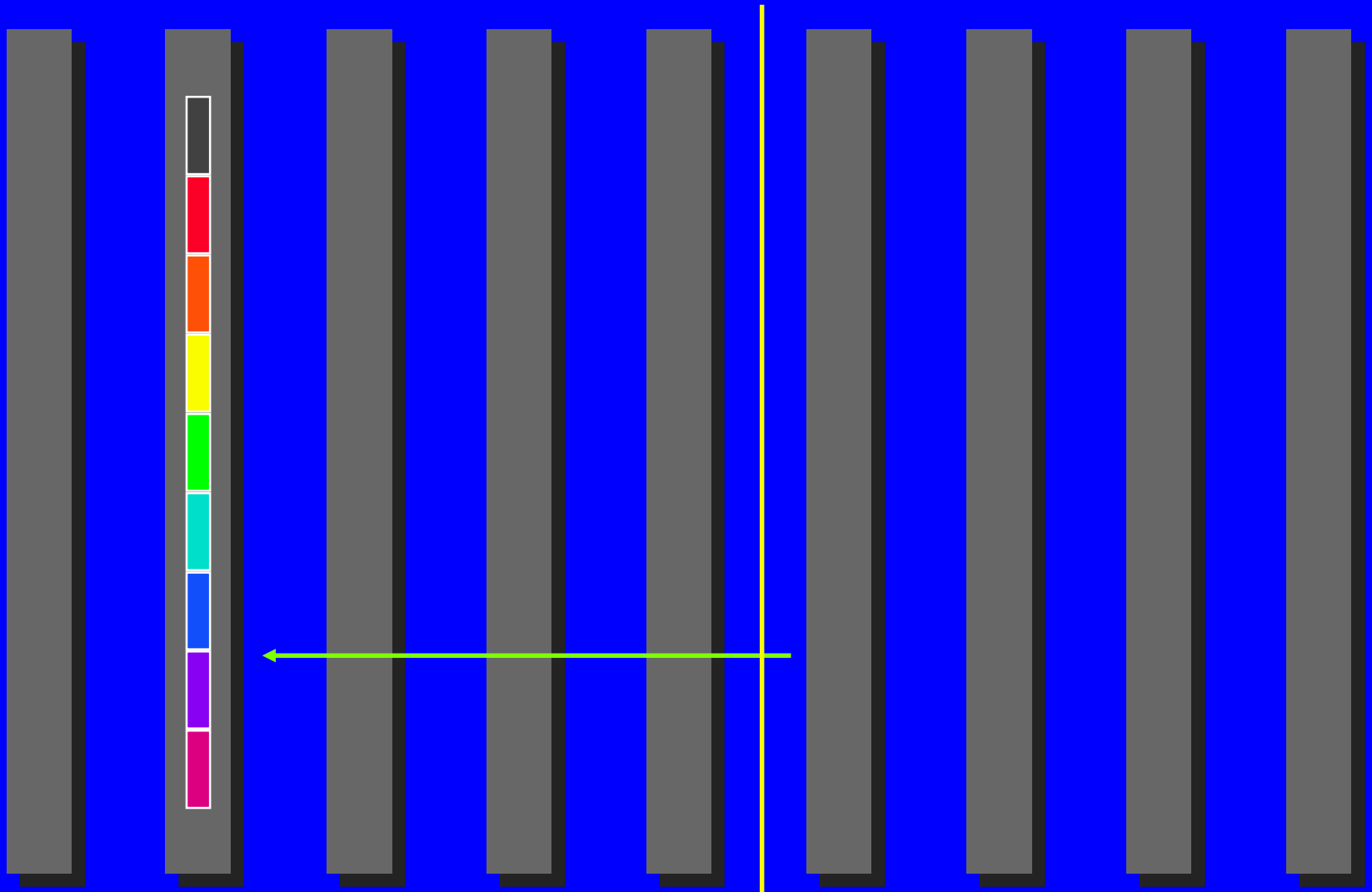


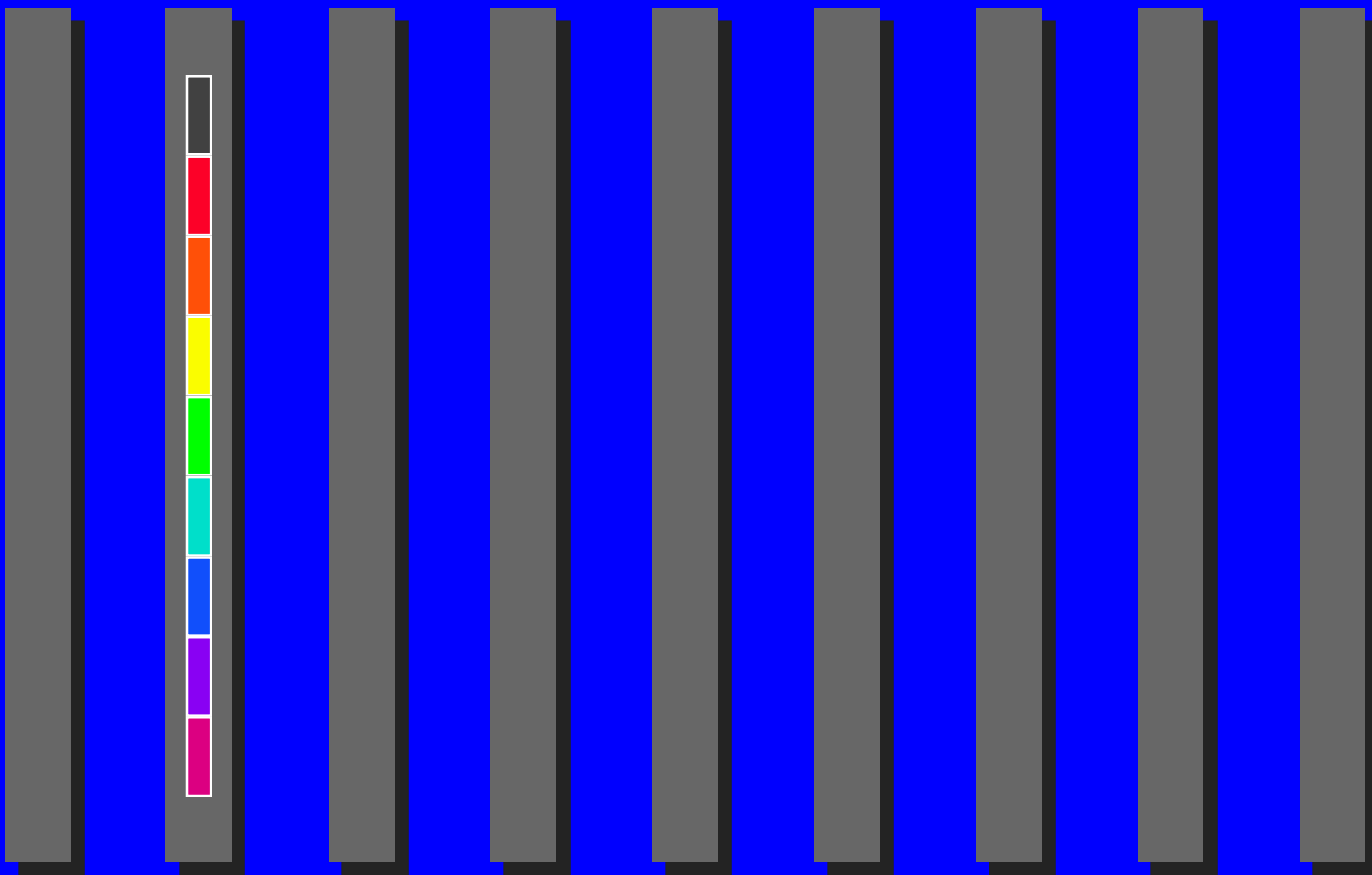












Cost of minimum spanning tree gather

- Assumption: power of two number of nodes

$$\sum_{k=1}^{\log(p)} \left(\alpha + \frac{n}{2^k} \beta \right) = \log(p) \alpha + \frac{p-1}{p} n \beta$$

Cost of minimum spanning tree gather

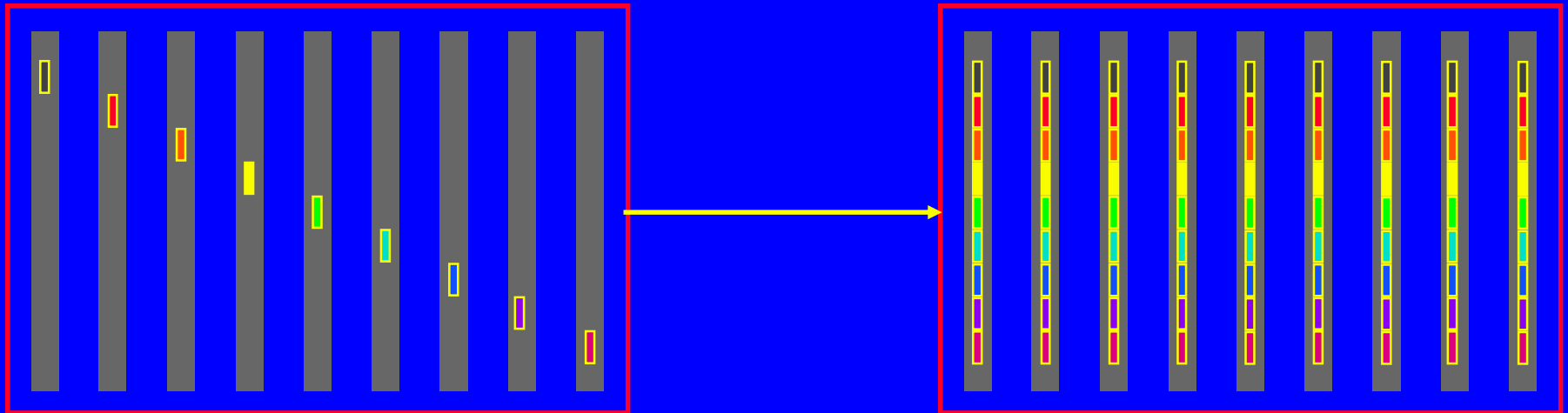
- Assumption: power of two number of nodes

$$\sum_{k=1}^{\log(p)} \left(\alpha + \frac{n}{2^k} \beta \right) = \log(p) \alpha + \frac{p-1}{p} n \beta$$

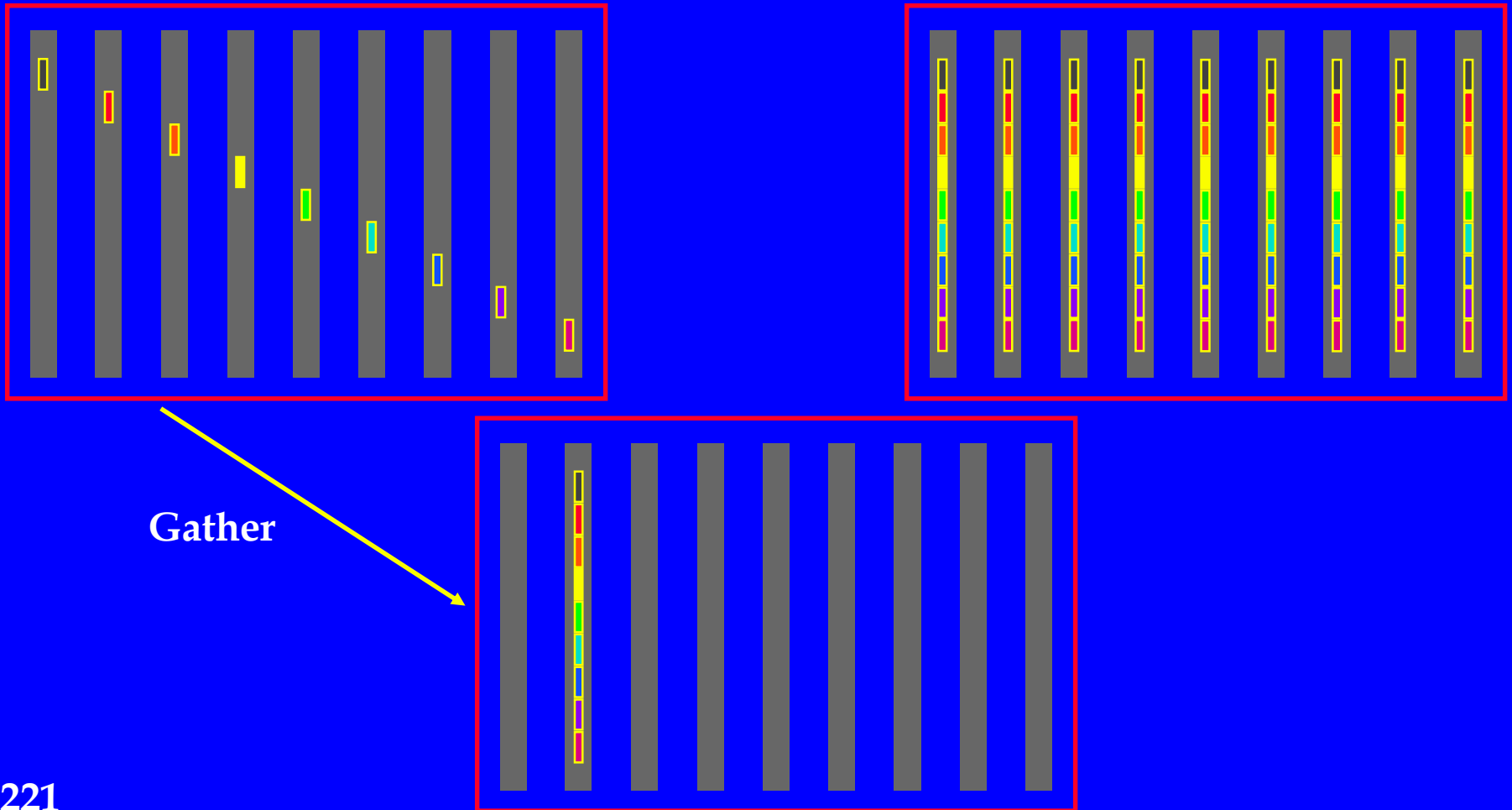
Notice: attains lower bound for latency and bandwidth components

Using the building blocks

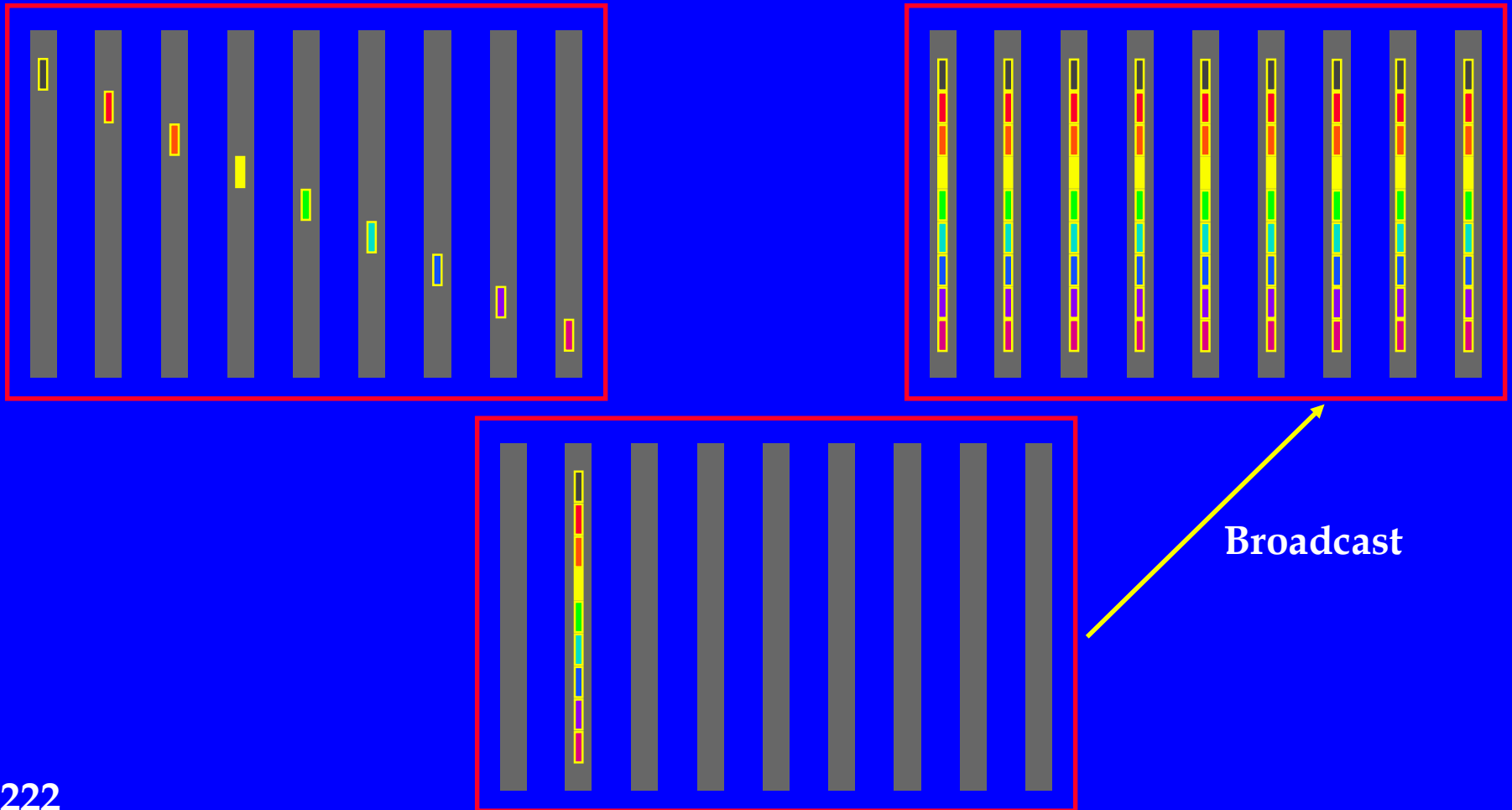
Allgather (short vector)



Allgather (short vector)



Allgather (short vector)



Cost of gather/broadcast allgather

- Assumption: power of two number of nodes

gather

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

broadcast

$$\log(p)(\alpha + n\beta)$$

$$\frac{\log(p)(\alpha + n\beta)}{2\log(p)\alpha + \left(\frac{p-1}{p} + \log(p)\right)n\beta}$$

Cost of gather/broadcast allgather

- Assumption: power of two number of nodes

gather

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

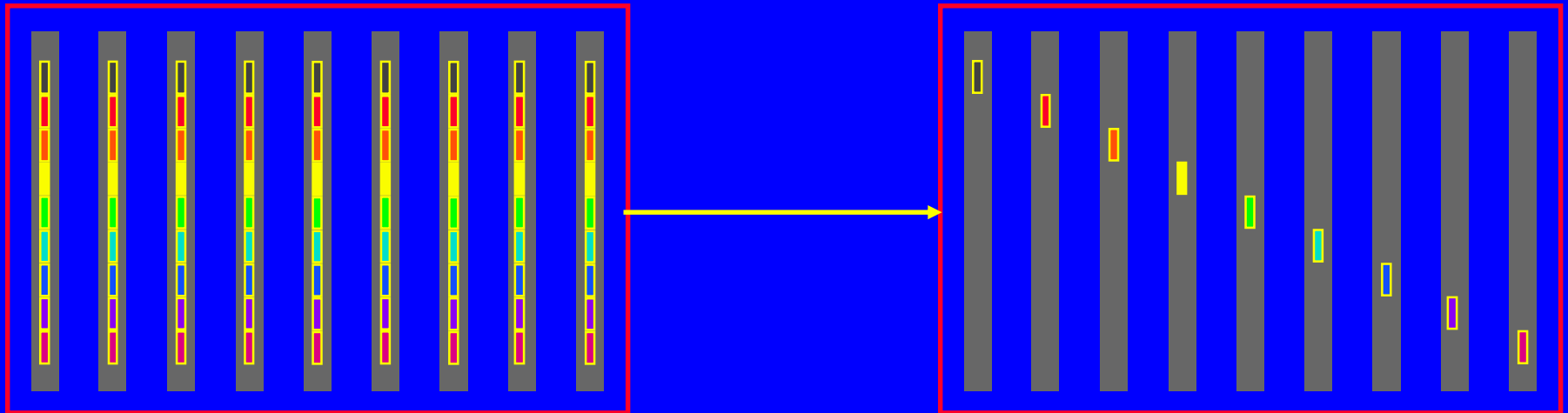
broadcast

$$\log(p)(\alpha + n\beta)$$

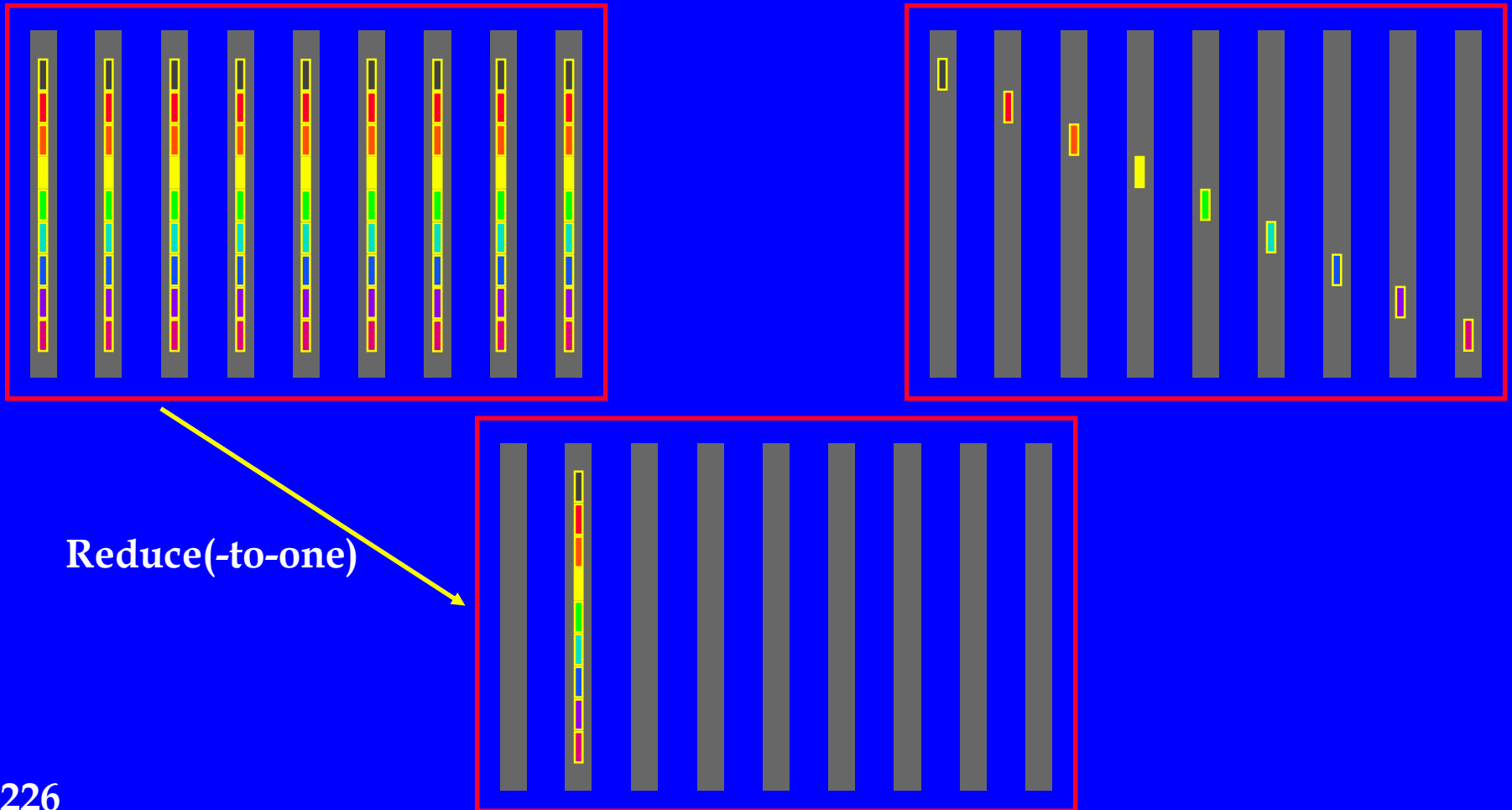
$$\frac{\log(p)(\alpha + n\beta)}{2\log(p)\alpha + \left(\frac{p-1}{p} + \log(p)\right)n\beta}$$

Notice: does not attain lower bound for latency or bandwidth components

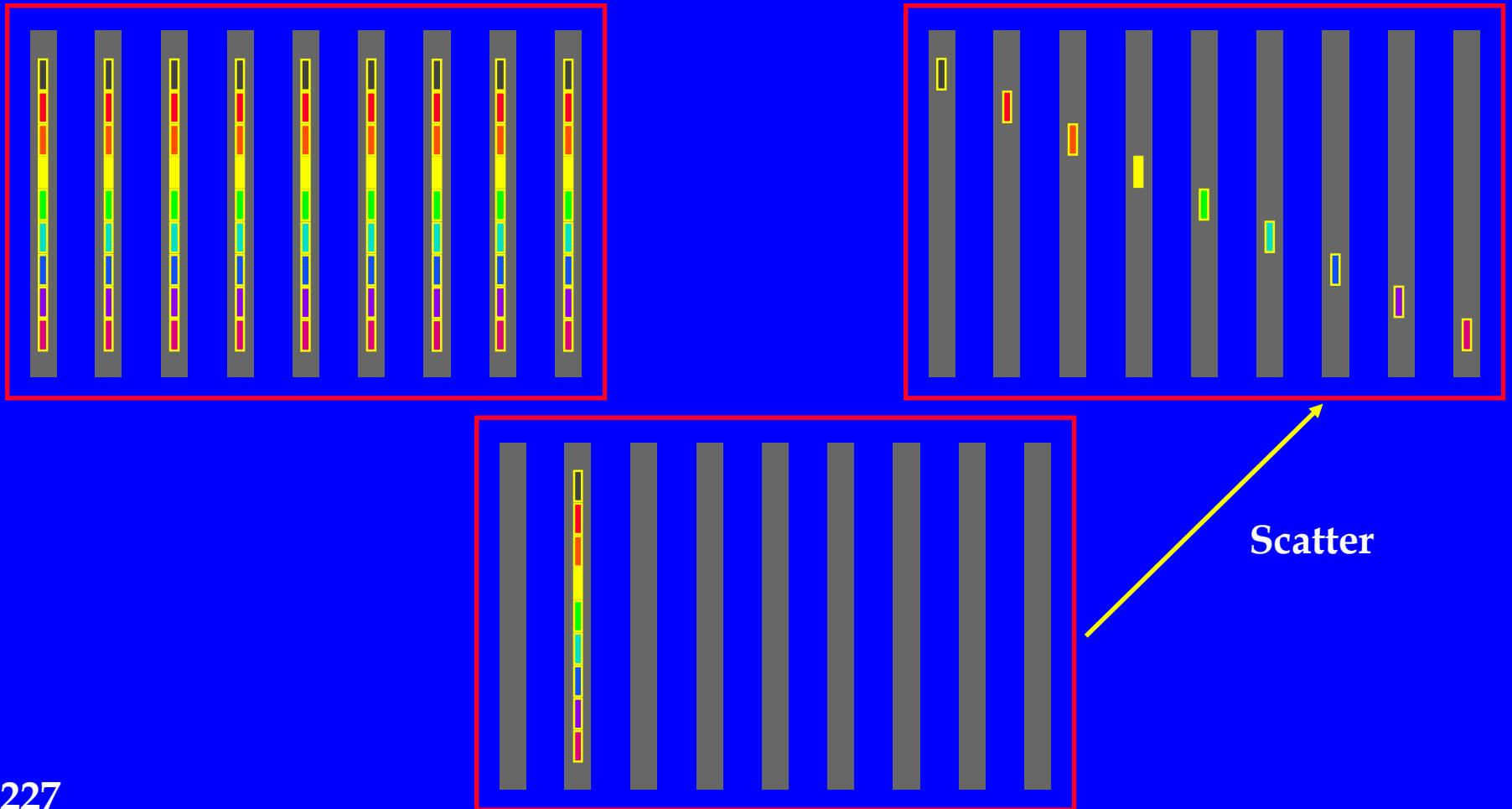
Reduce-scatter (short vector)



Reduce-scatter (short vector)



Reduce-scatter (short vector)



Cost of Reduce(-to-one)/scatter Reduce-scatter

- Assumption: power of two number of nodes

$$\begin{array}{l}
 \text{Reduce(-to-one)} \quad \log(p)(\alpha + n\beta + n\gamma) \\
 \text{scatter} \quad \log(p)\alpha + \frac{p-1}{p}n\beta
 \end{array}$$

$$2\log(p)\alpha + \left(\frac{p-1}{p} + \log(p) \right) n\beta + \log(p)n\gamma$$

Cost of Reduce(-to-one)/scatter reduce-scatter

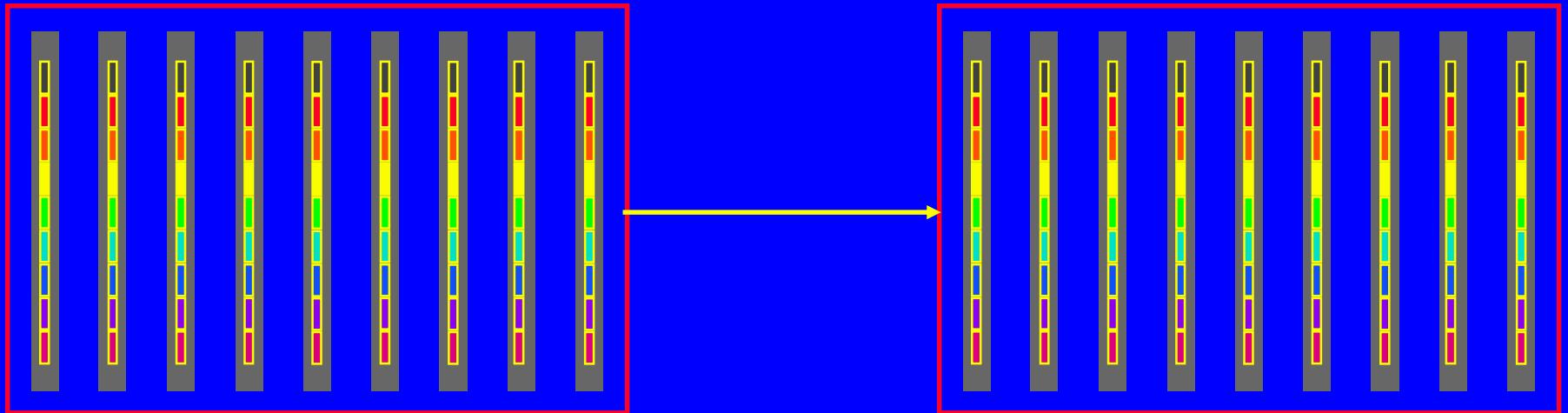
- Assumption: power of two number of nodes

$$\begin{array}{l}
 \text{Reduce(-to-one)} \quad \log(p)(\alpha + n\beta + n\gamma) \\
 \text{scatter} \quad \log(p)\alpha + \frac{p-1}{p}n\beta
 \end{array}$$

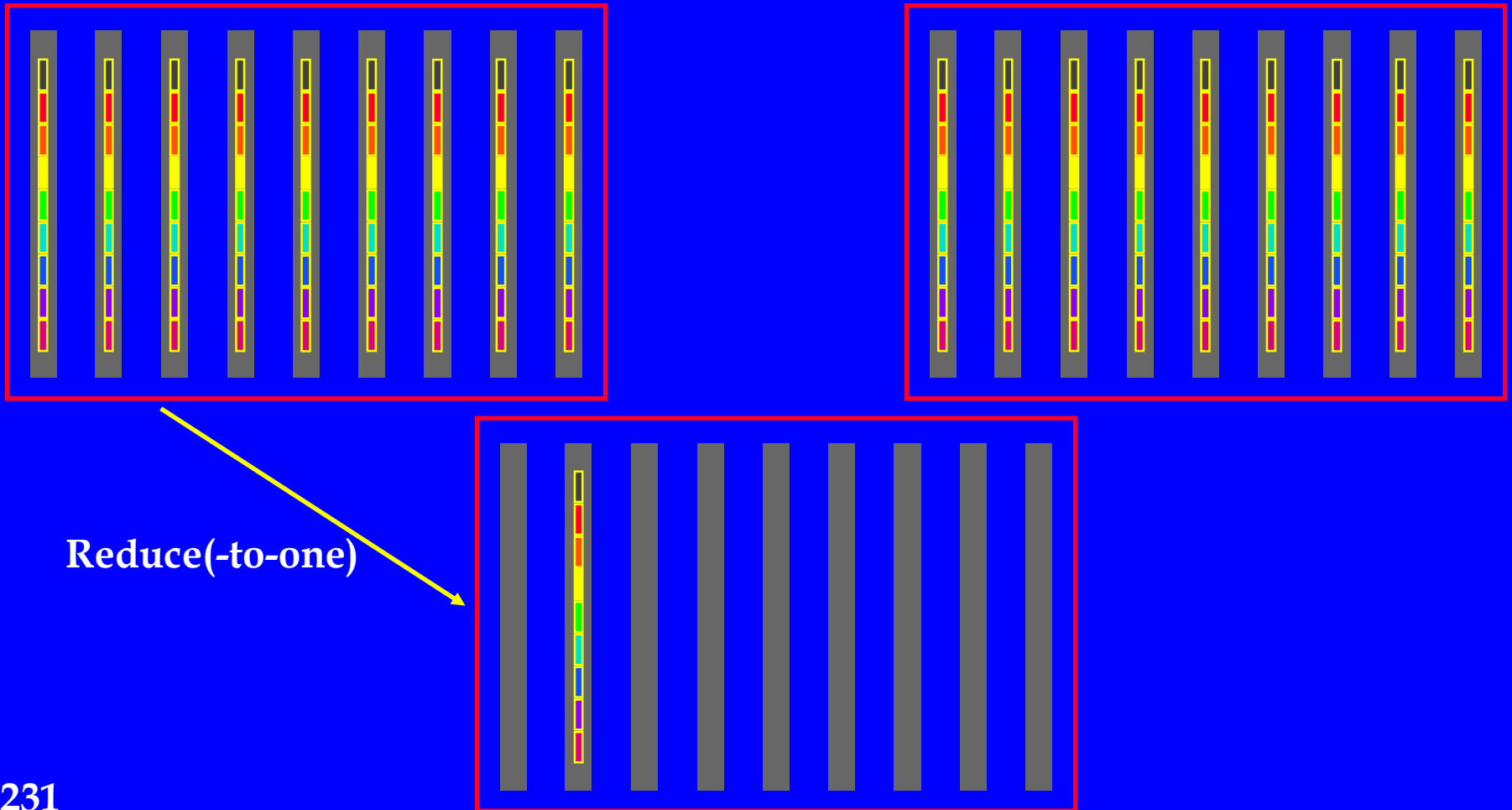
$$2\log(p)\alpha + \left(\frac{p-1}{p} + \log(p) \right) n\beta + \log(p)n\gamma$$

Notice: does not attain lower bound for latency or bandwidth components

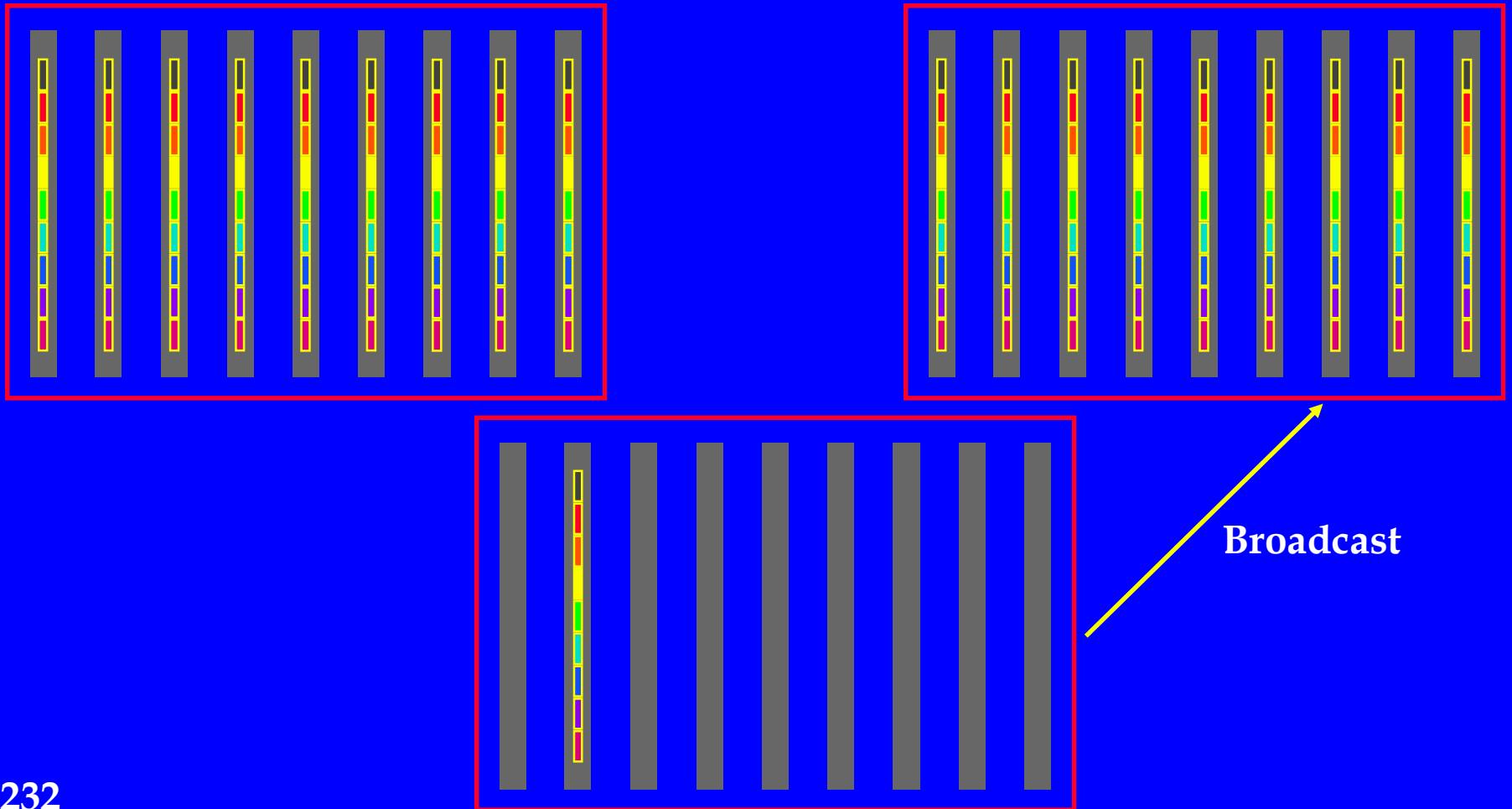
Allreduce (short vector)



Allreduce (short vector)



Allreduce (short vector)



Cost of reduce(-to-one)/broadcast Allreduce

- Assumption: power of two number of nodes

Reduce(-to-one)	$\log(p)(\alpha + n\beta + n\gamma)$
broadcast	$\log(p)(\alpha + n\beta)$
	<hr/>
	$2\log(p)\alpha + 2\log(p)n\beta + \log(p)n\gamma$

Cost of reduce(-to-one)/broadcast Allreduce

- Assumption: power of two number of nodes

Reduce(-to-one)	$\log(p)(\alpha + n\beta + n\gamma)$
broadcast	$\log(p)(\alpha + n\beta)$
<hr/>	
$2\log(p)\alpha + 2\log(p)n\beta + \log(p)n\gamma$	

Notice: does not attain lower bound for latency or bandwidth components

Recap

Reduce(-to-one)

$$\log(p)(\alpha + n\beta + n\gamma)$$

Scatter

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Gather

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Broadcast

$$\log(p)(\alpha + n\beta)$$

Reduce-scatter

Allreduce

Allgather

Recap

Reduce(-to-one)

$$\log(p)(\alpha + n\beta + n\gamma)$$

Scatter

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Gather

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Broadcast

$$\log(p)(\alpha + n\beta)$$

Reduce-scatter

$$2\log(p)\alpha + \log(p)n(\beta + \gamma) + \frac{p-1}{p}n\beta$$

Allreduce

Allgather

Recap

Reduce(-to-one)

$$\log(p)(\alpha + n\beta + n\gamma)$$

Scatter

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Gather

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Broadcast

$$\log(p)(\alpha + n\beta)$$

Reduce-scatter

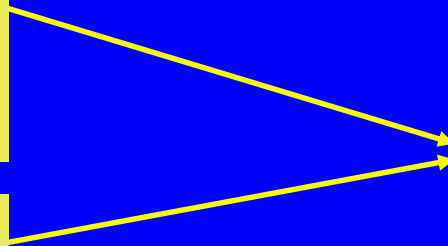
$$2\log(p)\alpha + \log(p)n(\beta + \gamma) + \frac{p-1}{p}n\beta$$

Allreduce

$$2\log(p)\alpha + \log(p)n(2\beta + \gamma)$$

Allgather

$$2\log(p)\alpha + \log(p)n\beta + \frac{p-1}{p}n\beta$$



Recap

Reduce(-to-one)

$$\log(p)(\alpha + n\beta + n\gamma)$$

Scatter

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Gather

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Broadcast

$$\log(p)(\alpha + n\beta)$$

Reduce-scatter

$$2\log(p)\alpha + \log(p)n(\beta + \gamma) + \frac{p-1}{p}n\beta$$

Allreduce

$$2\log(p)\alpha + \log(p)n(2\beta + \gamma)$$

Allgather

Recap

Reduce(-to-one)

$$\log(p)(\alpha + n\beta + n\gamma)$$

Scatter

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Gather

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Broadcast

$$\log(p)(\alpha + n\beta)$$

Reduce-scatter

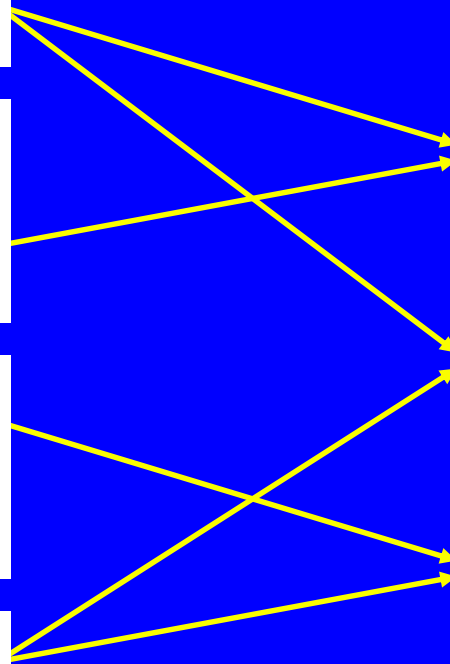
$$2\log(p)\alpha + \log(p)n(\beta + \gamma) + \frac{p-1}{p}n\beta$$

Allreduce

$$2\log(p)\alpha + \log(p)n(2\beta + \gamma)$$

Allgather

$$2\log(p)\alpha + \log(p)n\beta + \frac{p-1}{p}n\beta$$



A building block approach to library implementation

- Short vector case
- Long vector case
- Hybrid algorithms

Long vector case

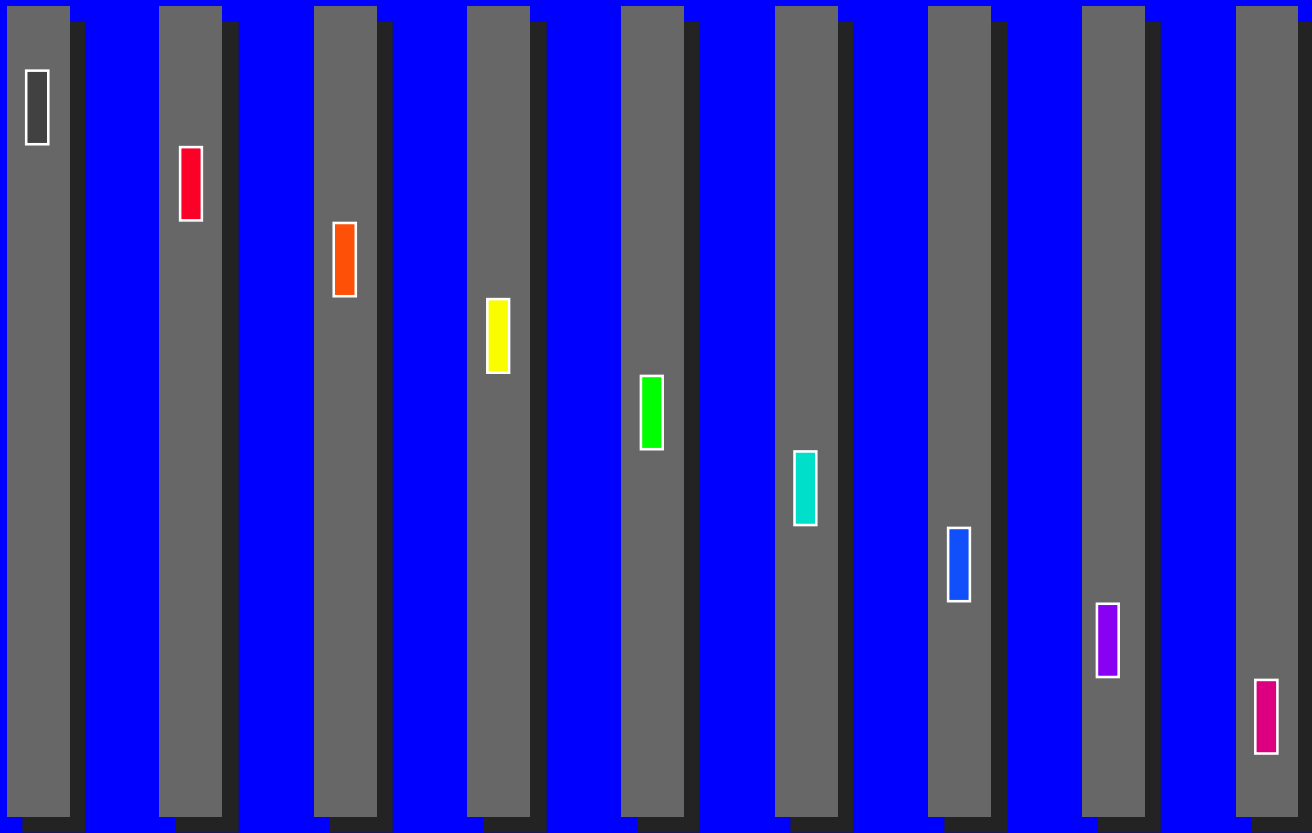
- **Primary concern:**
 - algorithms must have low cost due to vector length
 - algorithms must avoid network conflicts
- **Secondary concerns:**
 - algorithms must work for arbitrary number of nodes
 - » in particular, not just for power-of-two numbers of nodes

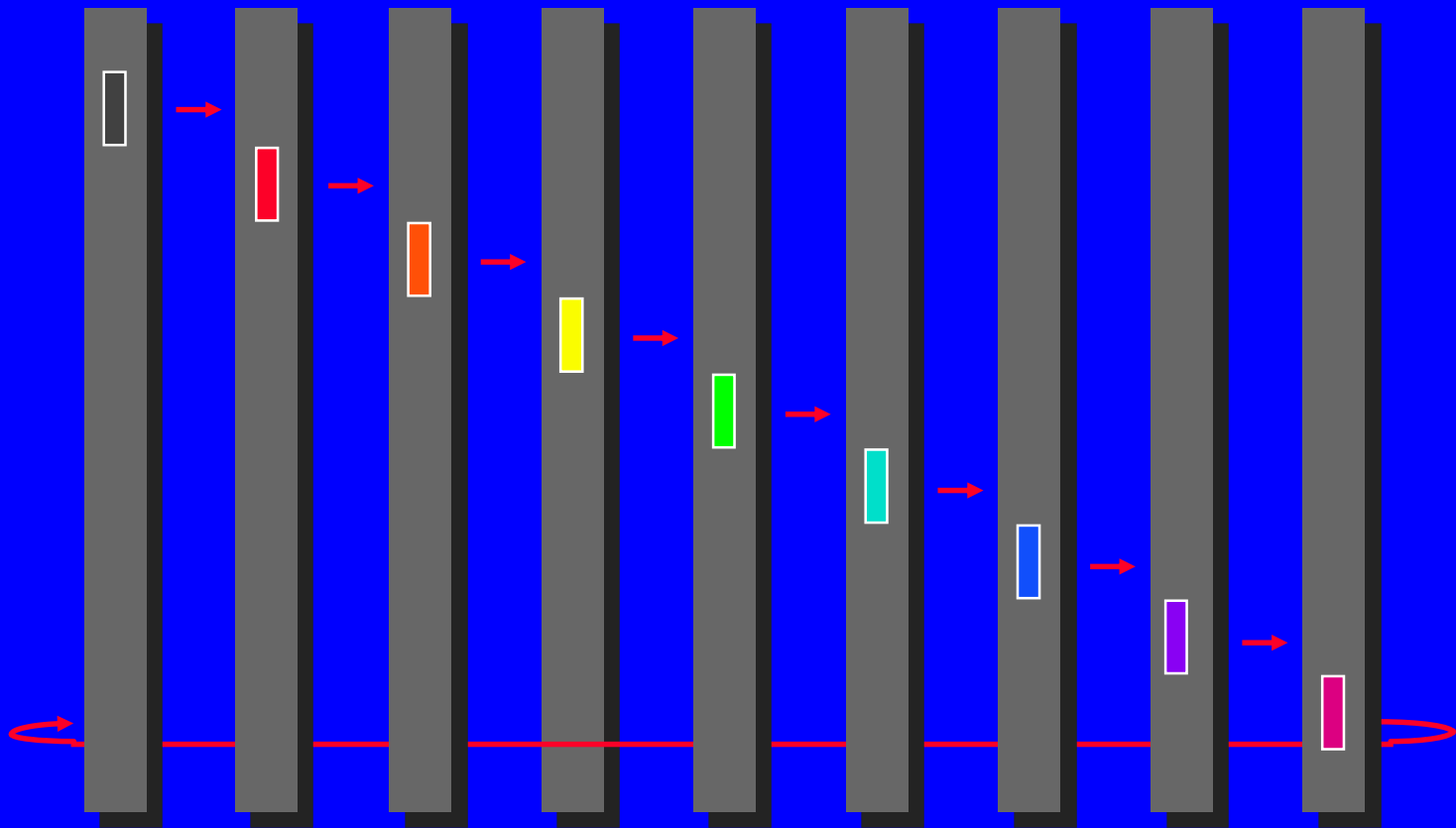
Long vector building blocks

- We will show how the following building blocks:
 - collect/distributed combine
 - scatter/gathercan be implemented using “bucket” algorithms while attaining
 - minimal cost due to length of vectors
 - implementation for arbitrary numbers of nodes
 - no network conflicts
- NOTICE: scatter and gather already satisfy these conditions

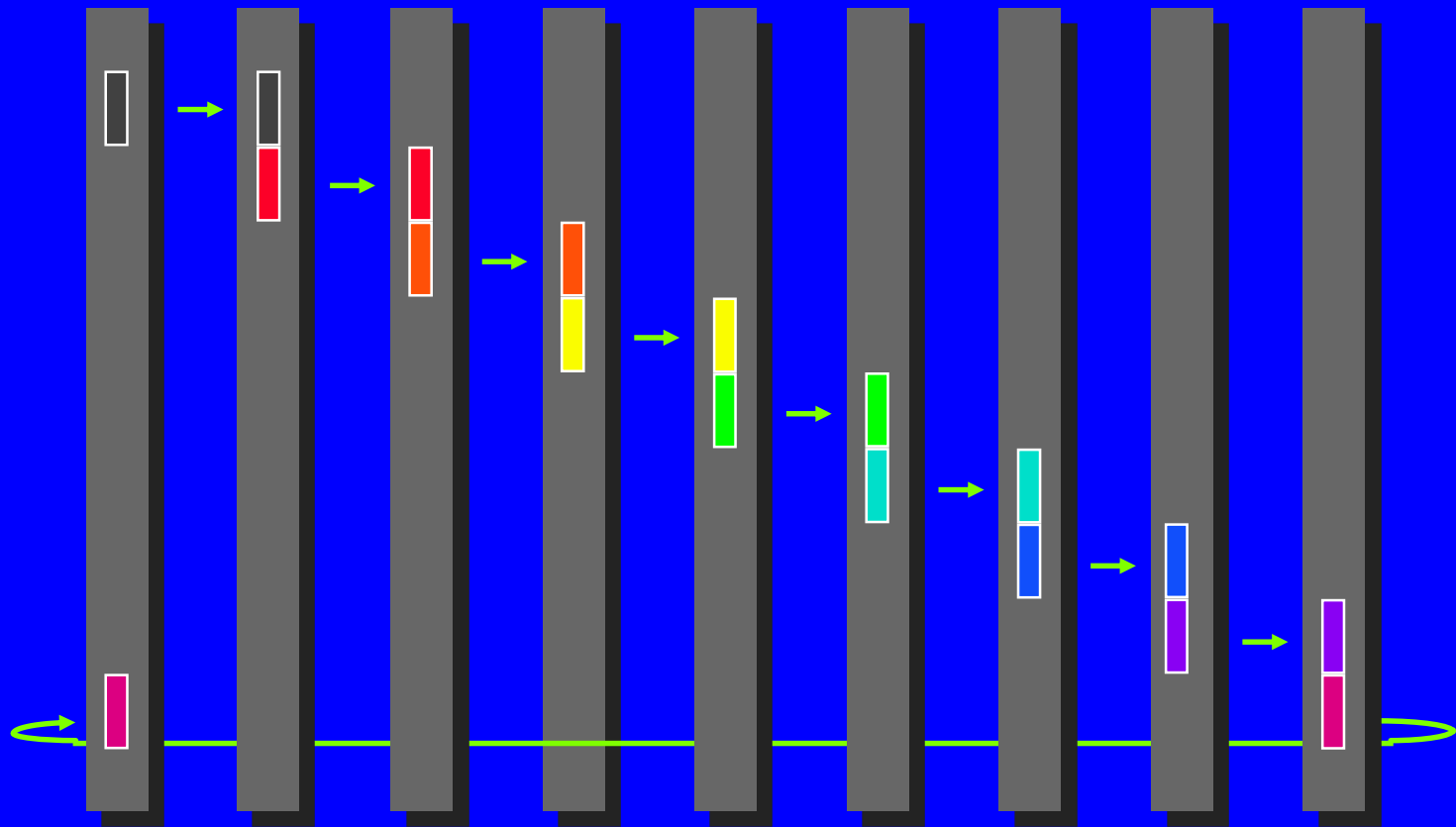
General principles

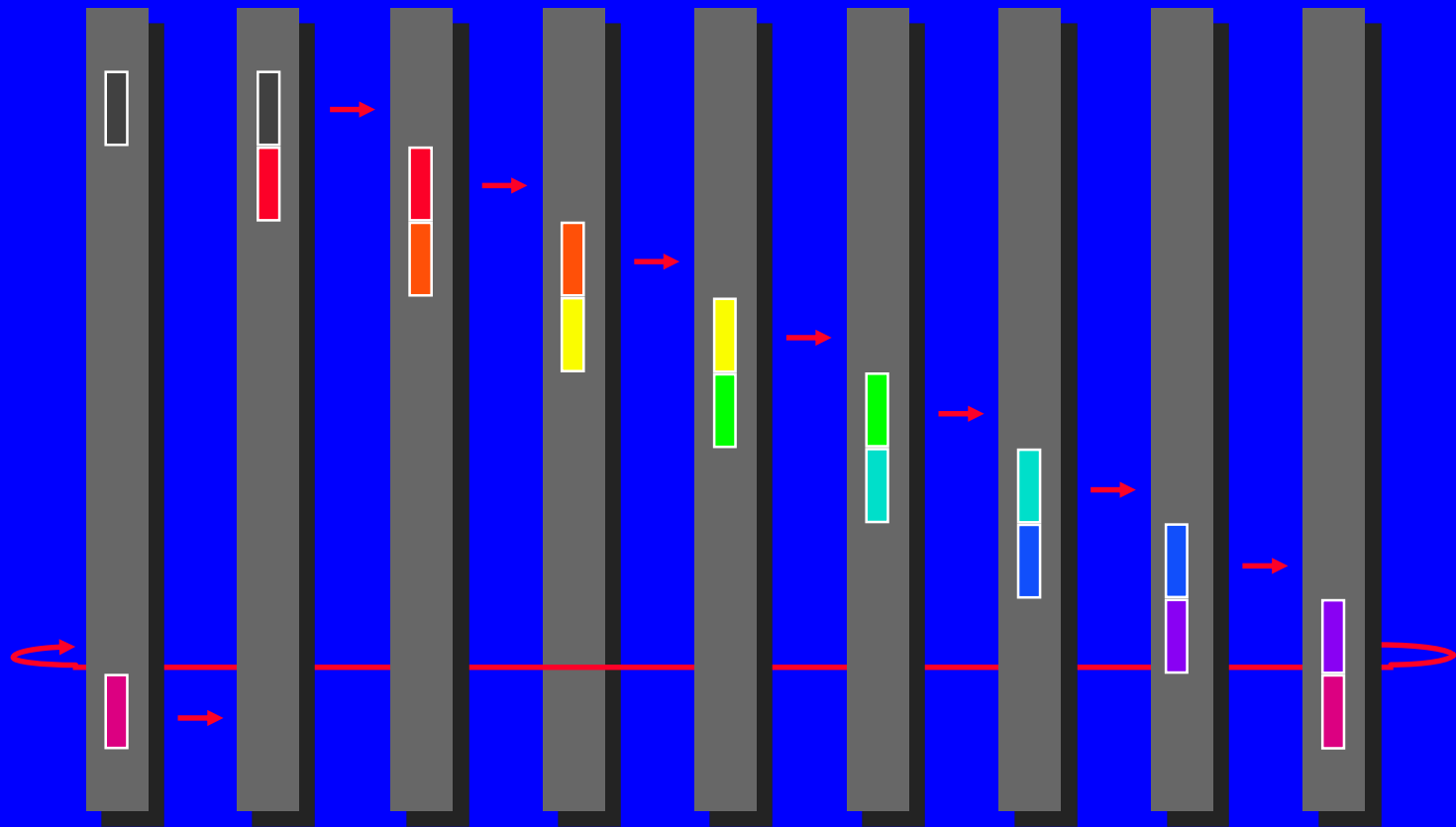
- A logical ring can be embedded in a physical linear array with worm-hole routing, since the “wrap-around” message doesn’t conflict
 - This is used to “drop off” messages or to “pick up” contributions

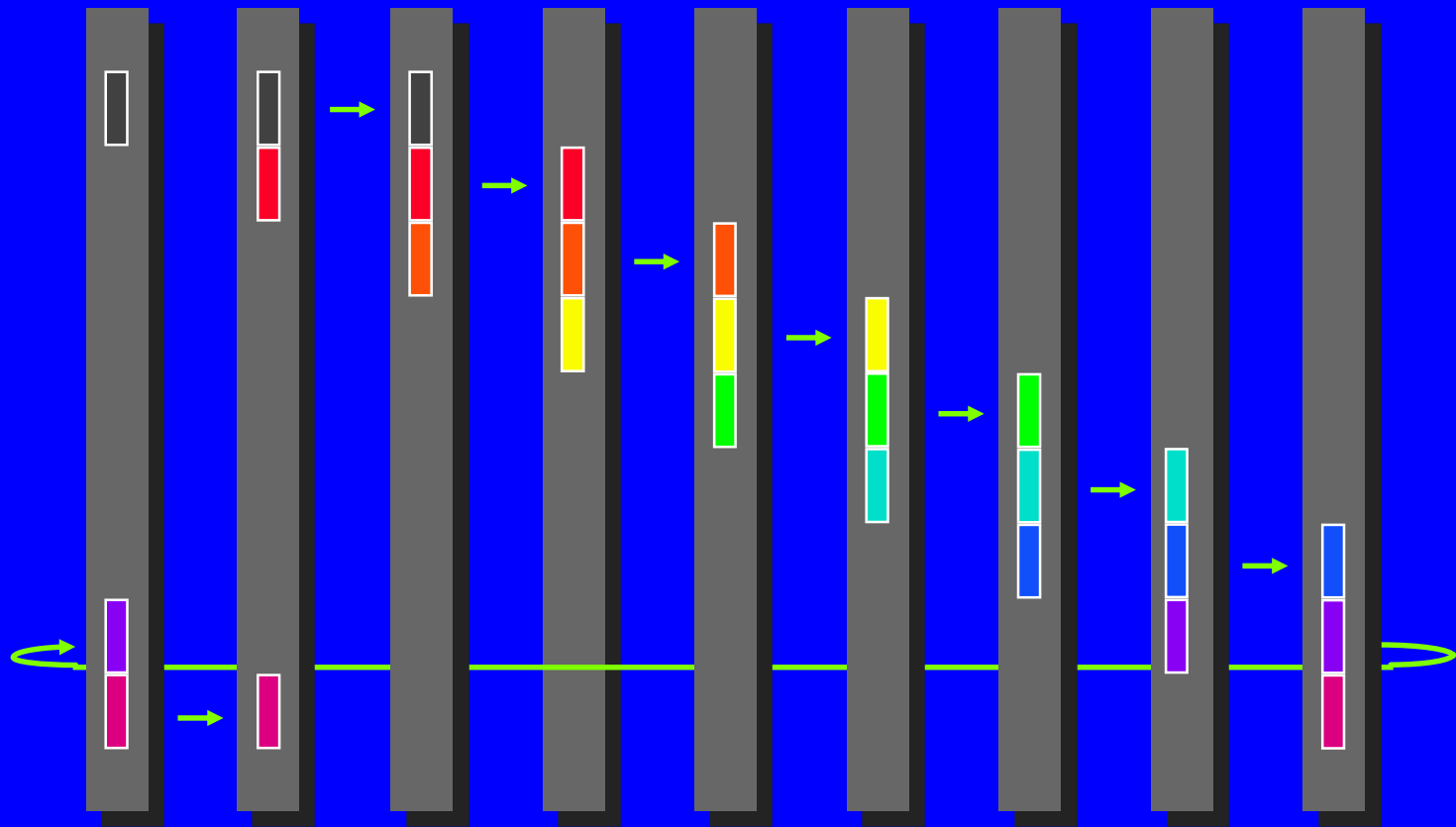




- A logical ring can be embedded in a physical linear array with worm-hole routing, since the “wrap-around” message doesn’t conflict







General principles

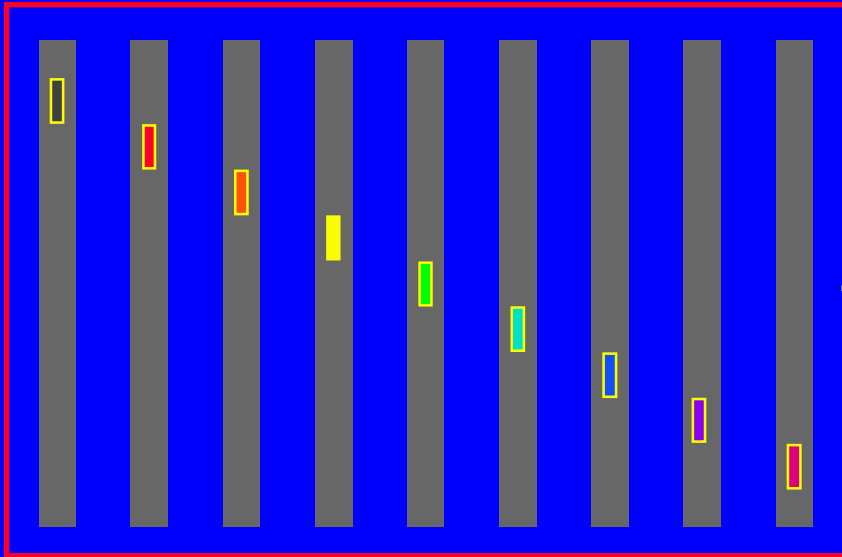
- Can be used to implement the following building blocks:
 - collect
 - distributed combine

using a bucket algorithm embedded in the physical linear array while attaining

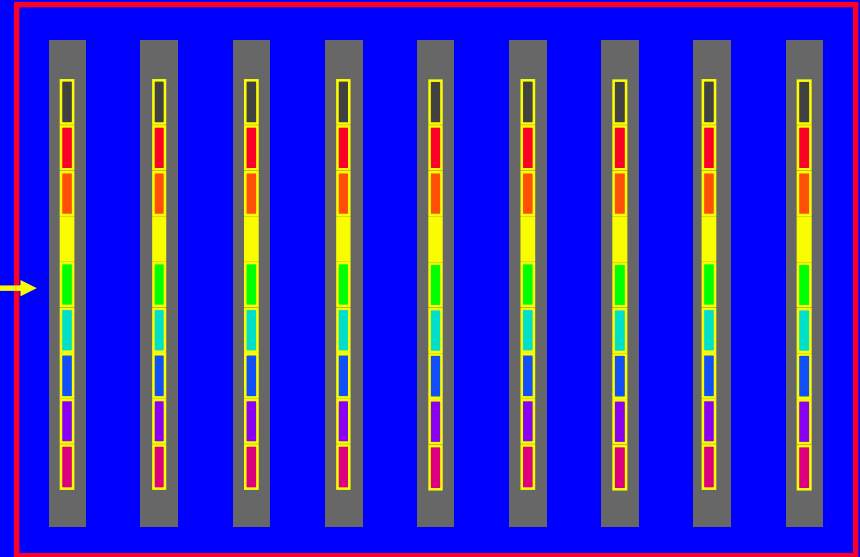
- minimal cost due to vector length
- implementation for arbitrary numbers of nodes
- no network conflicts

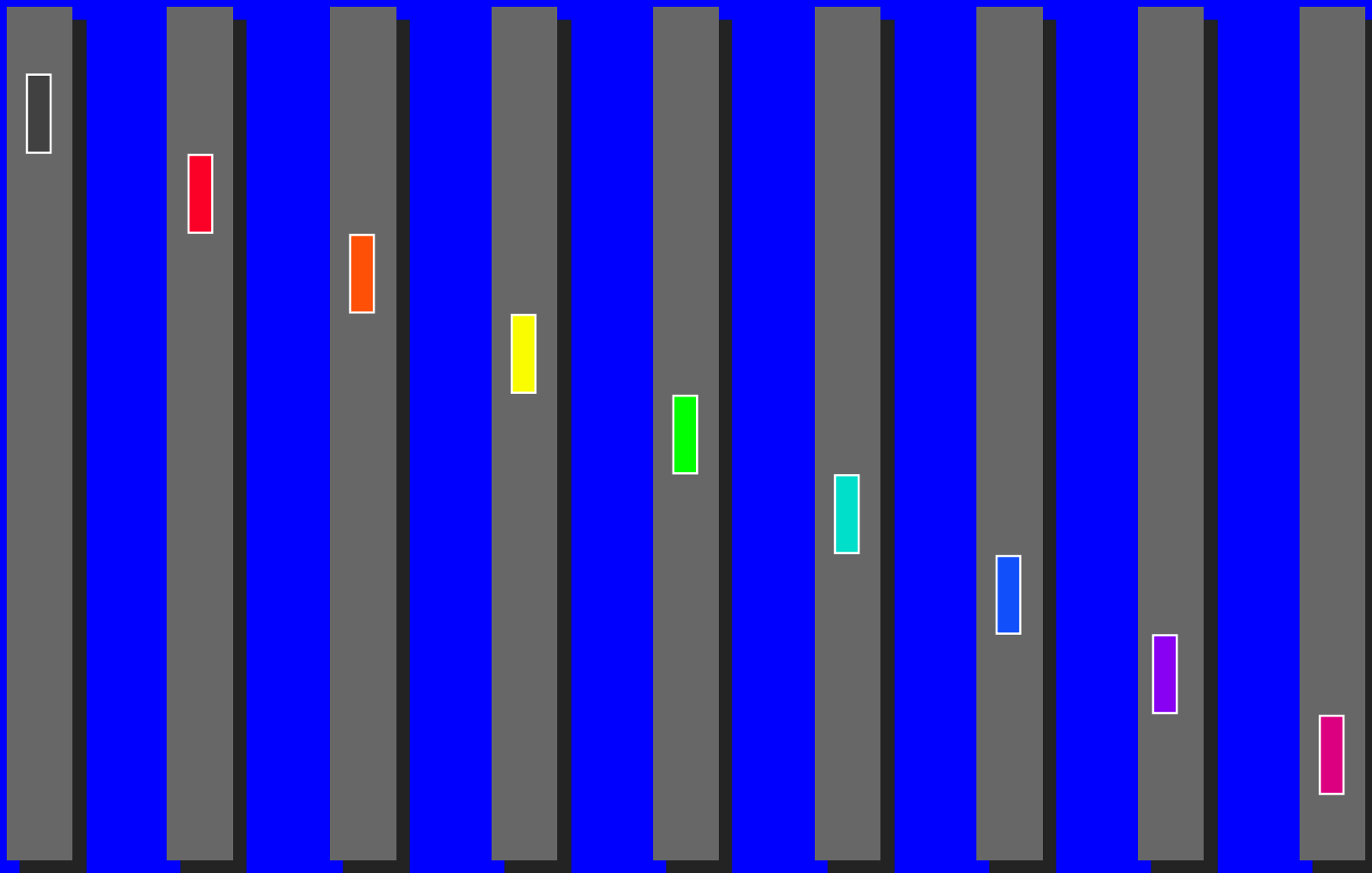
Allgather

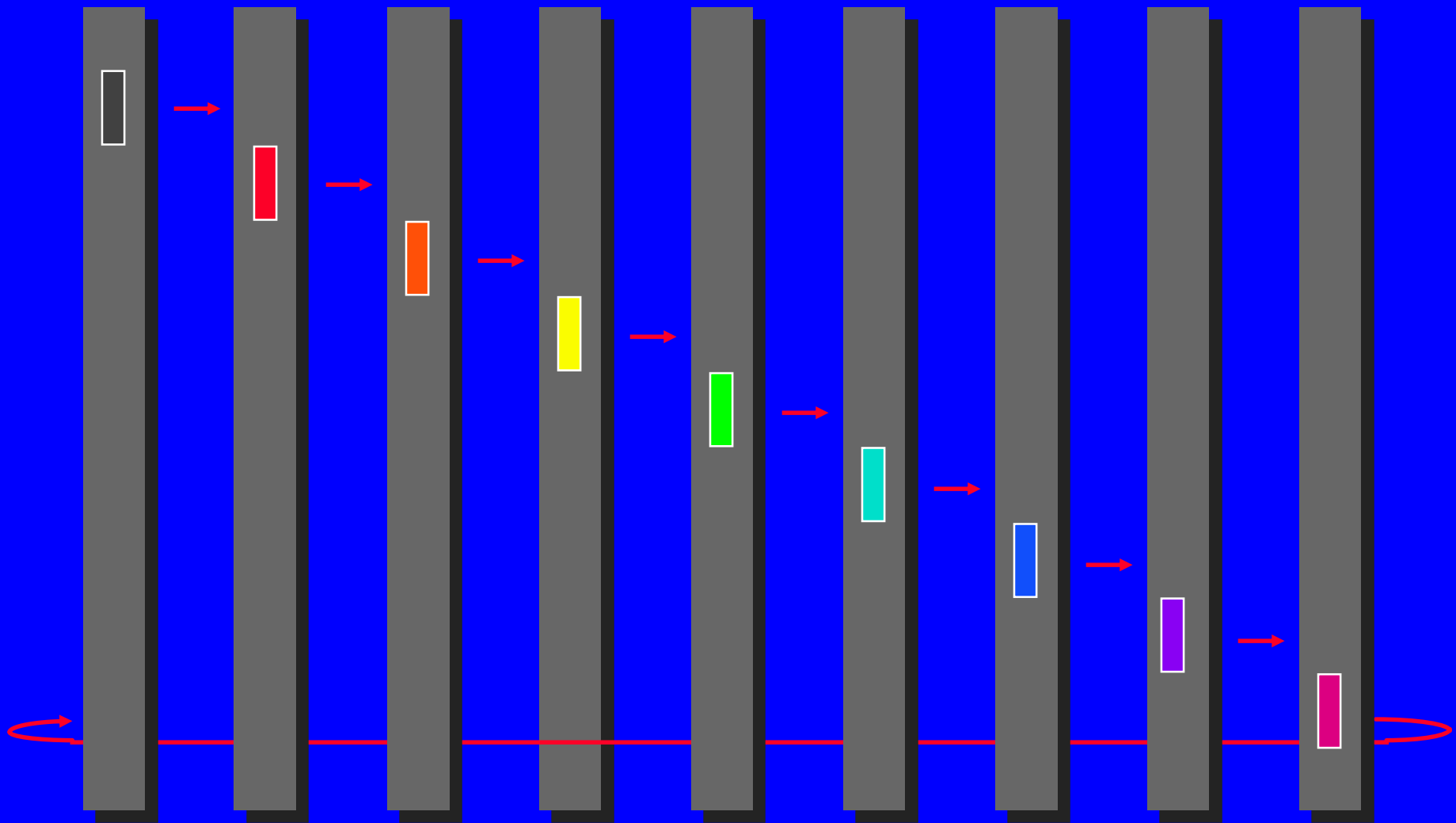
Before

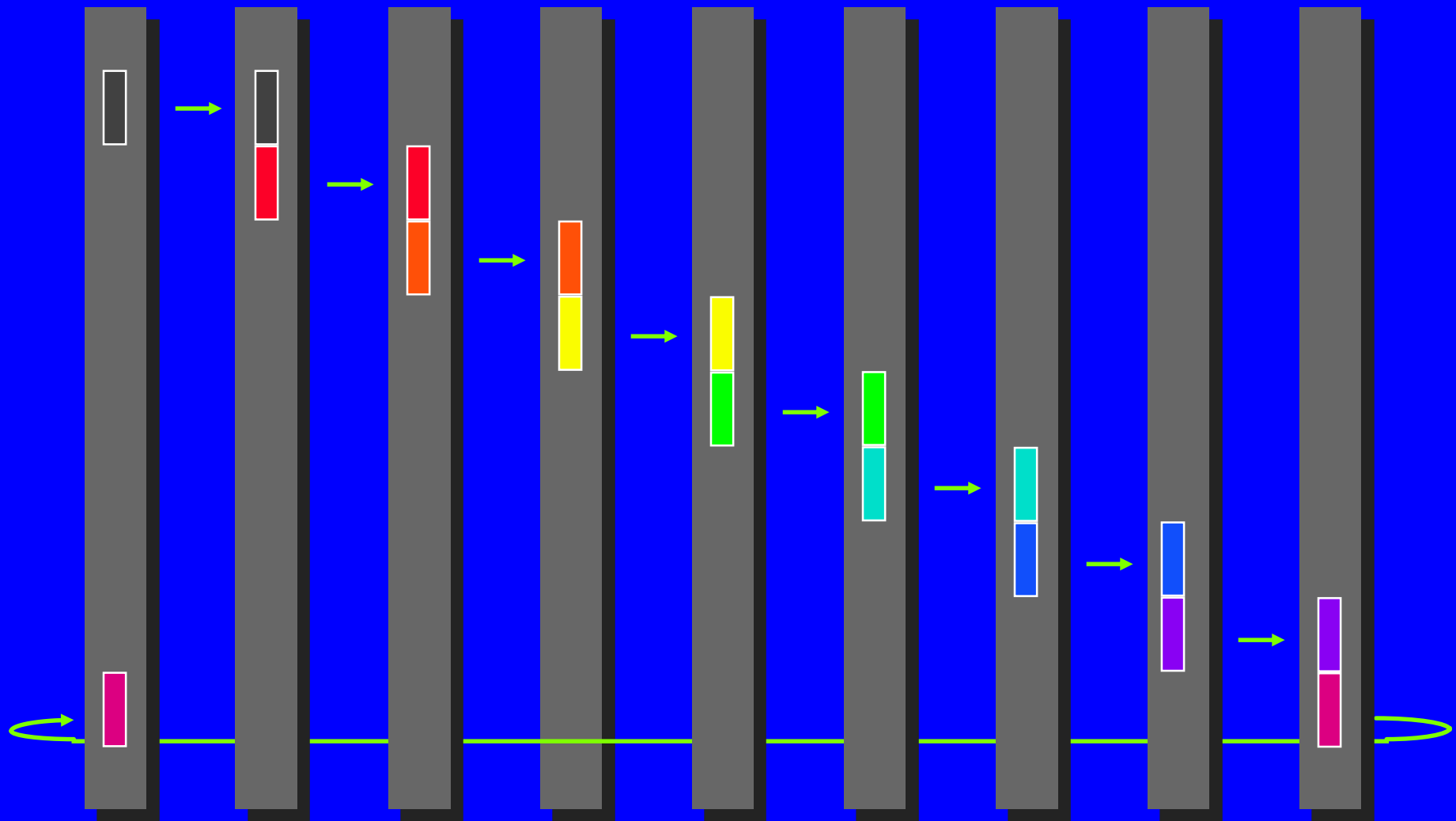


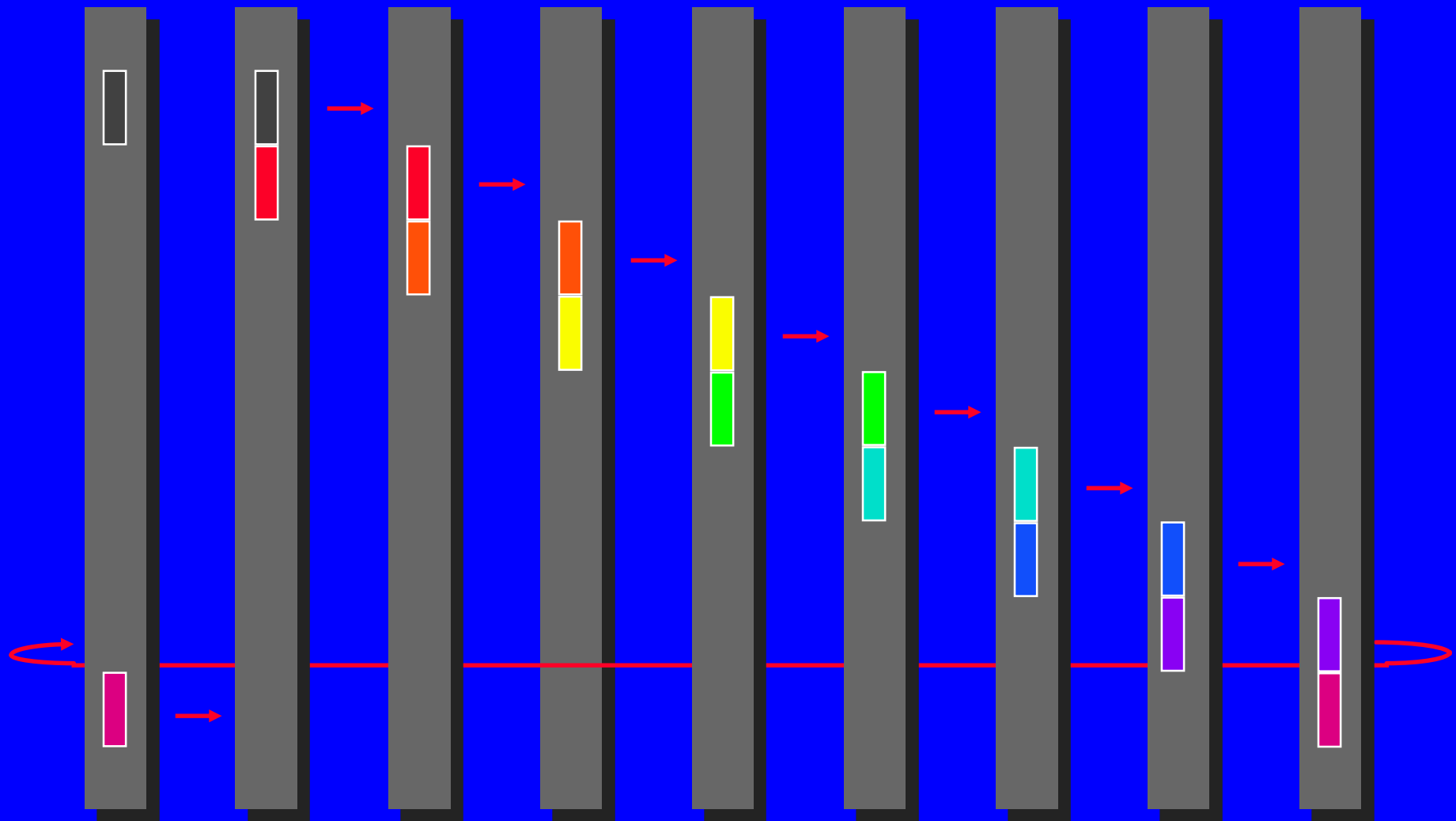
After

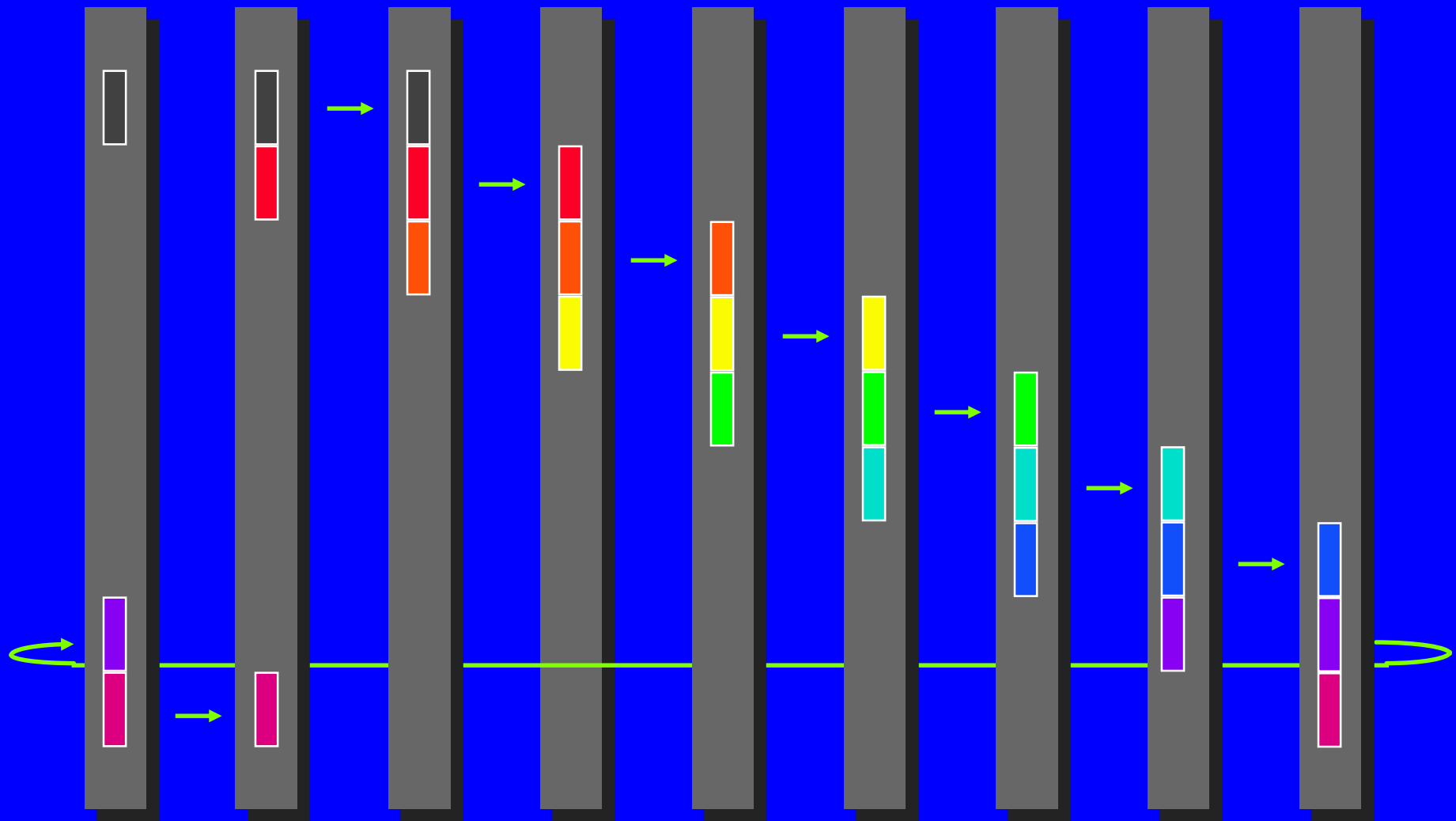


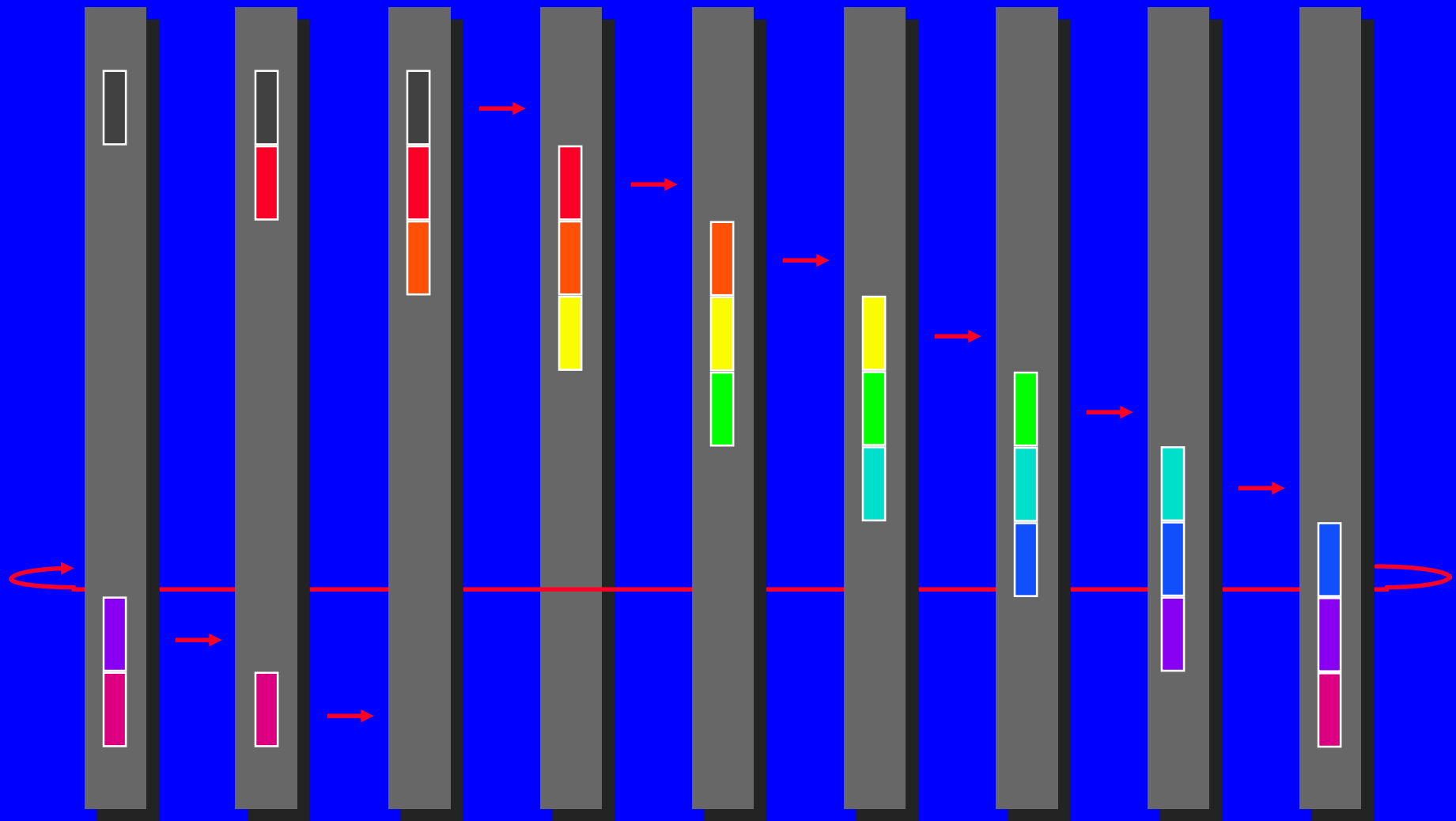


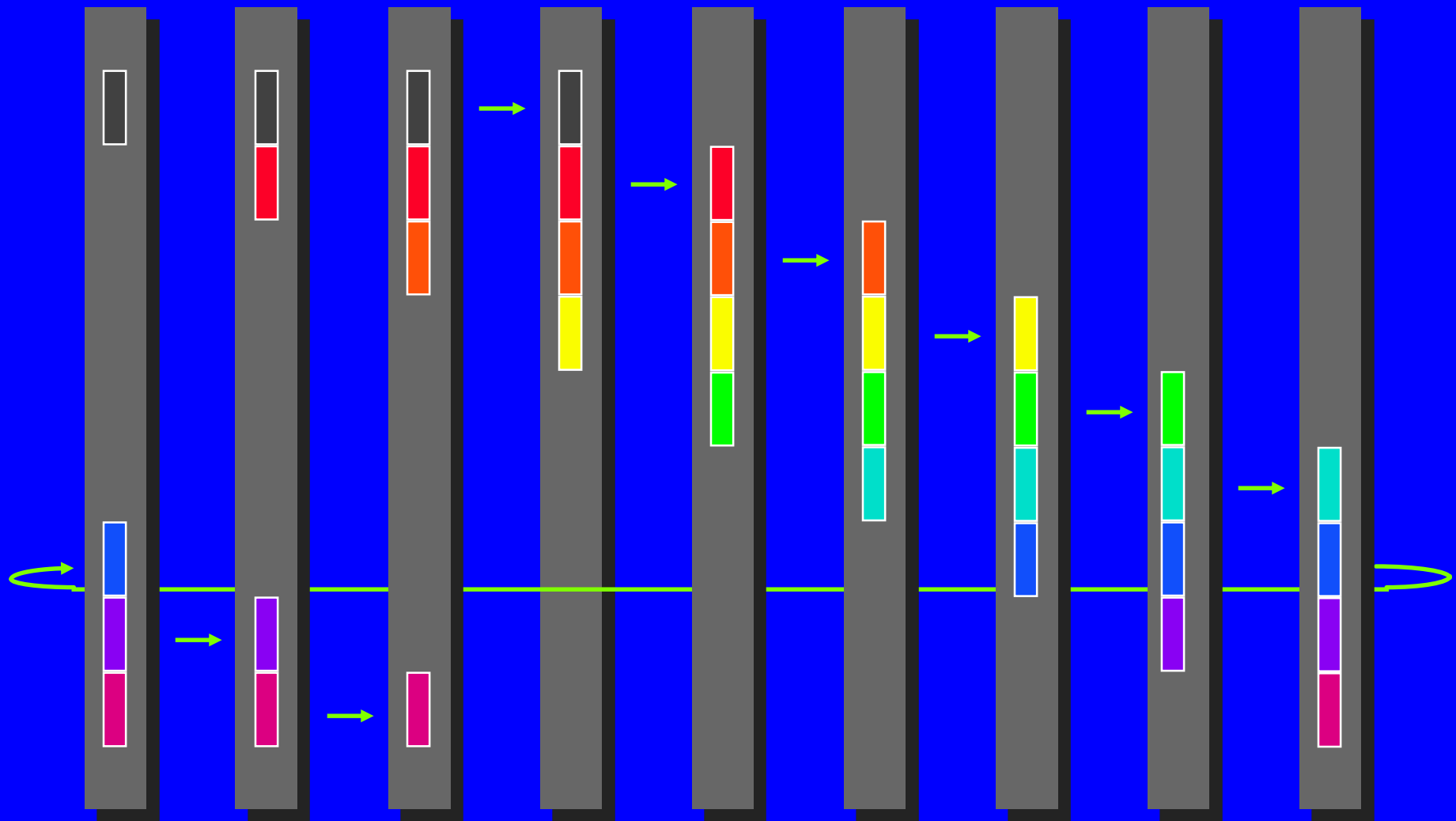


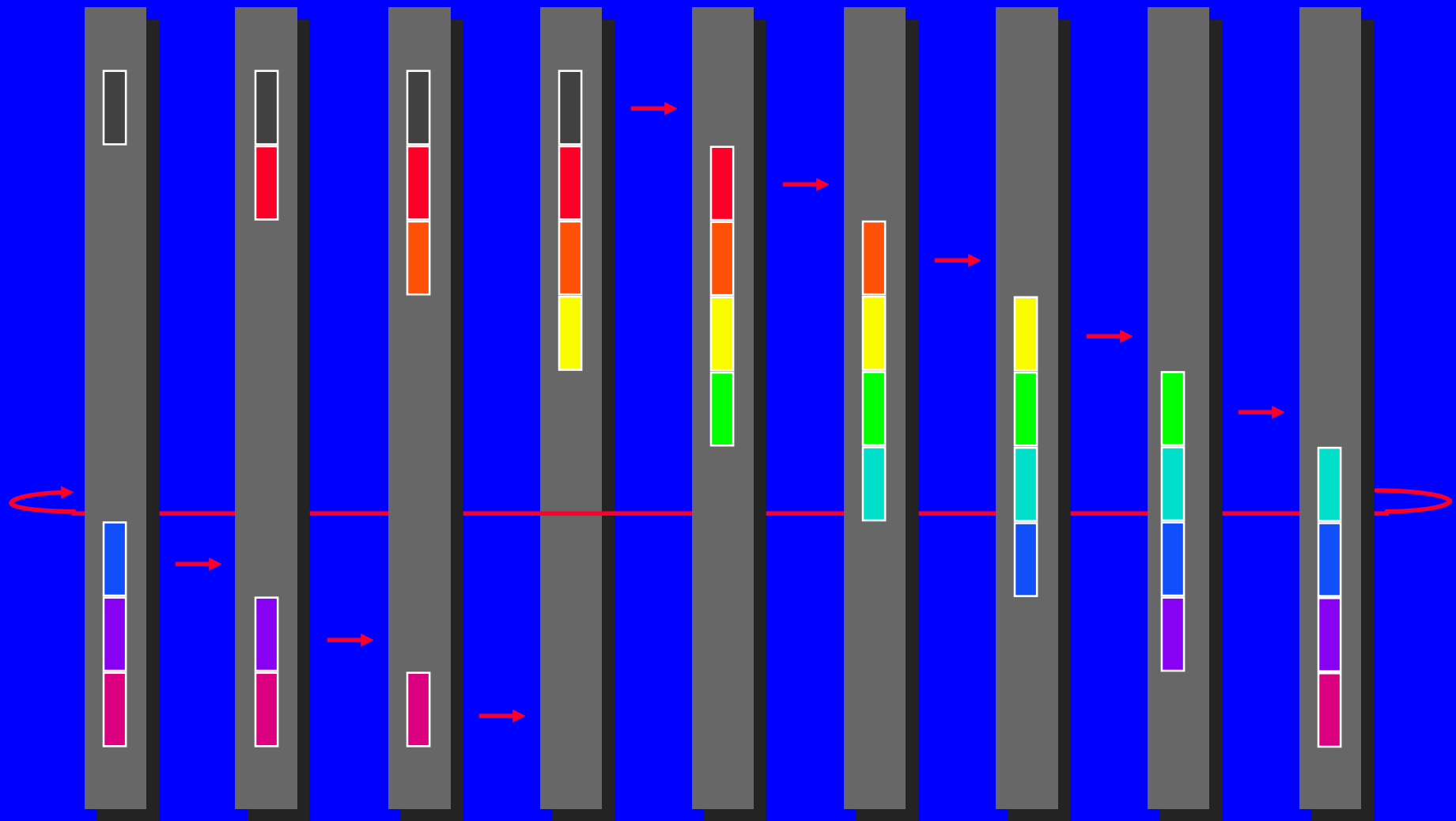




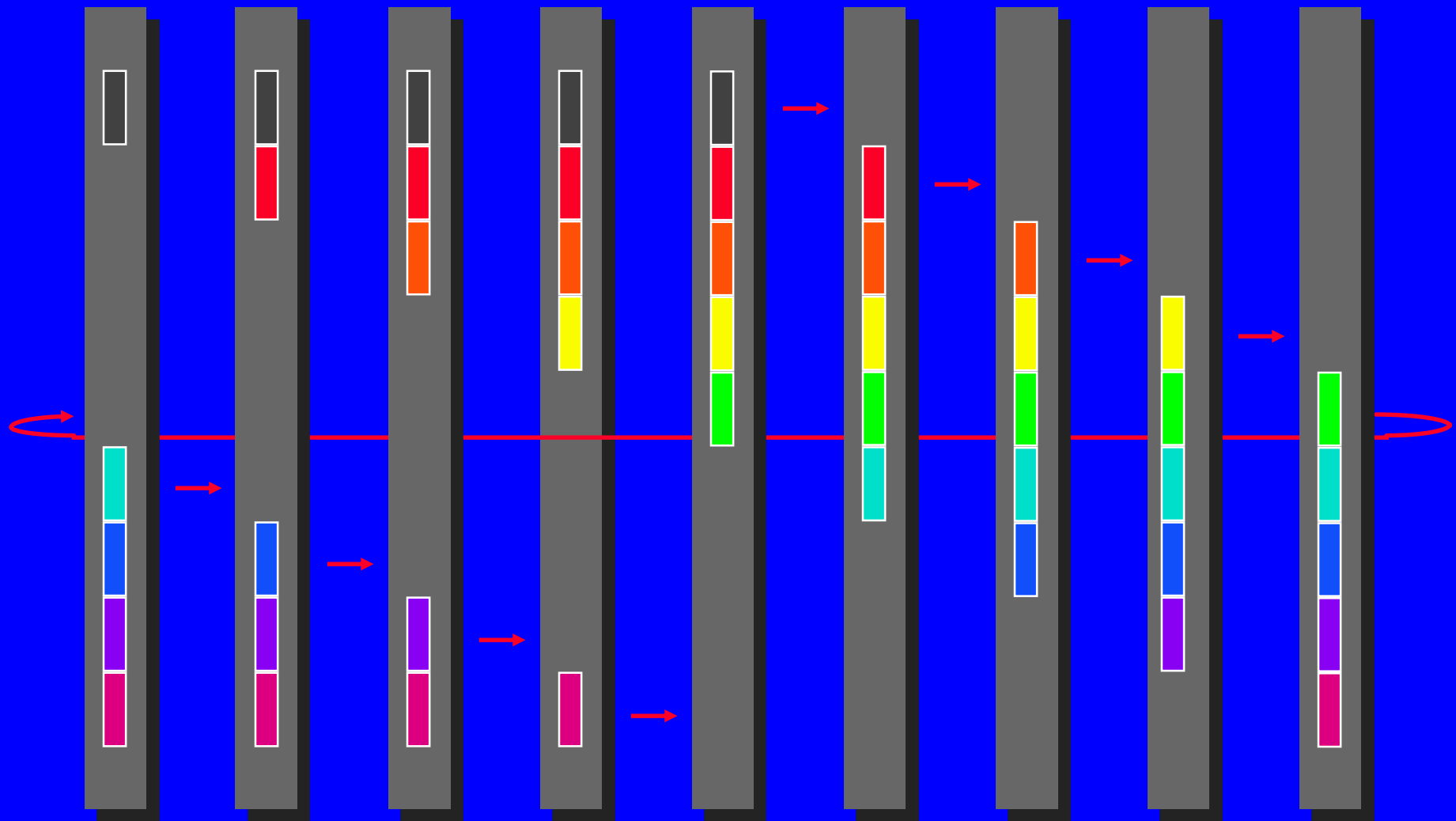


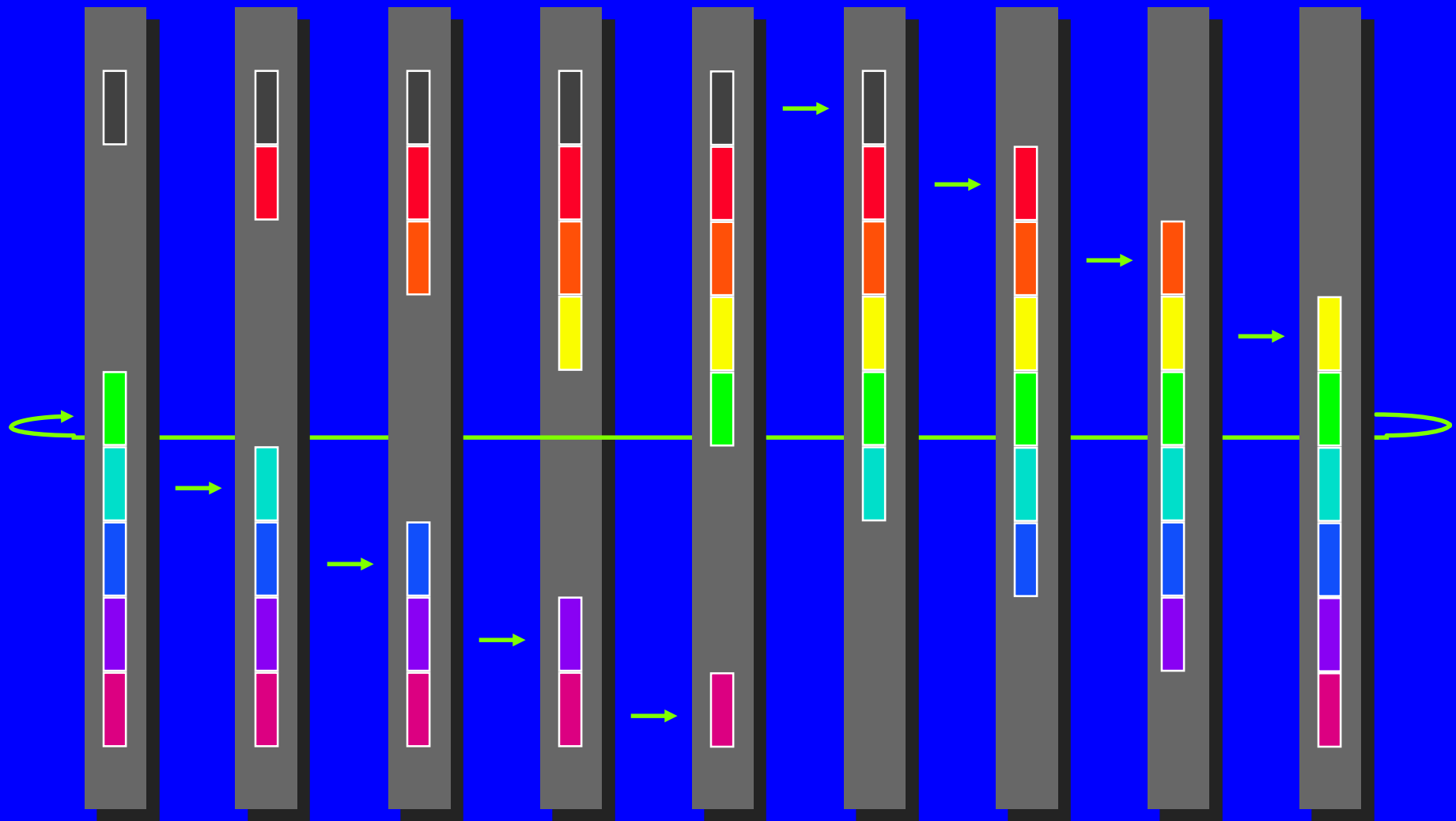


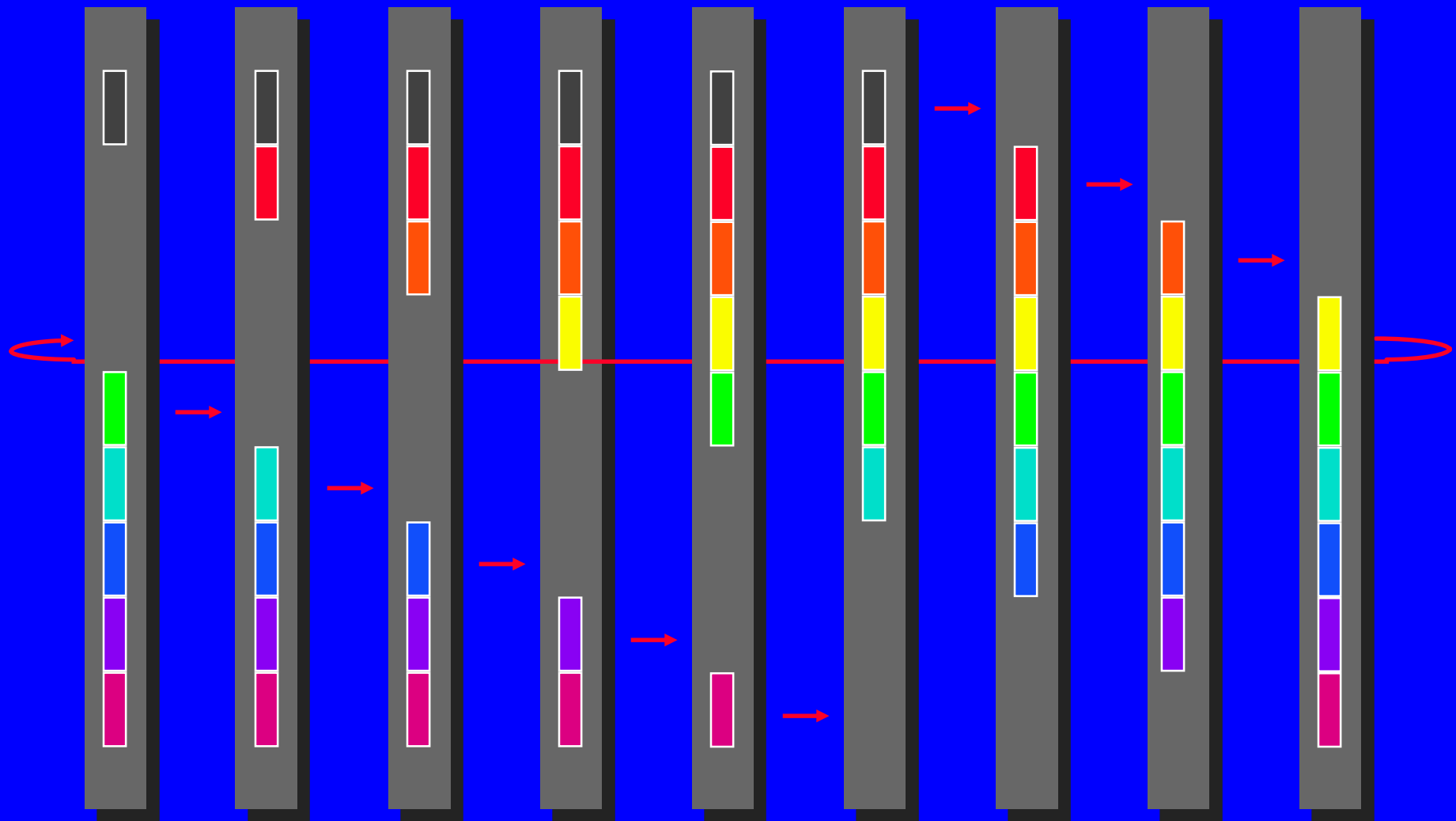


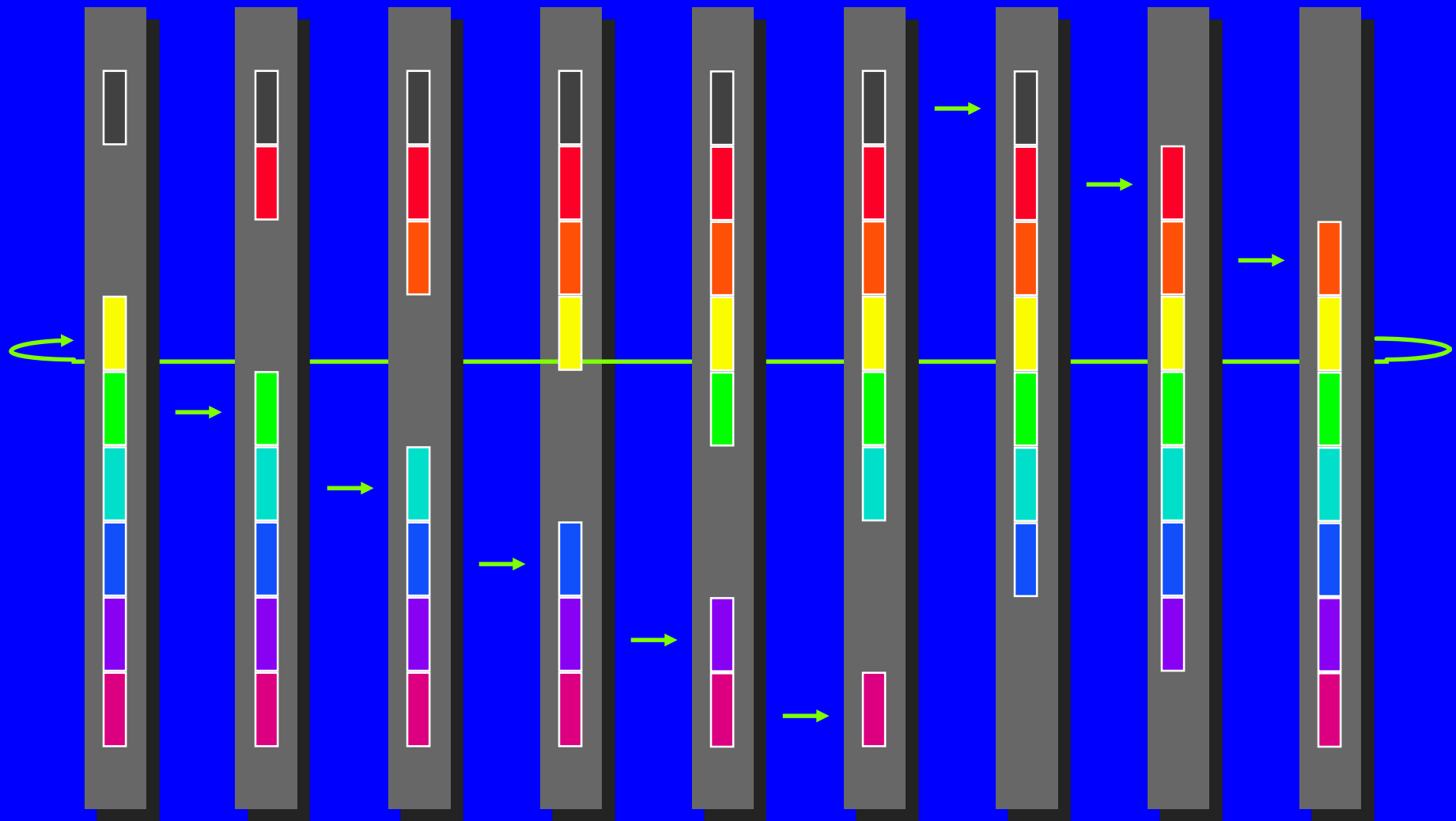


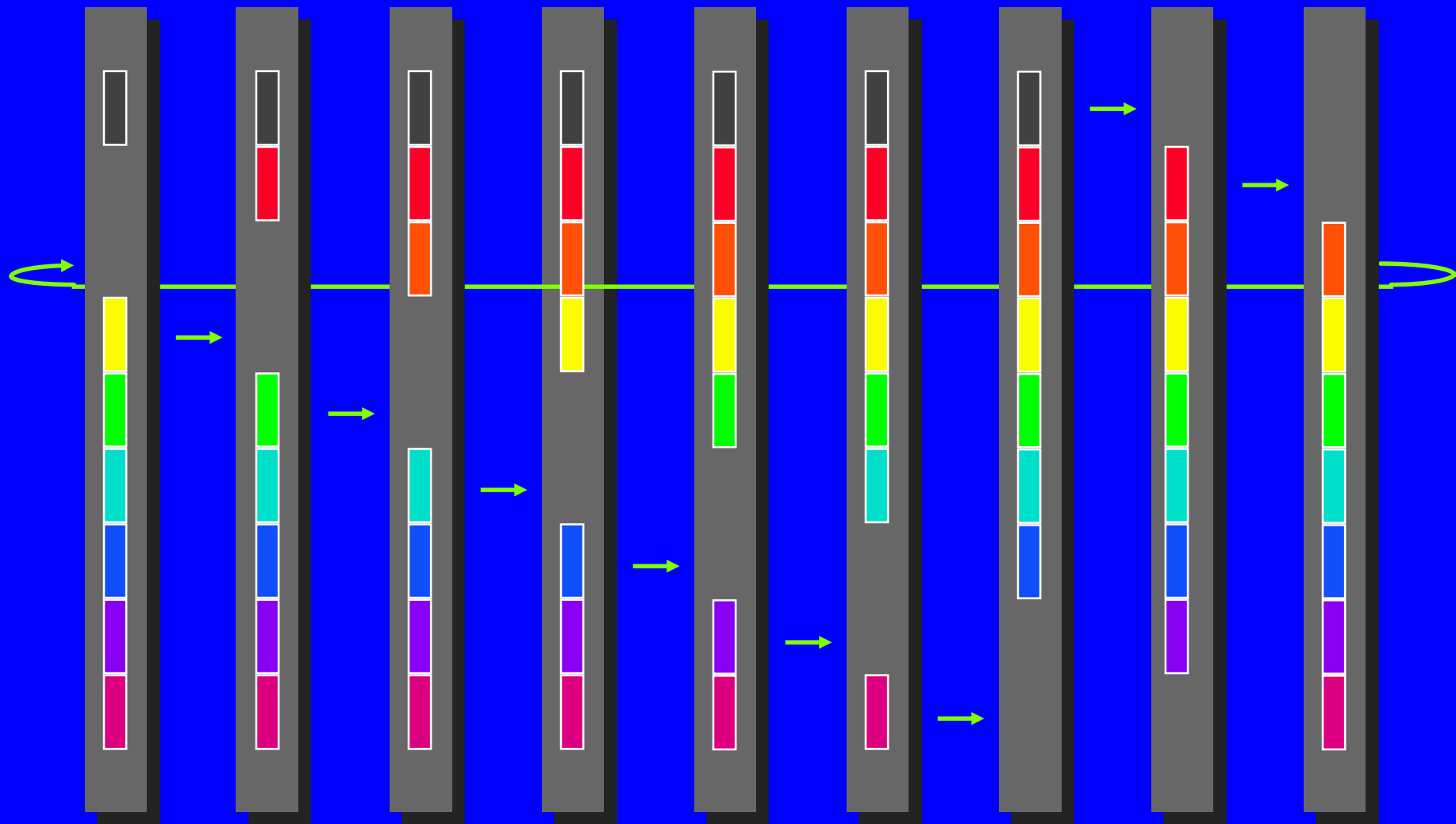


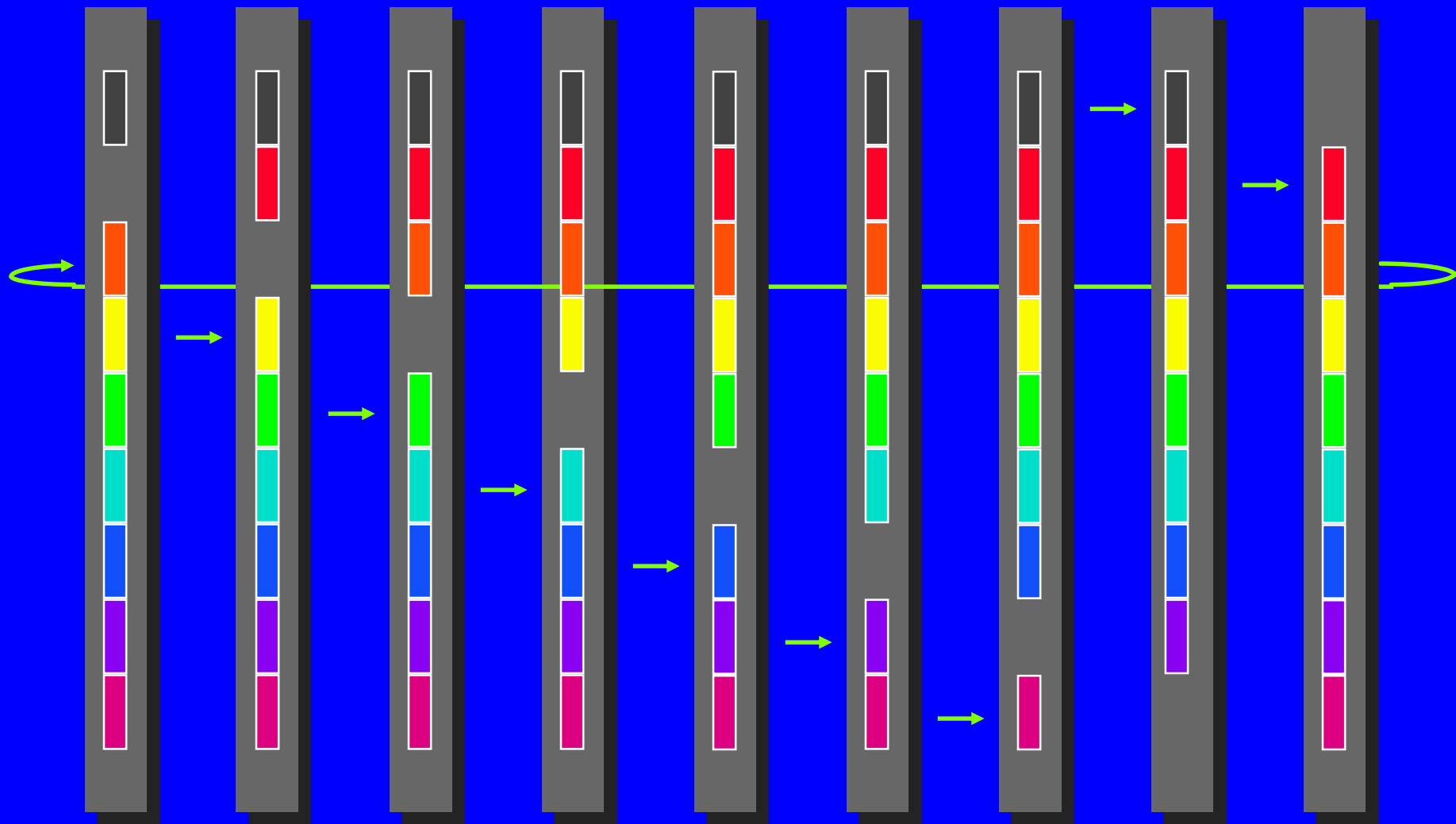


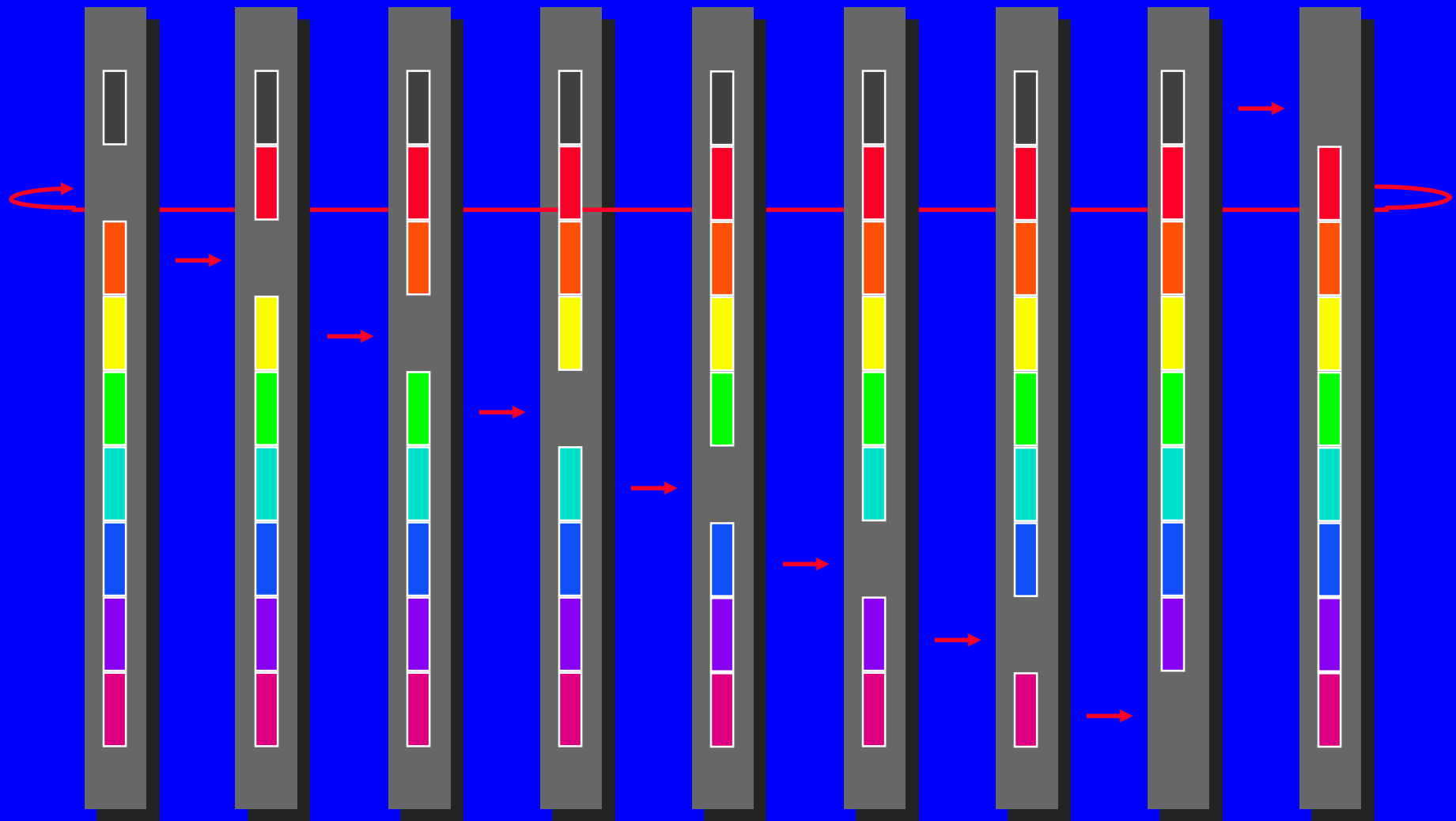


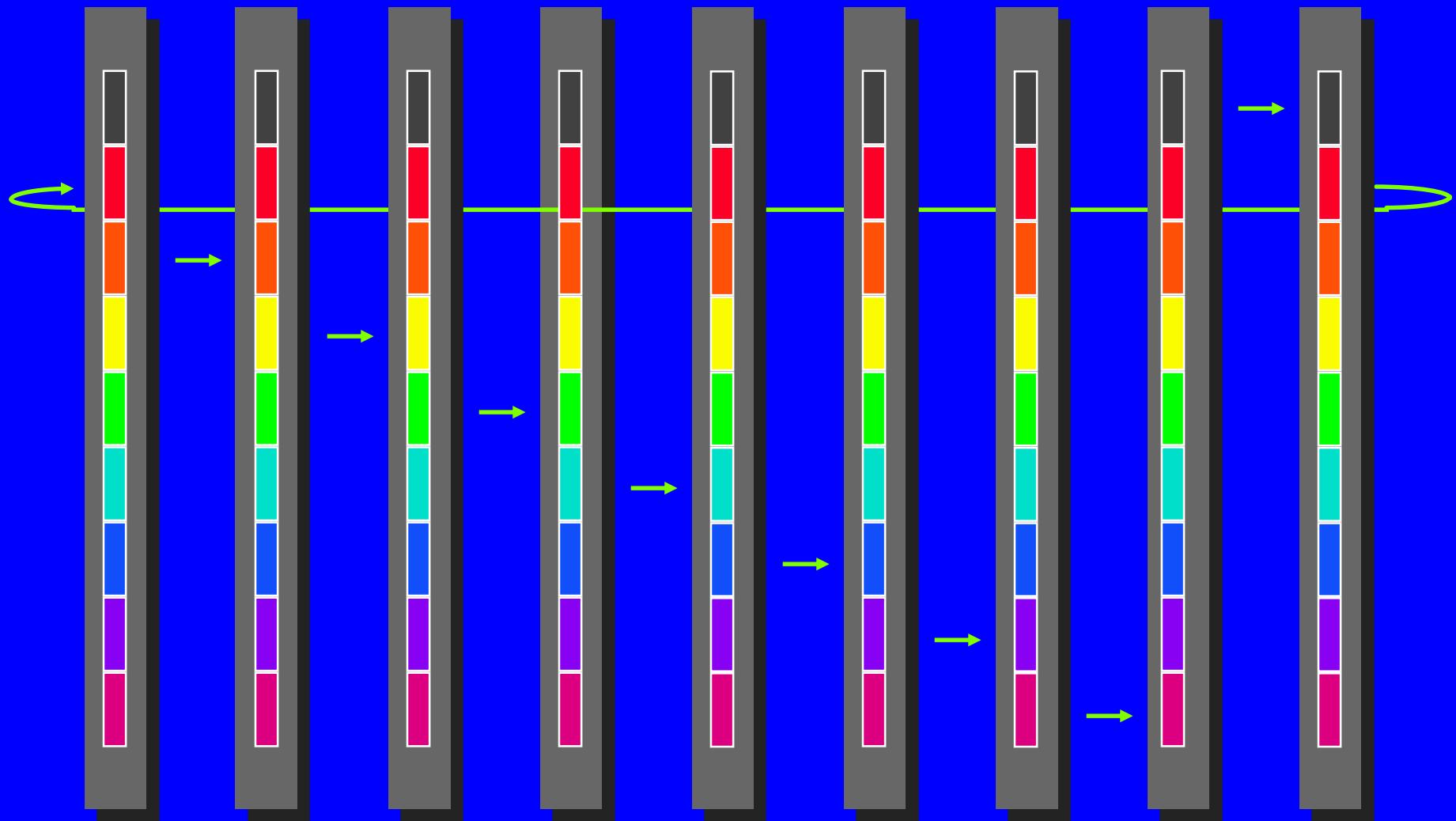


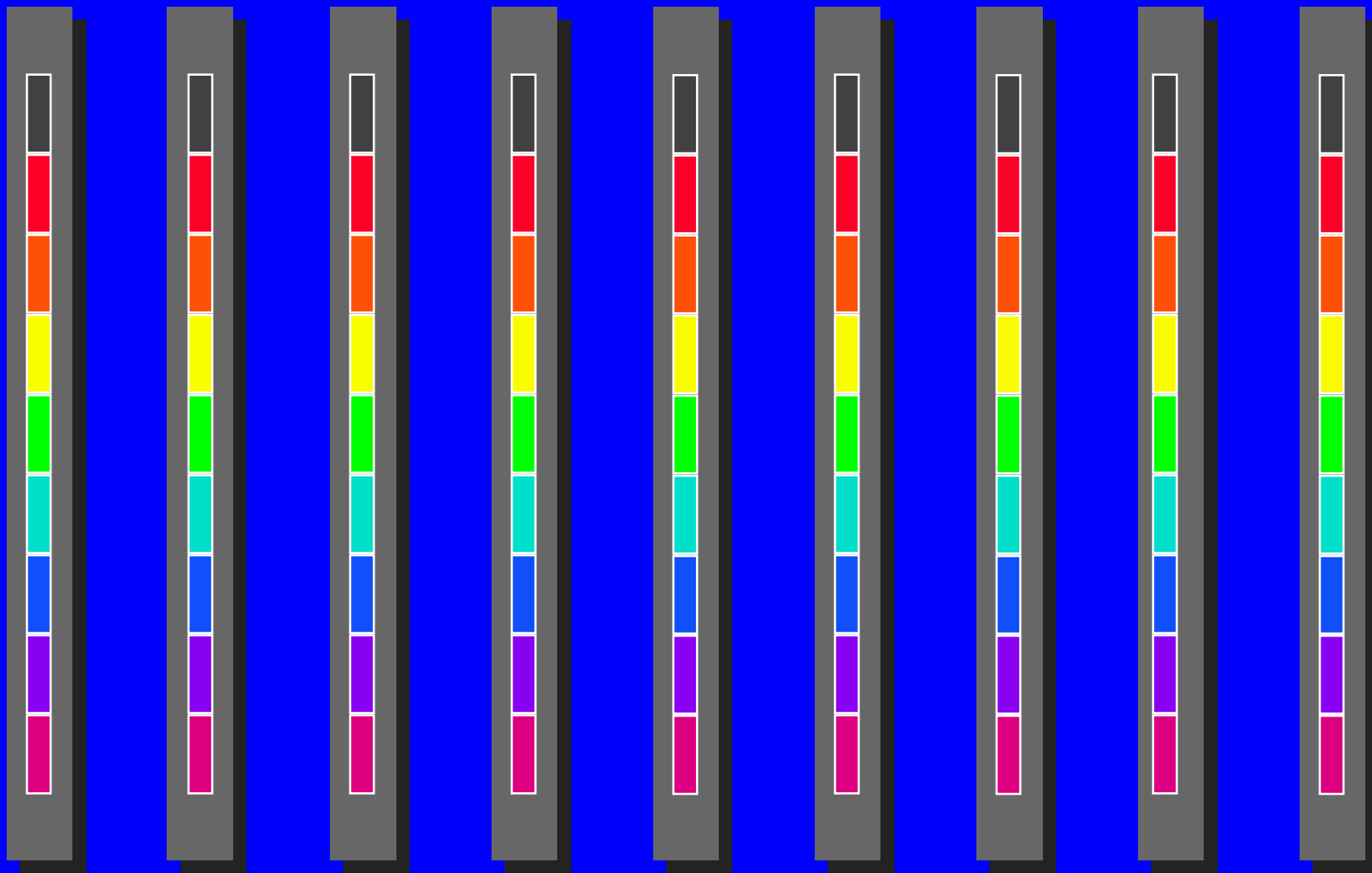












Cost of bucket Allgather

The diagram illustrates the cost of bucket Allgather. It features a central equation with two parts. The first part, $(p-1) \left(\alpha + \frac{n}{p} \beta \right)$, is enclosed in a yellow box. A blue arrow points from the text "number of steps" to this box. The second part, $(p-1) \alpha + \frac{p-1}{p} n \beta$, is enclosed in a red box. A red arrow points from the text "cost per steps" to this box. An equals sign is placed between the two expressions.

$$(p-1) \left(\alpha + \frac{n}{p} \beta \right) = (p-1) \alpha + \frac{p-1}{p} n \beta$$

number of steps

cost per steps

Cost of bucket Allgather

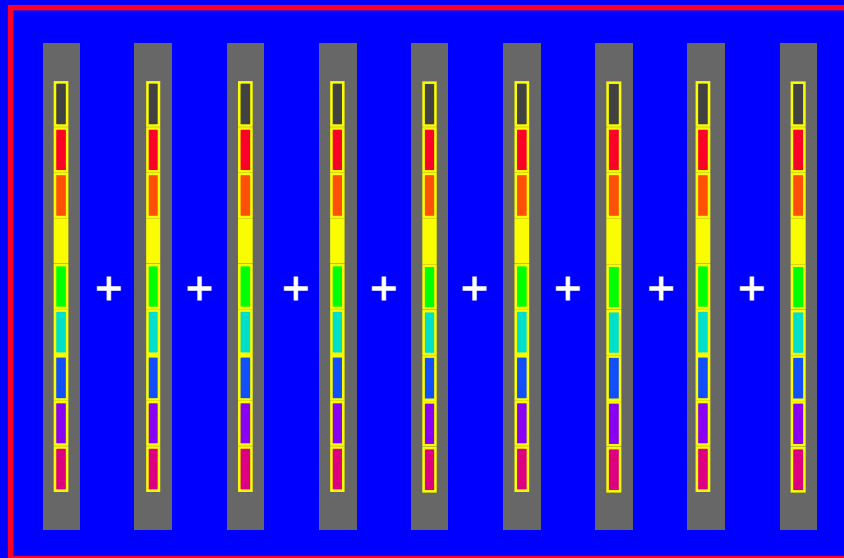
The diagram illustrates the cost of bucket Allgather. It features a central equation with two parts. The top part is $(p-1) \left(\alpha + \frac{n}{p} \beta \right)$. A blue box highlights $(p-1)$, with a blue arrow pointing to it from the label "number of steps". A red box highlights $\left(\alpha + \frac{n}{p} \beta \right)$, with a red arrow pointing to it from the label "cost per steps". Below this, an equals sign is followed by the expanded form of the equation: $(p-1)\alpha + \frac{p-1}{p}n\beta$.

$$(p-1) \left(\alpha + \frac{n}{p} \beta \right) = (p-1)\alpha + \frac{p-1}{p}n\beta$$

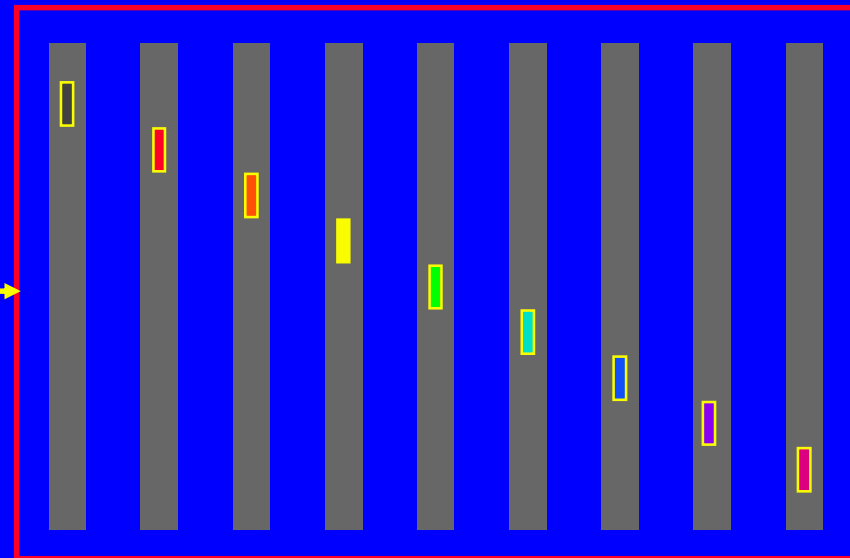
Notice: attains lower bound for bandwidth component

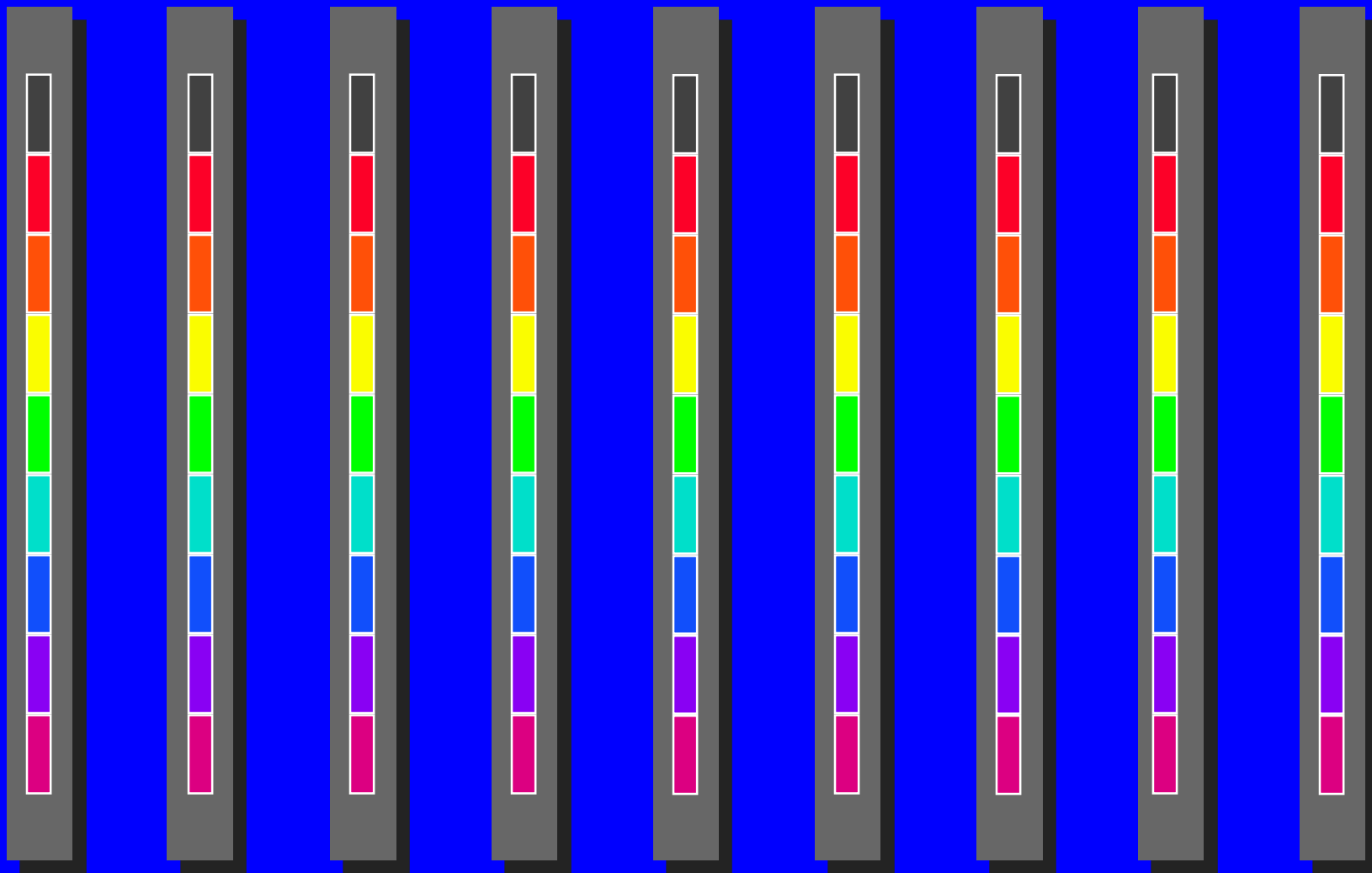
Reduce-scatter

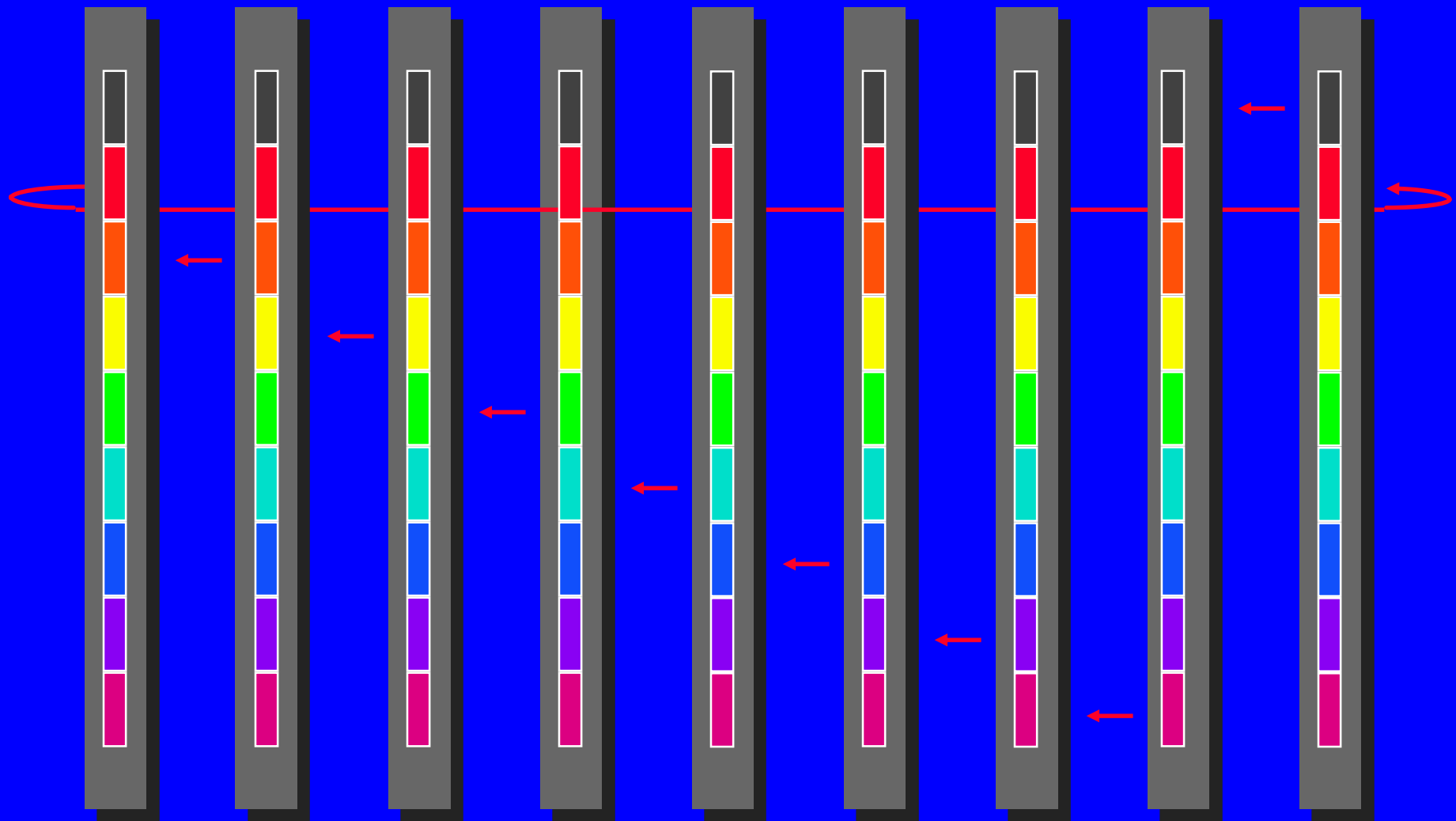
Before

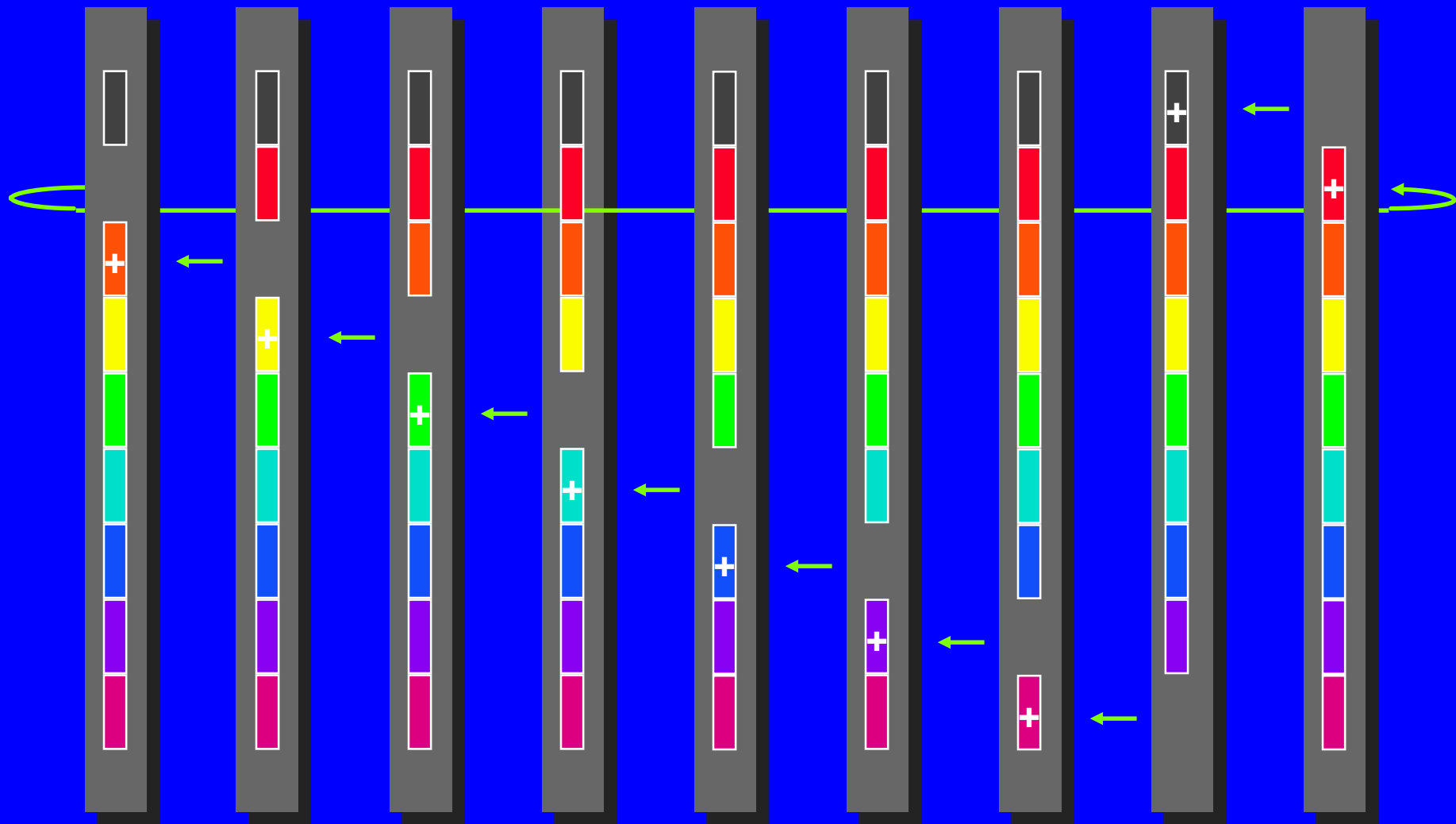


After

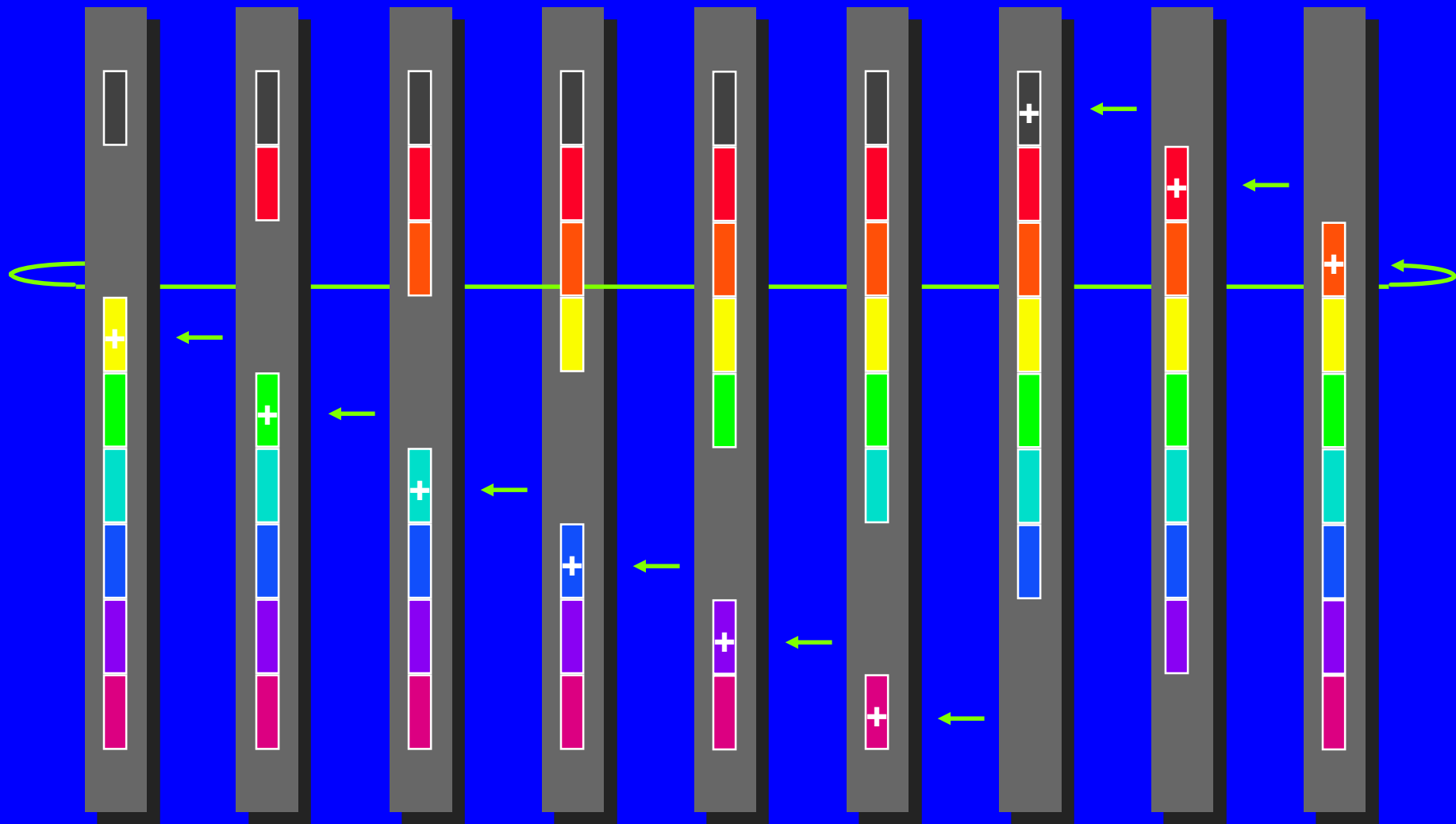


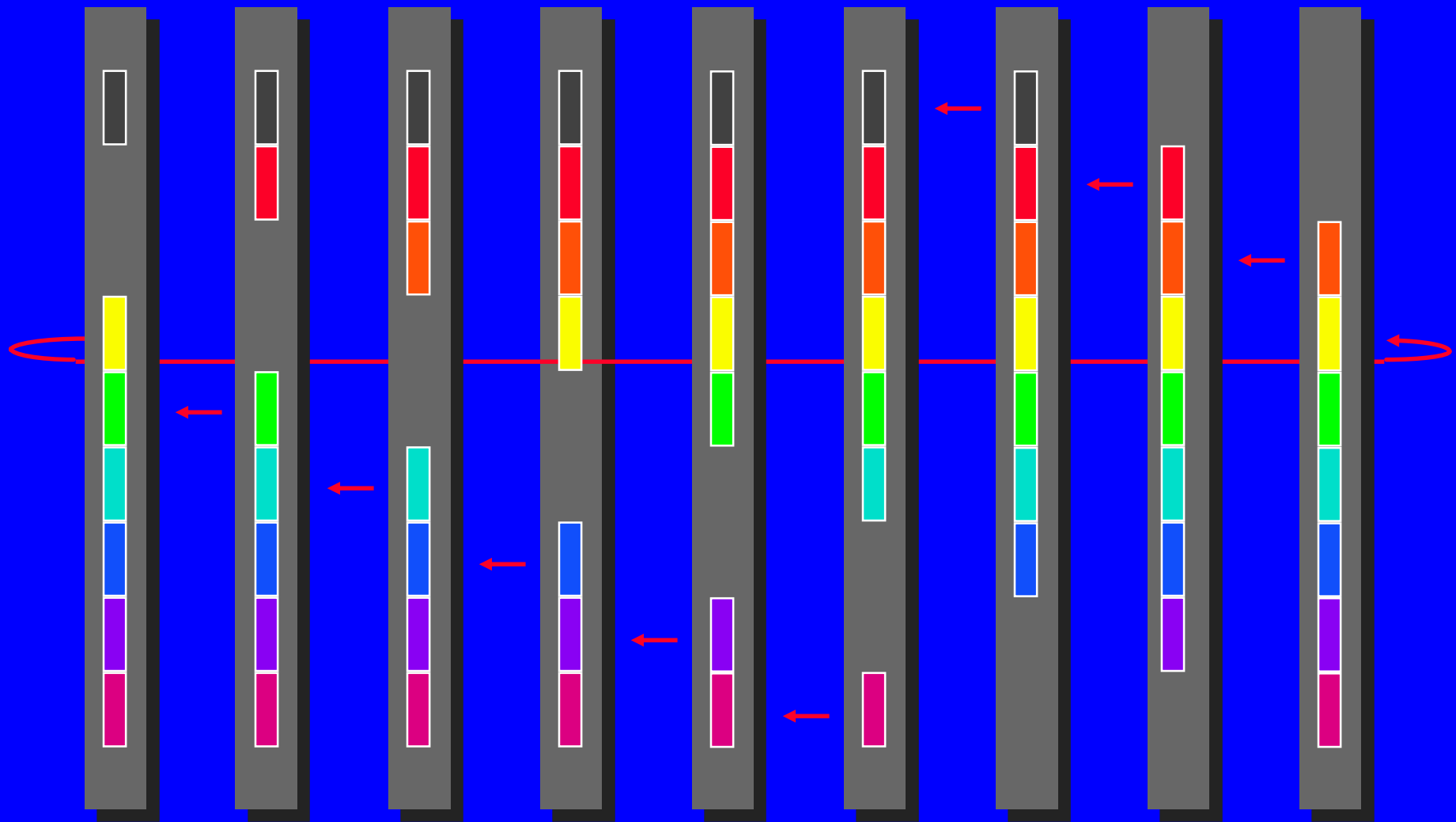


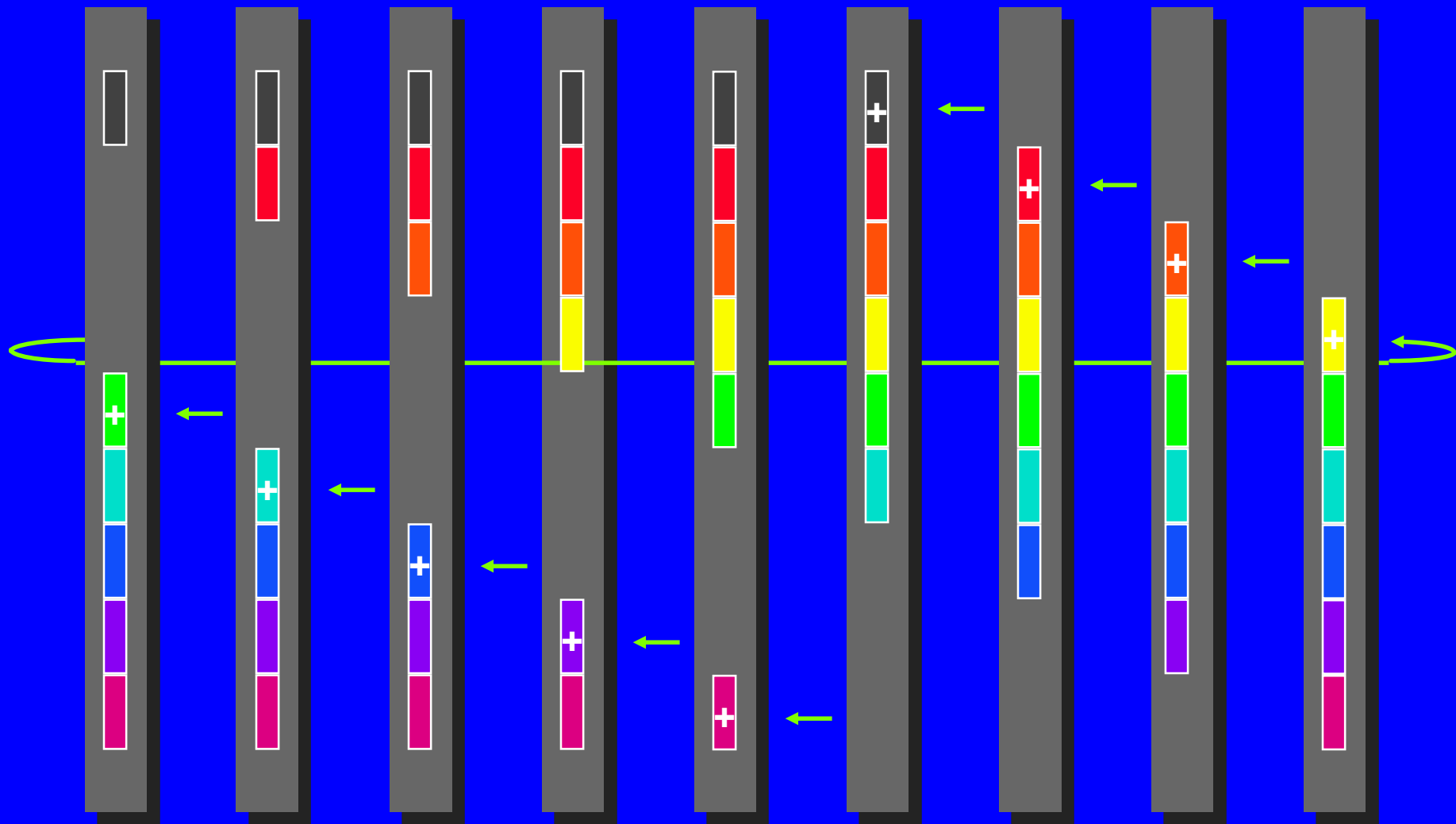


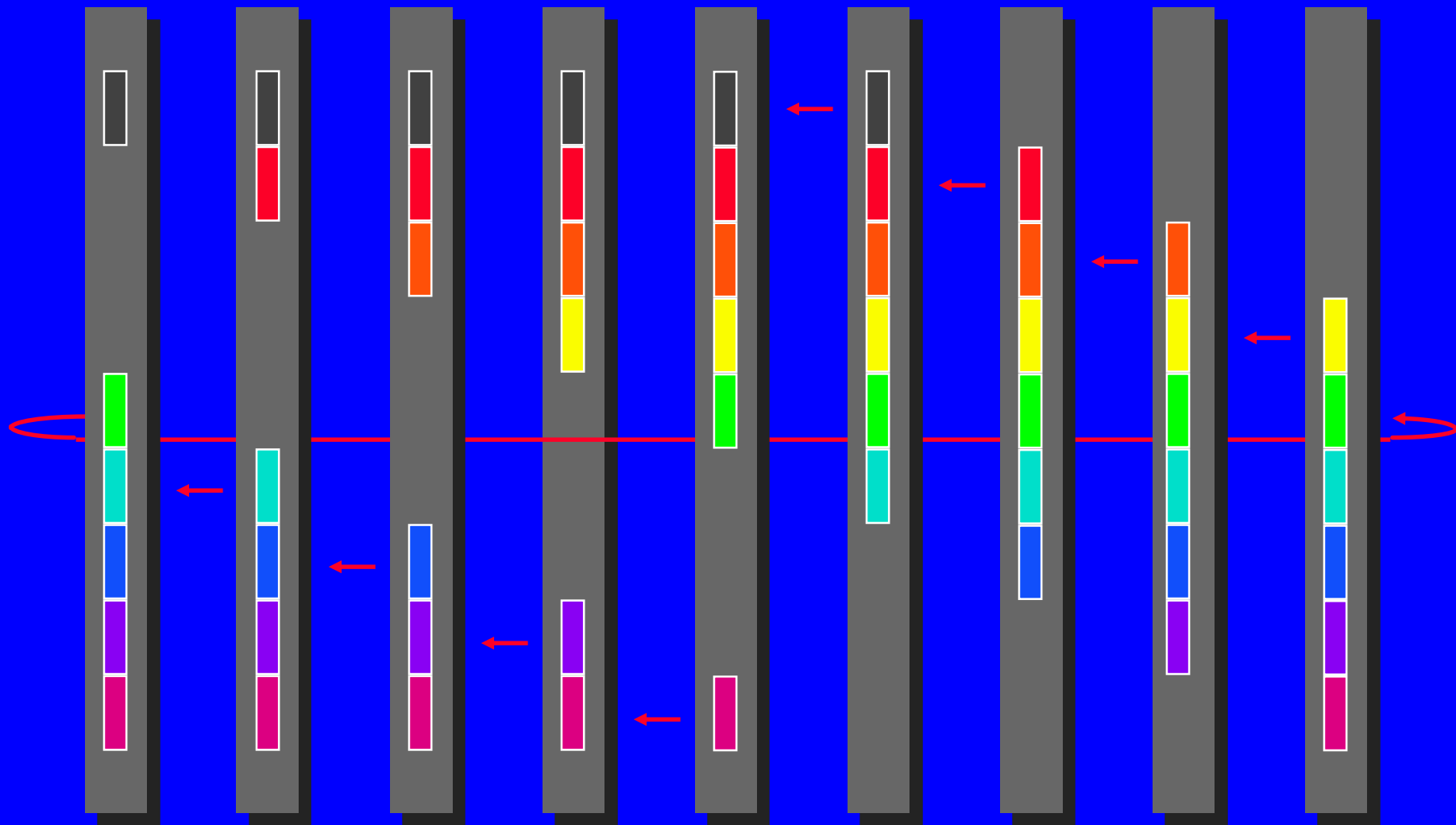


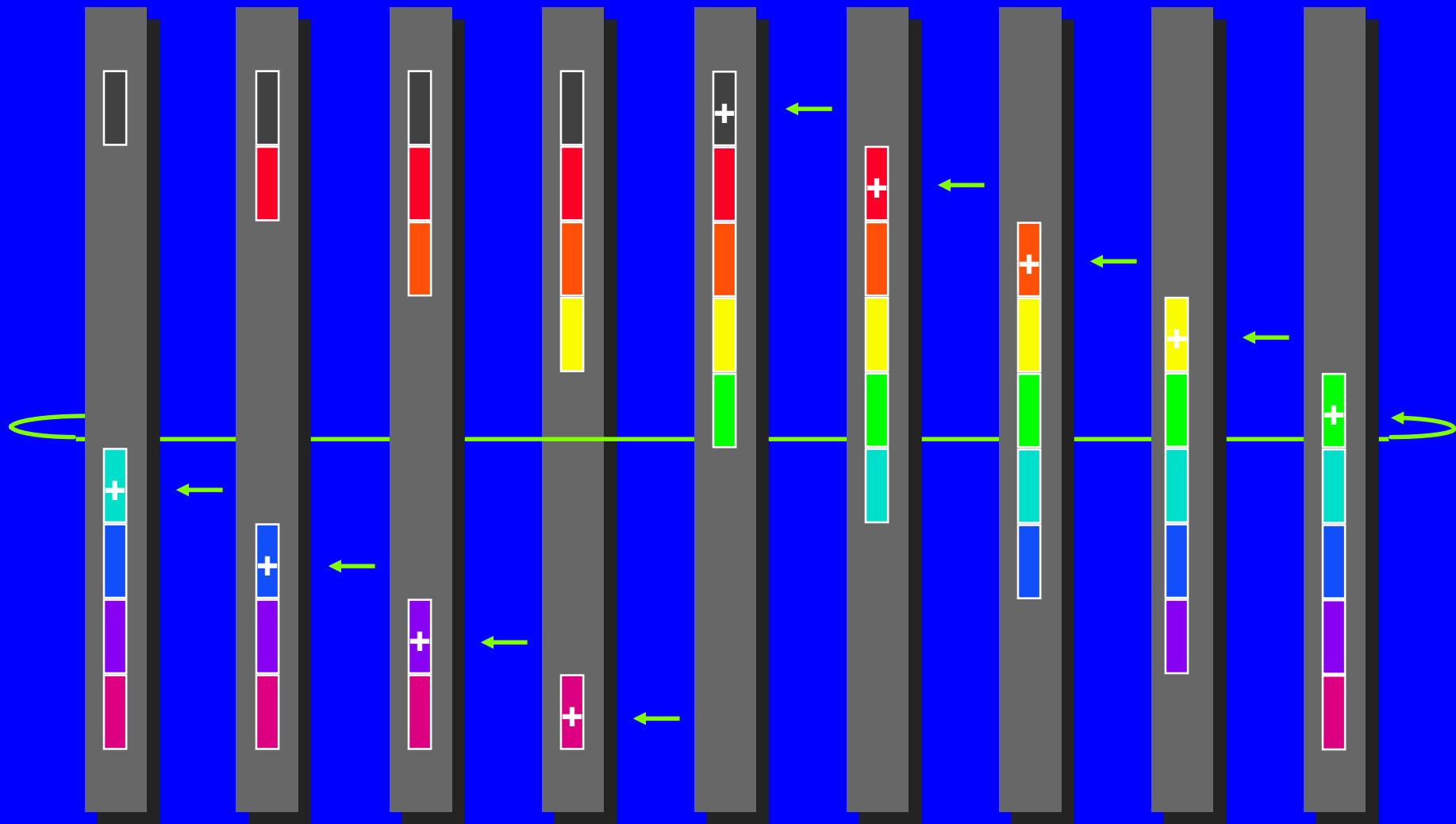


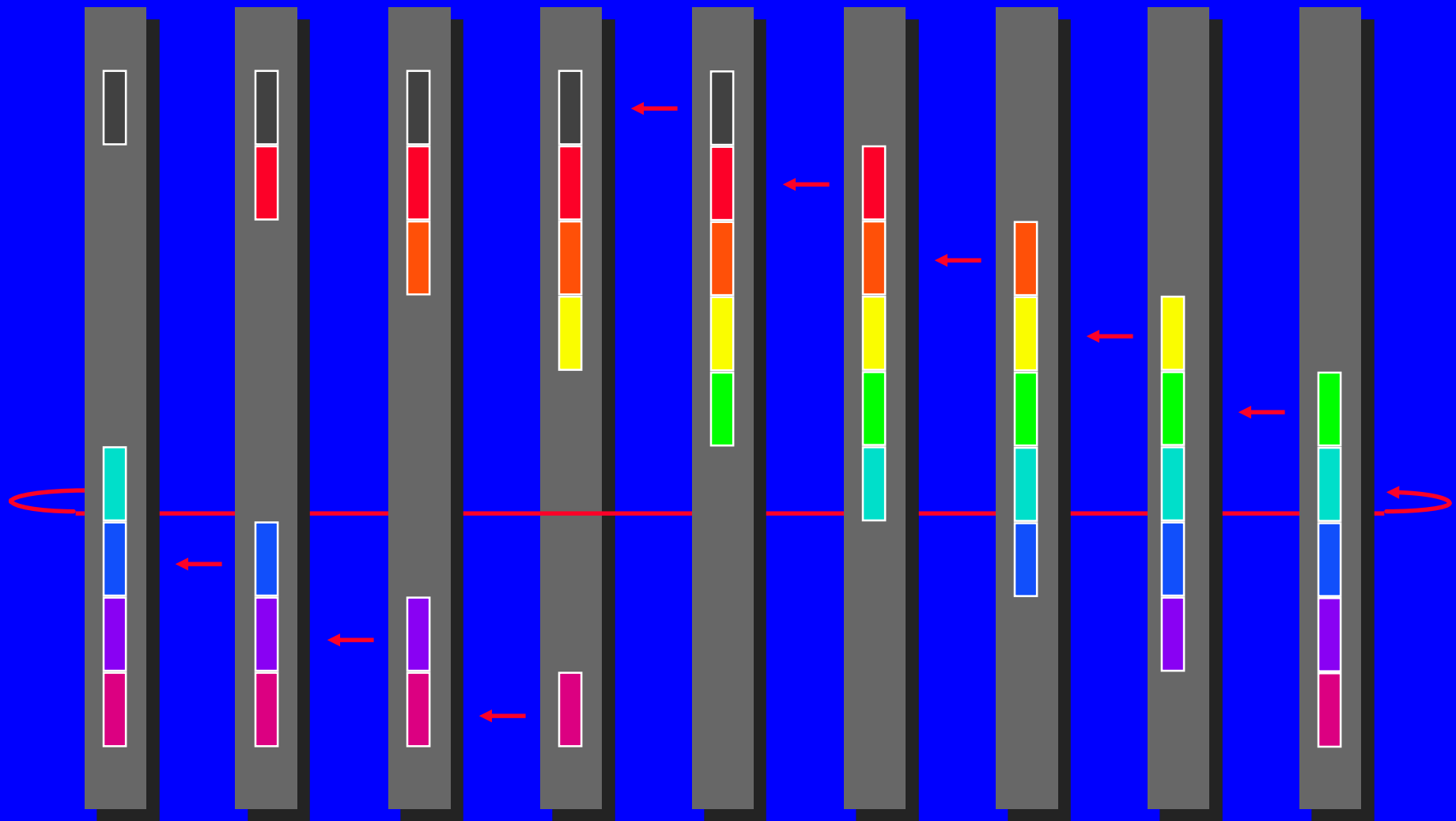


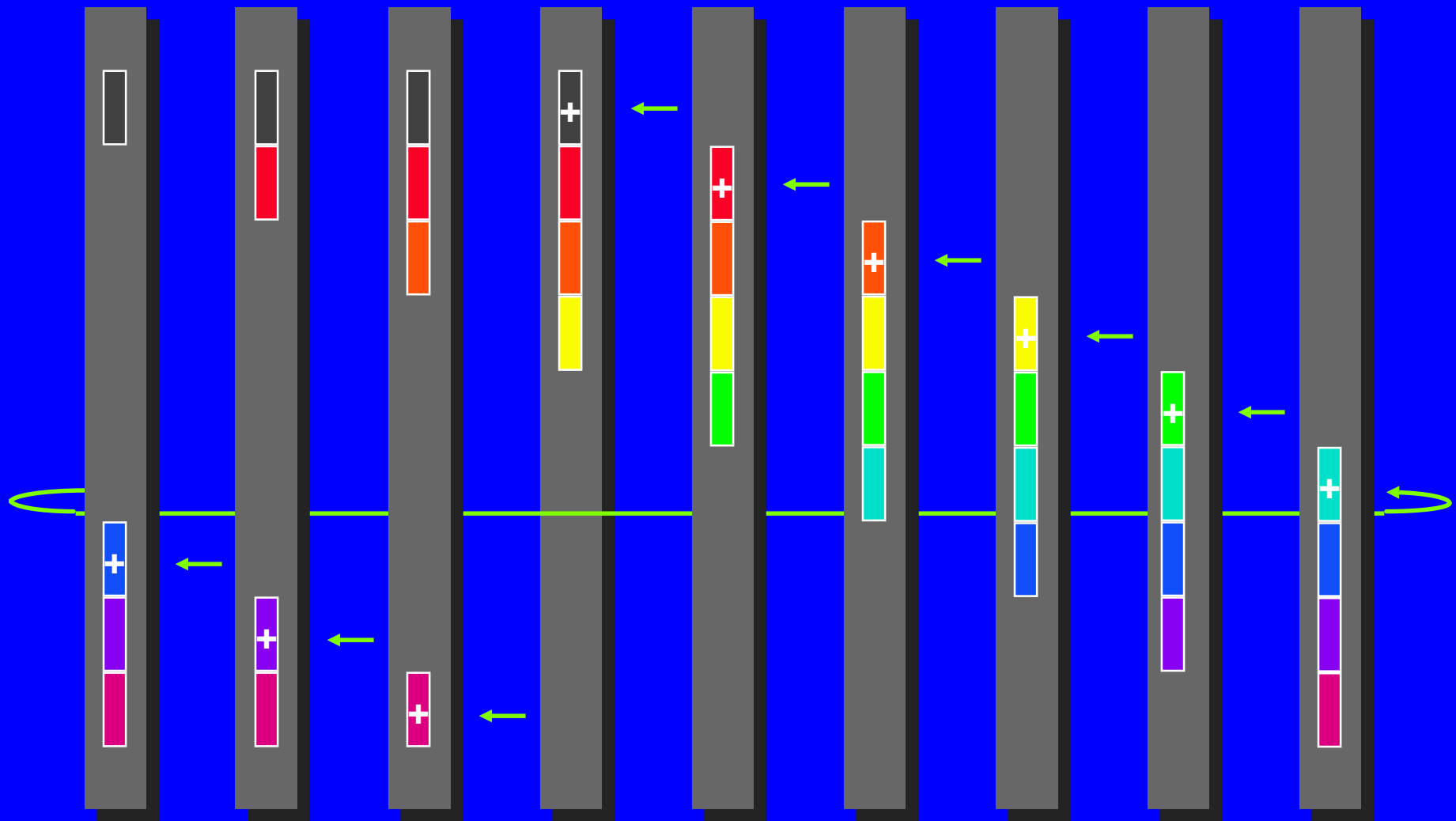


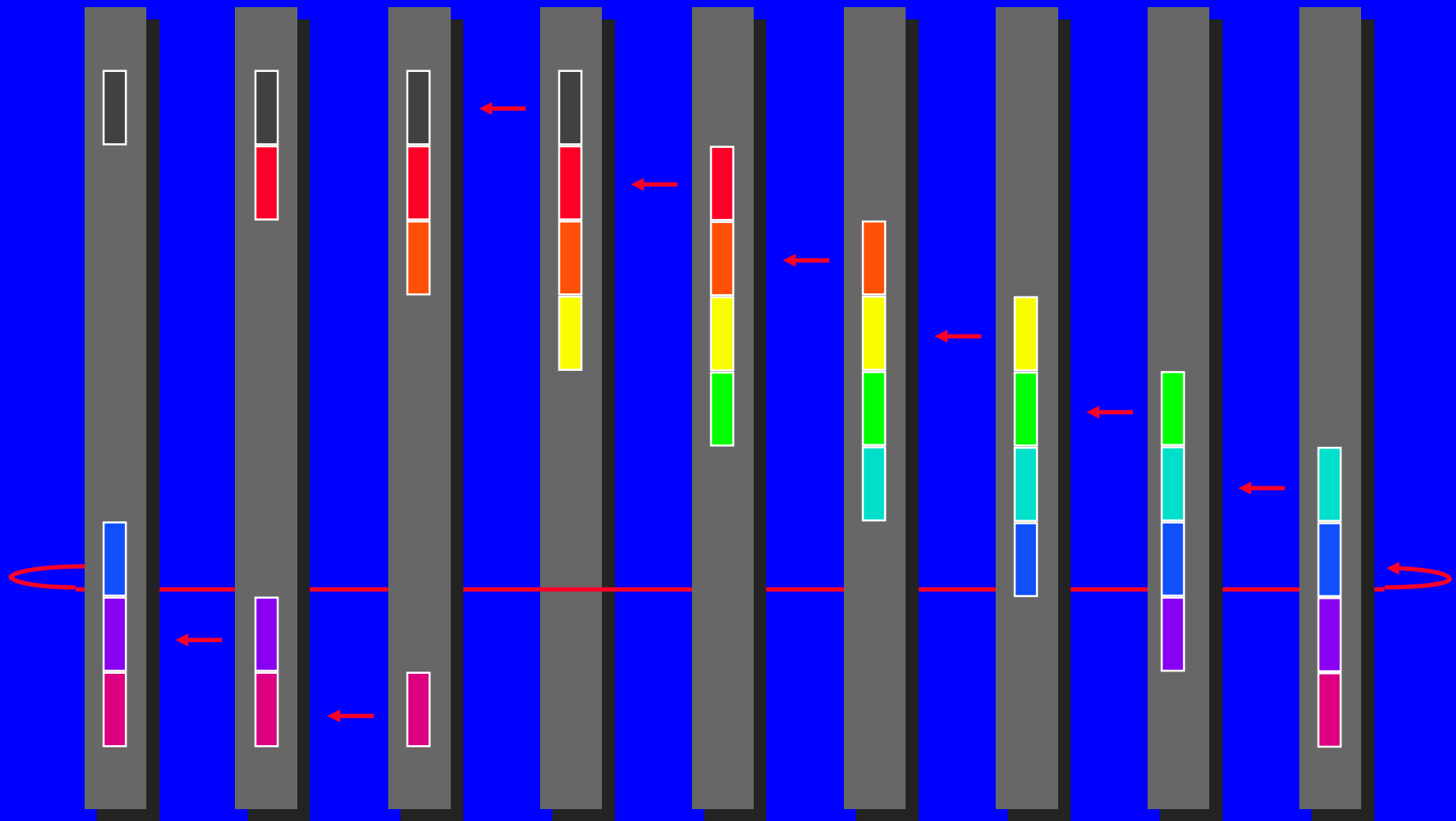


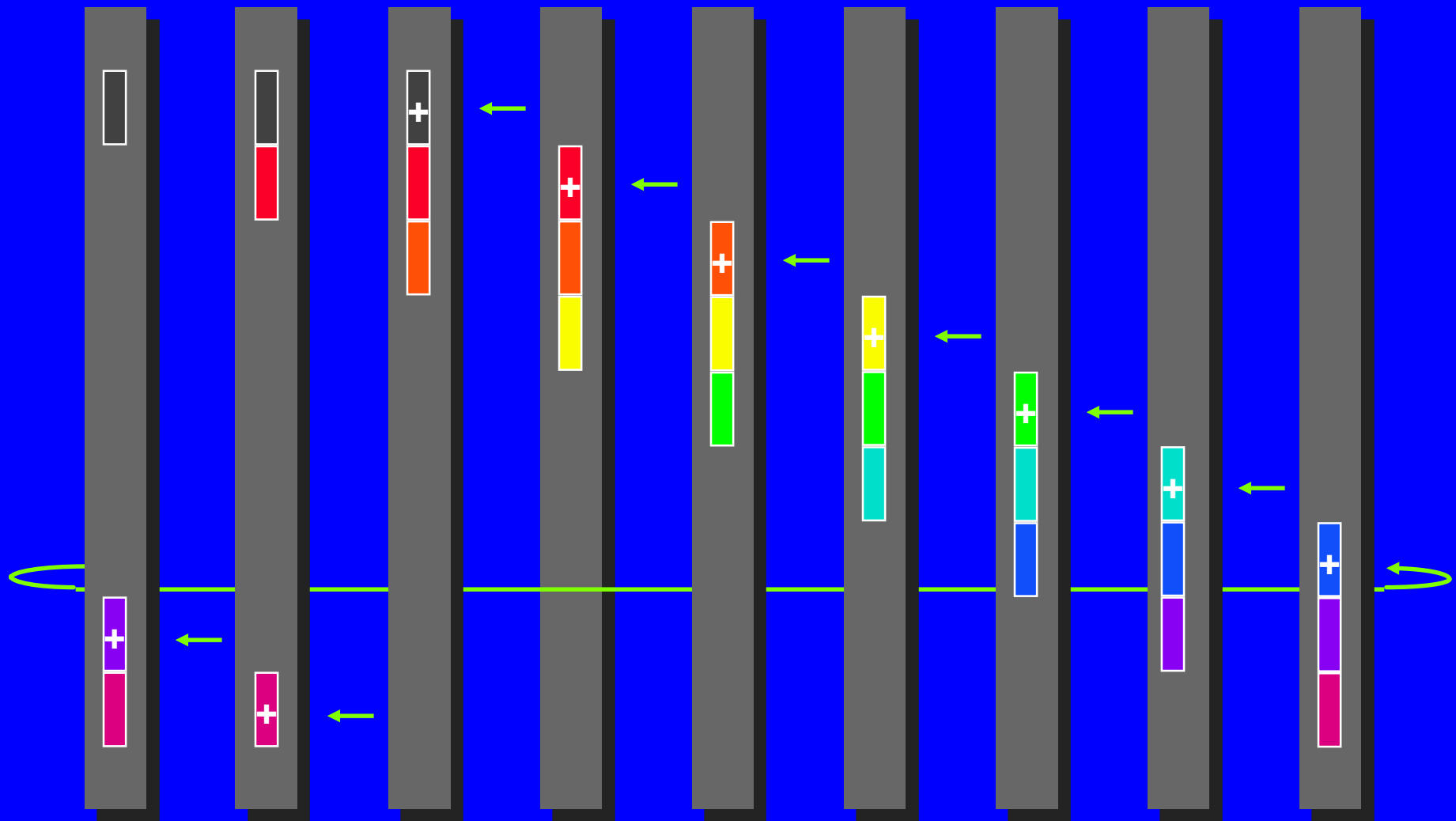


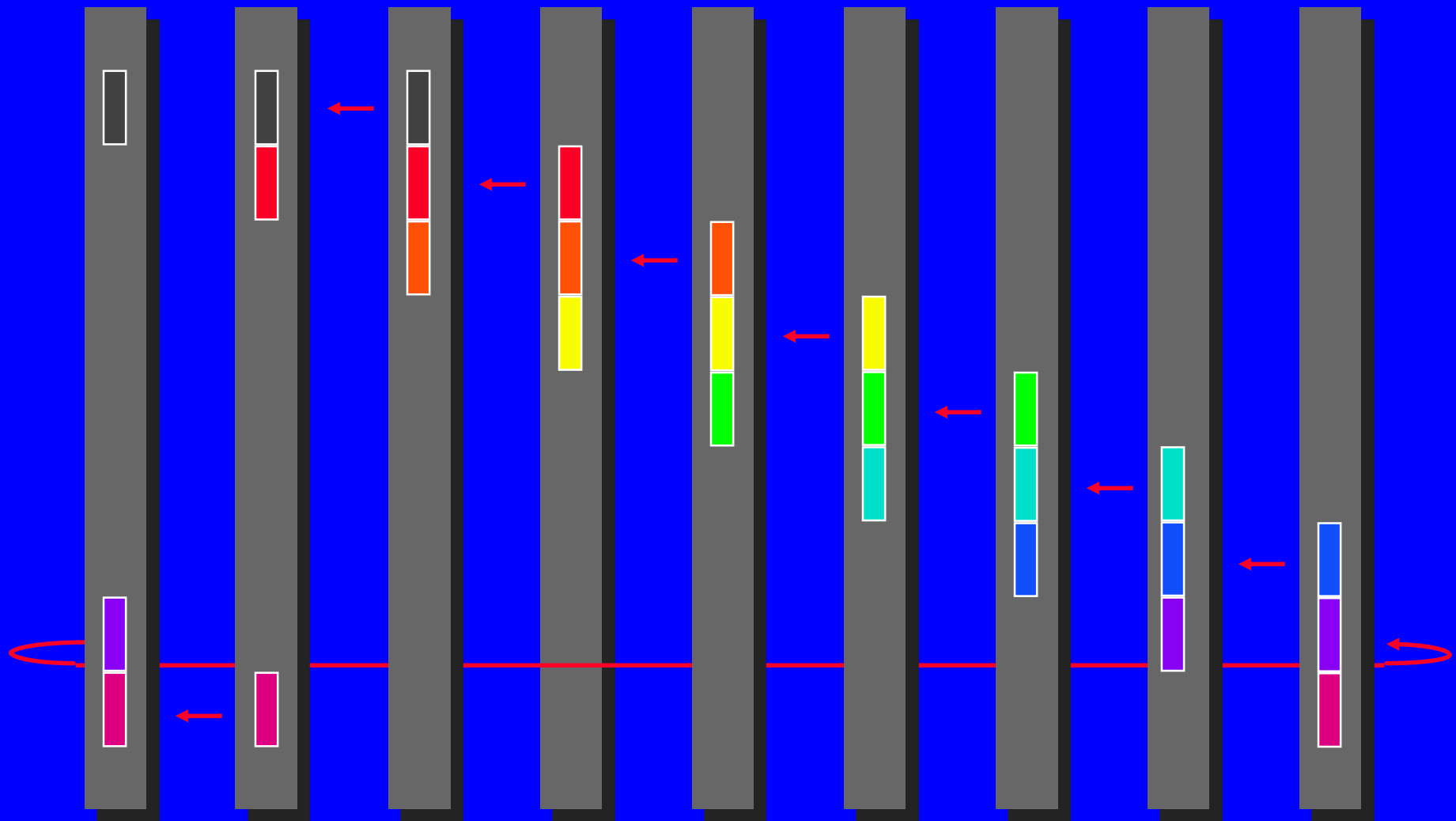


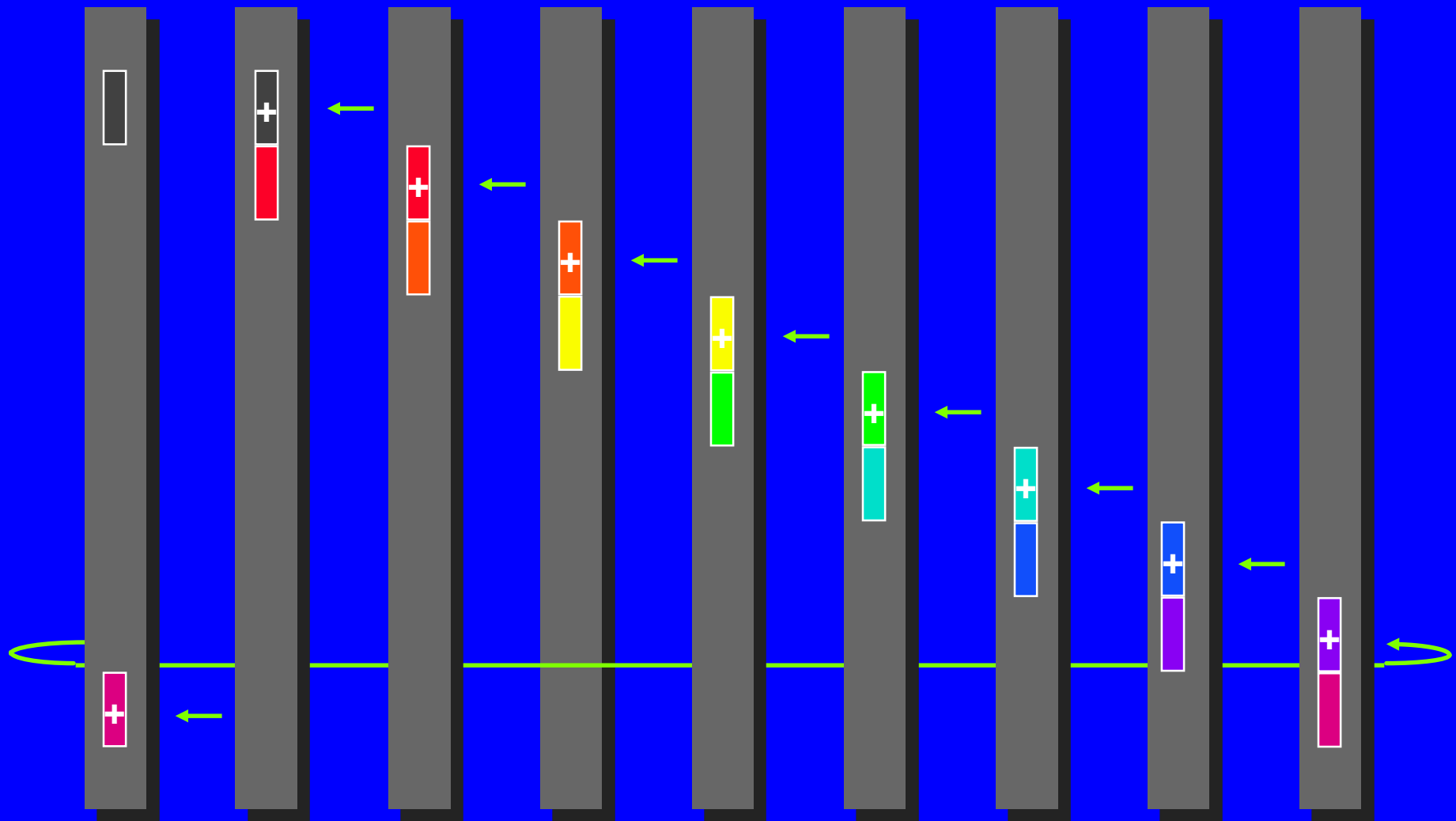


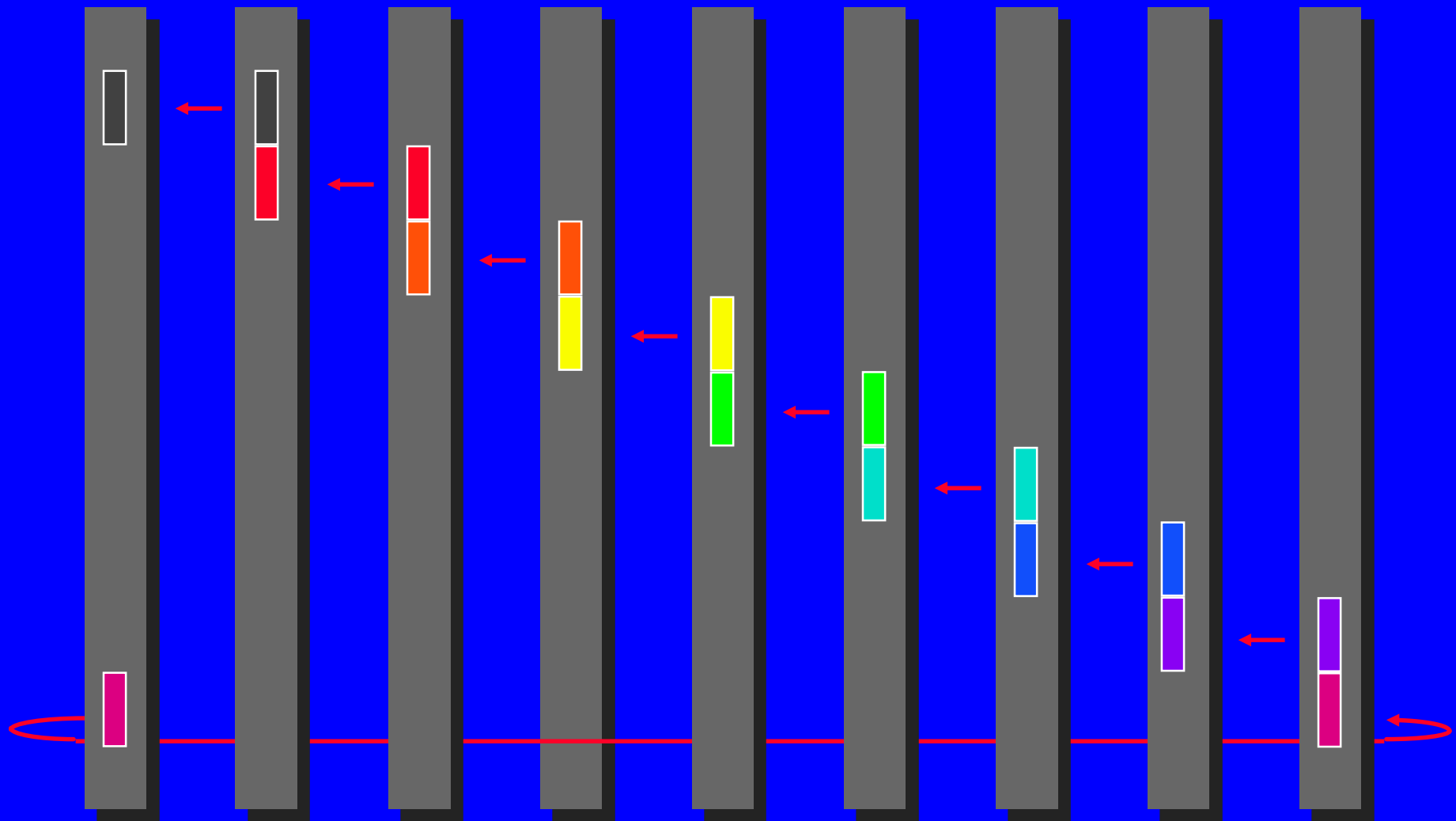


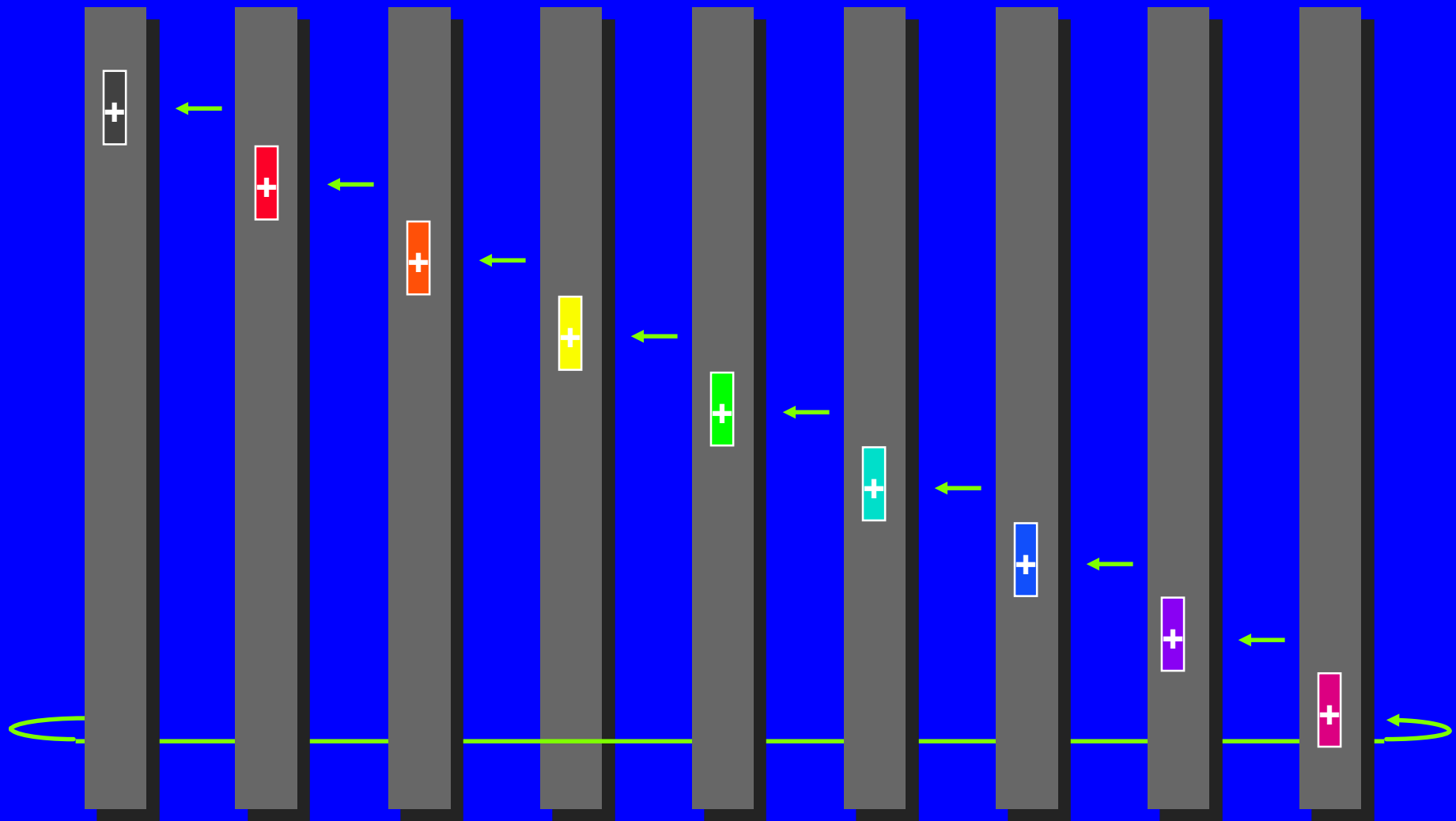


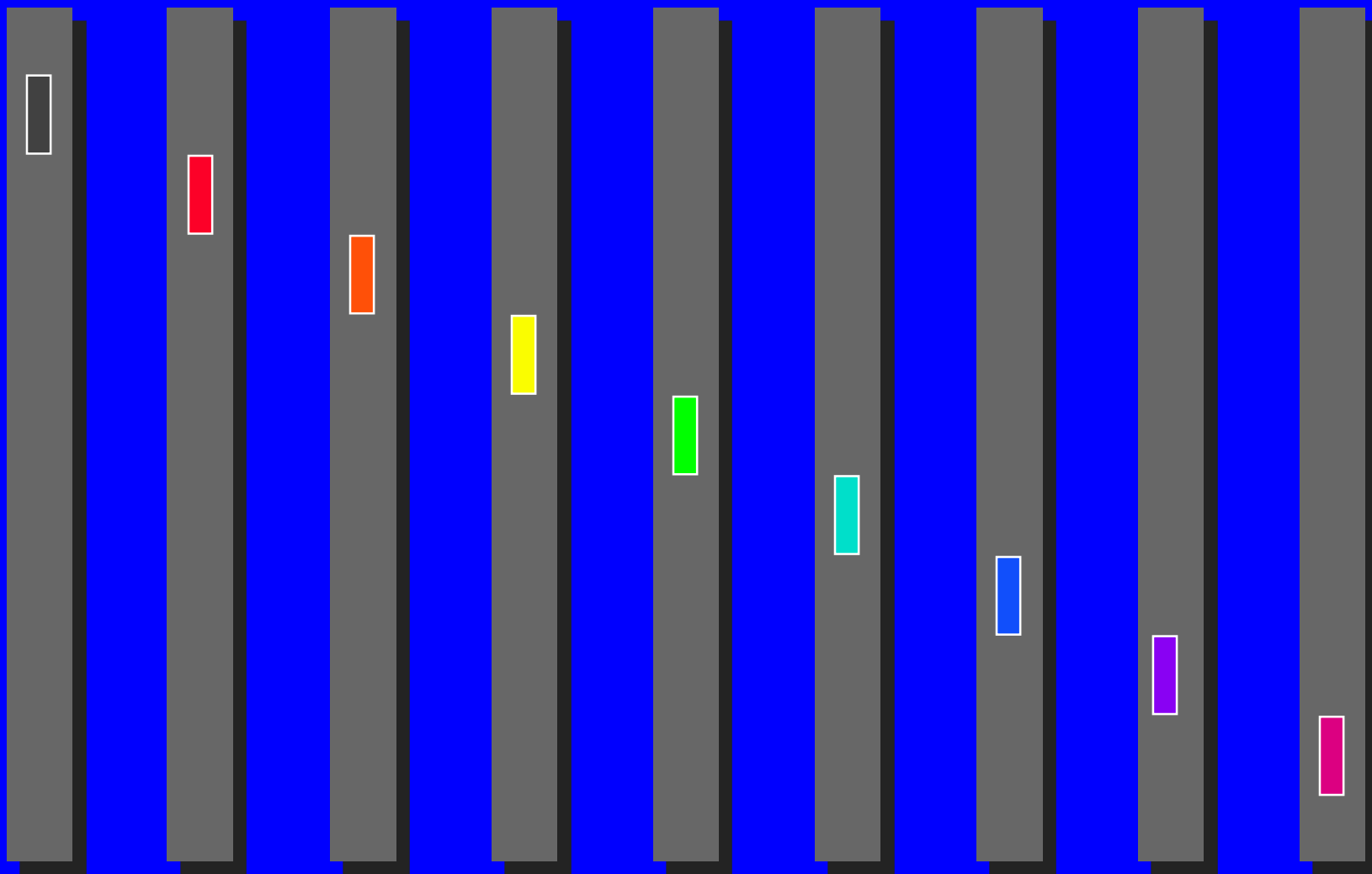












Cost of bucket distributed combine

The diagram illustrates the cost of bucket distributed combine. It features a central equation with two parts: a product of a number of steps and a cost per step, followed by an equals sign and a sum of three terms. Annotations with arrows point to the components of the equation.

$$(p-1) \left(\alpha + \frac{n}{p}\beta + \frac{n}{p}\beta \right) = (p-1)\alpha + \frac{p-1}{p}n\beta + \frac{p-1}{p}n\gamma$$

number of steps

cost per steps

Cost of bucket Reduce-scatter

The diagram illustrates the cost of bucket Reduce-scatter. It features a central equation with two parts: a product of a number of steps and a cost per step. The number of steps, $(p-1)$, is highlighted with a yellow box and labeled with a blue arrow. The cost per step, $\left(\alpha + \frac{n}{p} \beta + \frac{n}{p} \gamma \right)$, is highlighted with a red box and labeled with a red arrow. Below these, the expanded equation is shown: $(p-1)\alpha + \frac{p-1}{p}n\beta + \frac{p-1}{p}n\gamma$.

$$(p-1) \left(\alpha + \frac{n}{p} \beta + \frac{n}{p} \gamma \right) = (p-1)\alpha + \frac{p-1}{p}n\beta + \frac{p-1}{p}n\gamma$$

number of steps

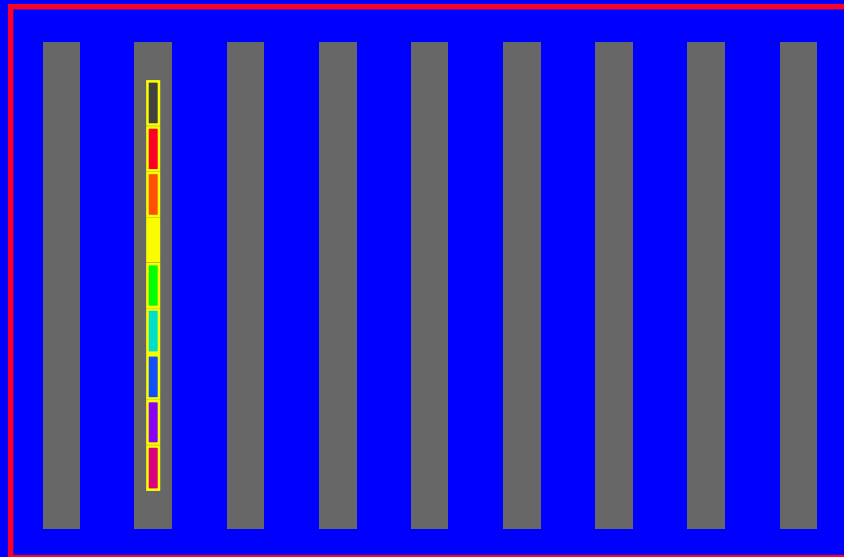
cost per steps

Notice: attains lower bound for bandwidth and computation component

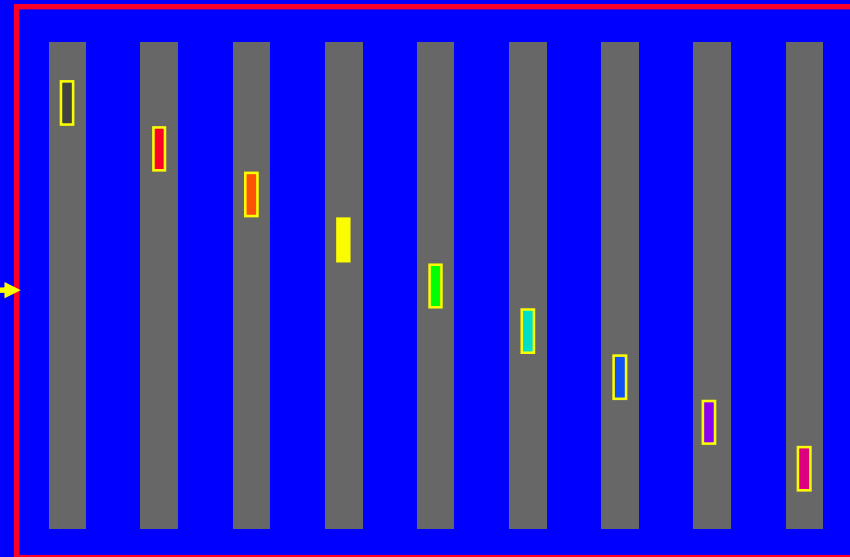
Scatter

Notice: Scatter as implemented before was optimal in latency and bandwidth components

Before



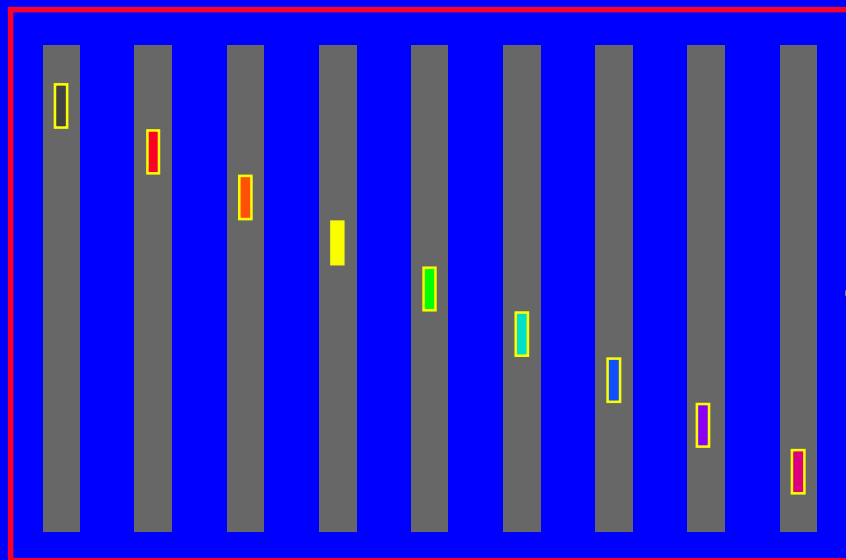
After



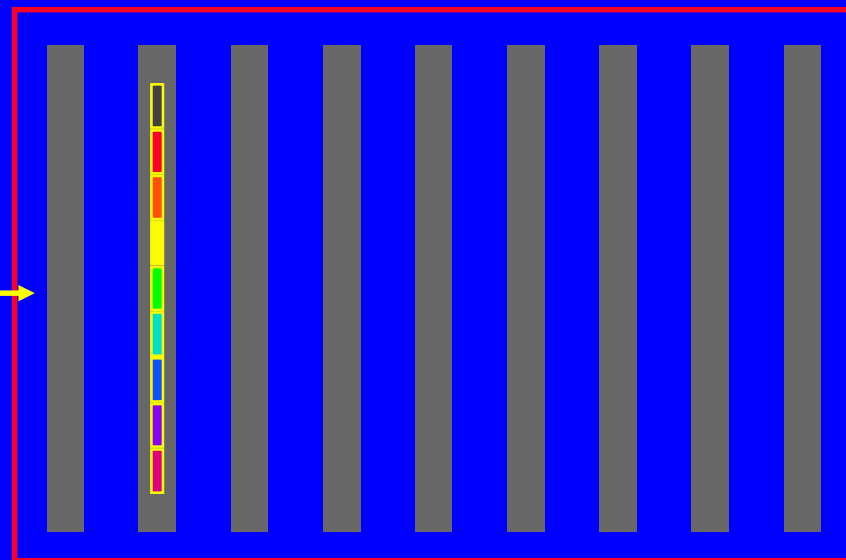
Gather

Notice: Gather as implemented before was optimal in **latency** and bandwidth components

Before

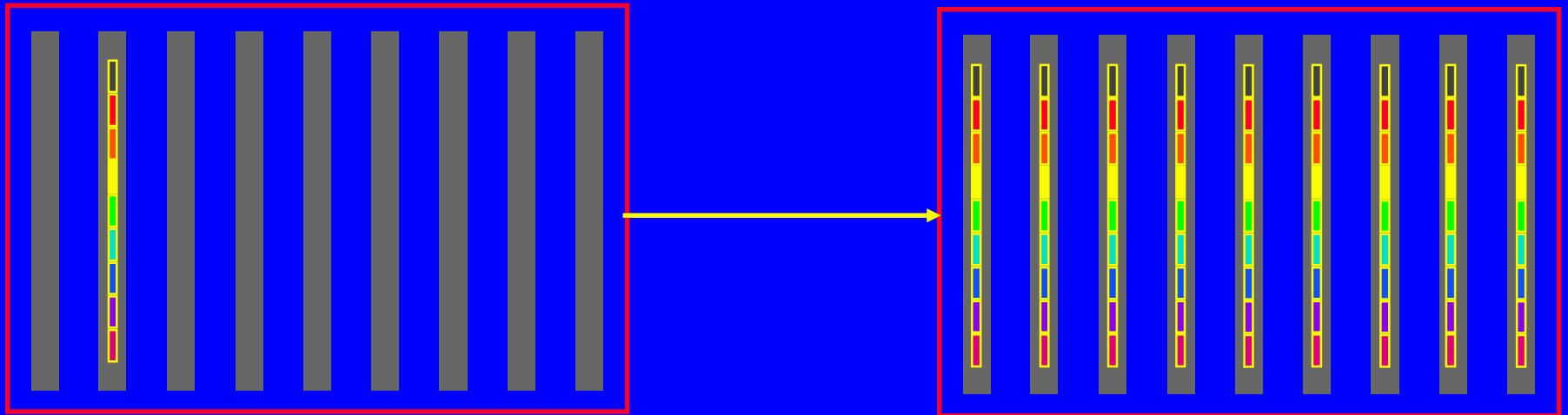


After

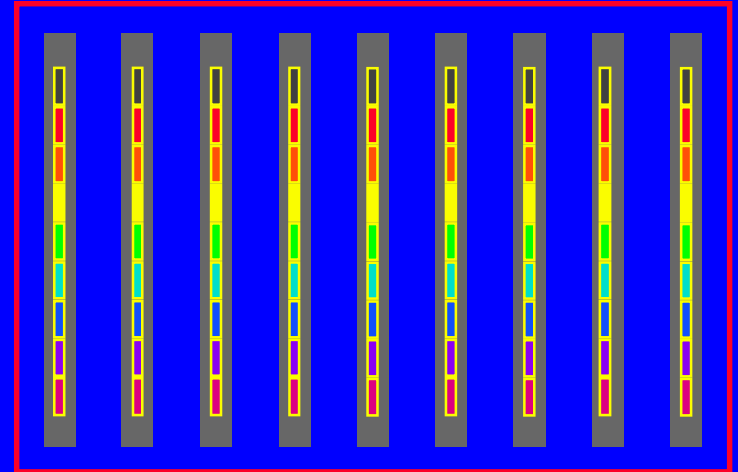
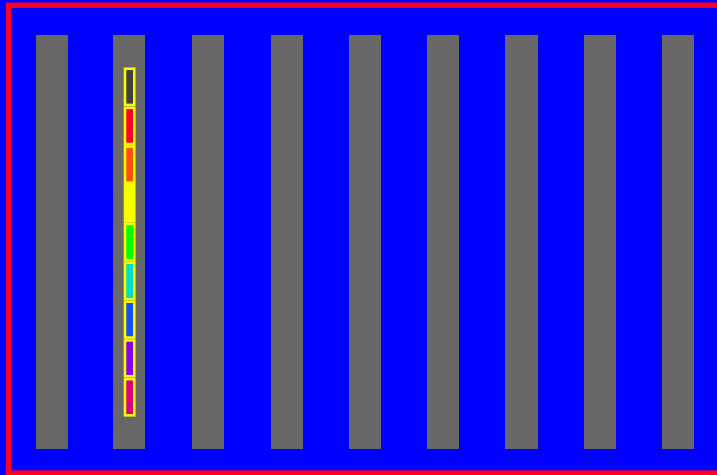


Using the building blocks

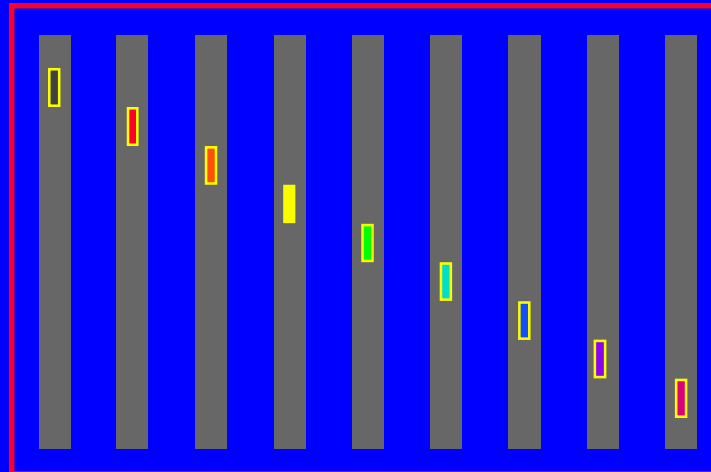
Broadcast (long vector)



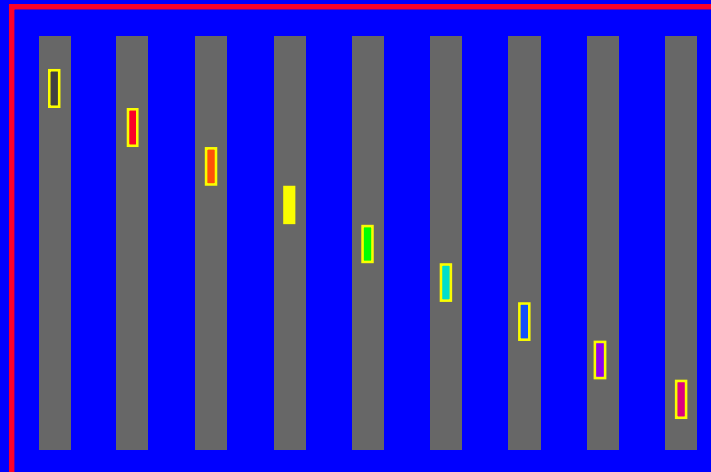
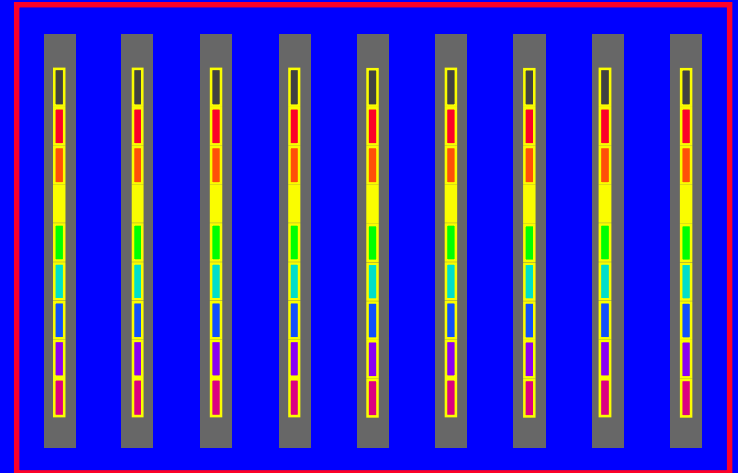
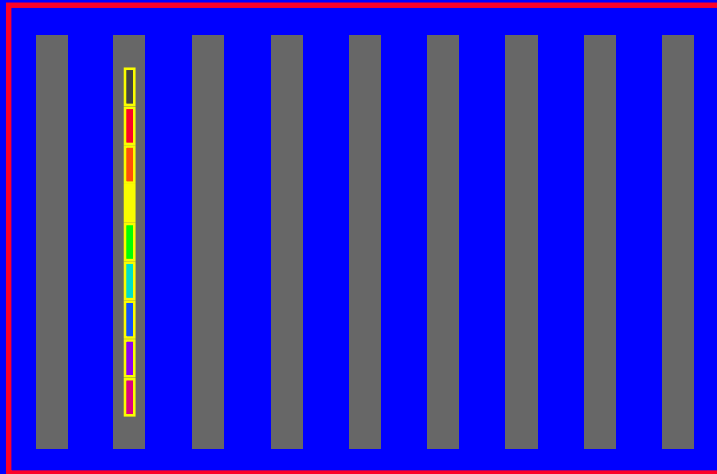
Broadcast (long vector)



Scatter



Broadcast (long vector)



Allgather

Cost of scatter/allgather broadcast

- Assumption: power of two number of nodes

scatter

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

allgather

$$(p-1)\alpha + \frac{p-1}{p}n\beta$$

$$(\log(p) + p - 1)\alpha + 2\frac{p-1}{p}n\beta$$

Cost of scatter/allgather broadcast

- Assumption: power of two number of nodes

scatter

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

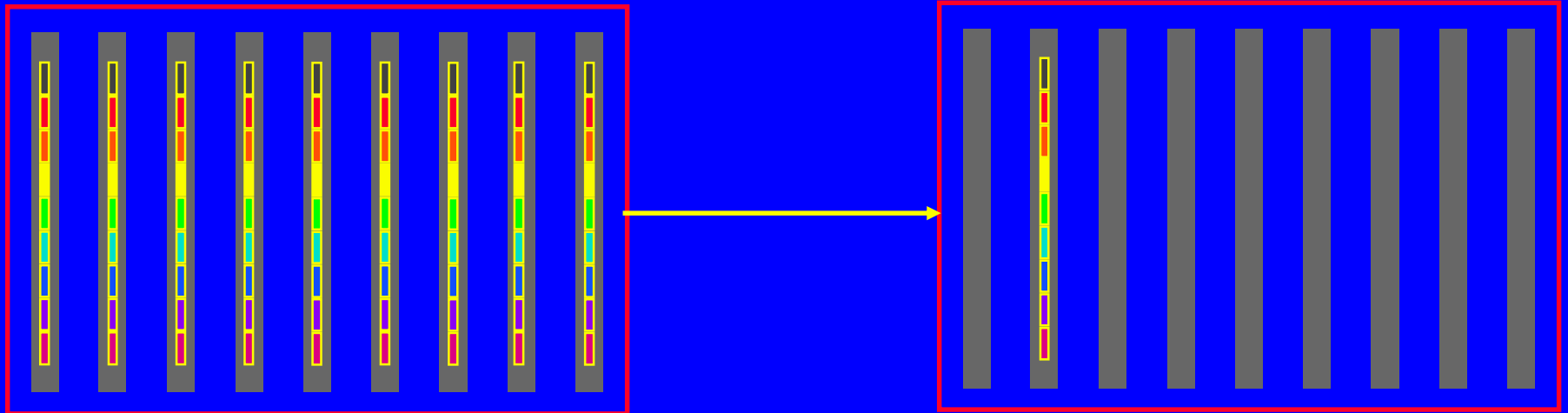
allgather

$$(p-1)\alpha + \frac{p-1}{p}n\beta$$

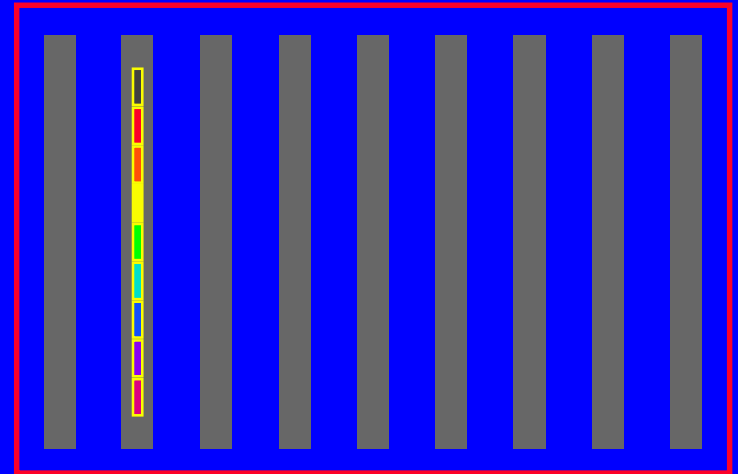
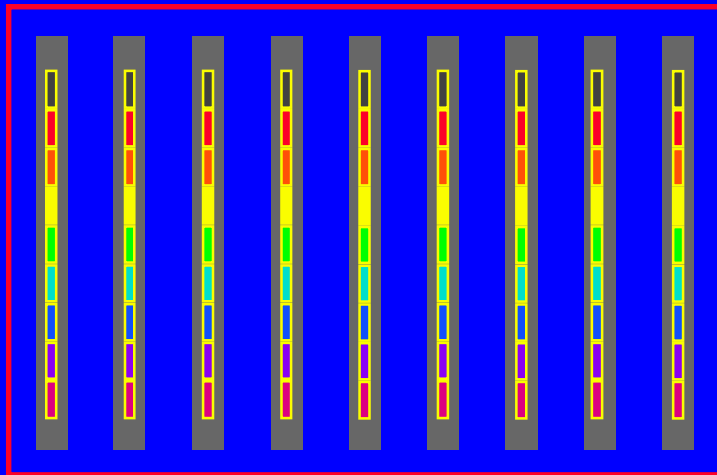
$$(\log(p) + p - 1)\alpha + 2\frac{p-1}{p}n\beta$$

Notice: attains within a factor of two of the lower bound for bandwidth

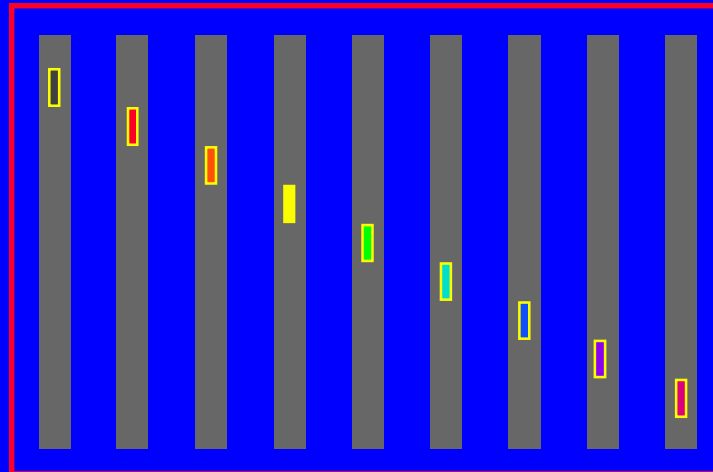
Reduce(-to-one) (long vector)



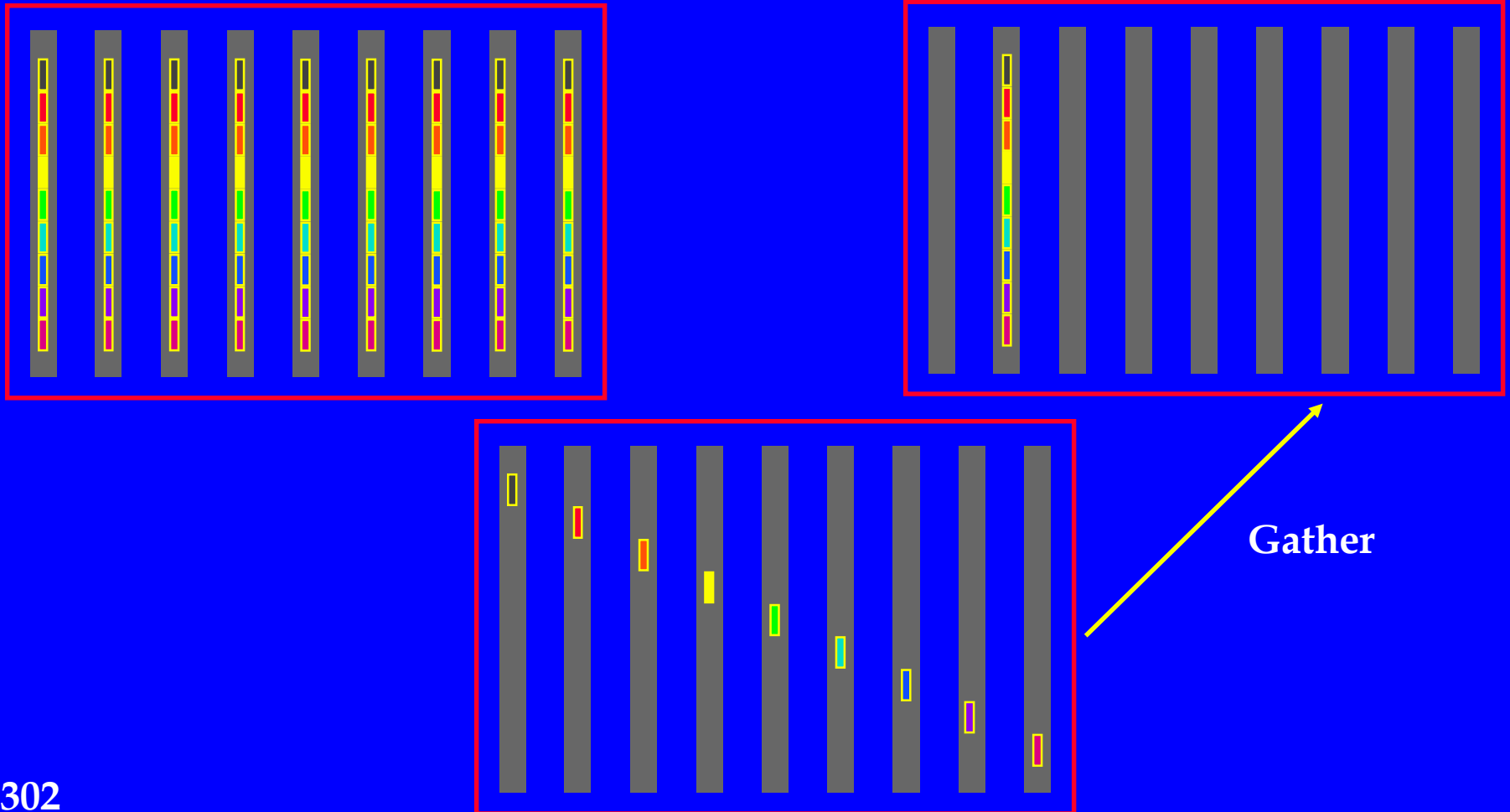
Combine-to-one (long vector)



Reduce-scatter



Combine-to-one (long vector)



Cost of Reduce-scatter/Gather

Reduce(-to-one)

- Assumption: power of two number of nodes

$$\begin{array}{l}
 \text{Reduce-scatter} \quad (p-1)\alpha + \frac{p-1}{p}n\beta + \frac{p-1}{p}n\gamma \\
 \text{gather} \quad \log(p)\alpha + \frac{p-1}{p}n\beta \\
 \hline
 (\log(p) + p-1)\alpha + 2\frac{p-1}{p}n\beta + \frac{p-1}{p}n\gamma
 \end{array}$$

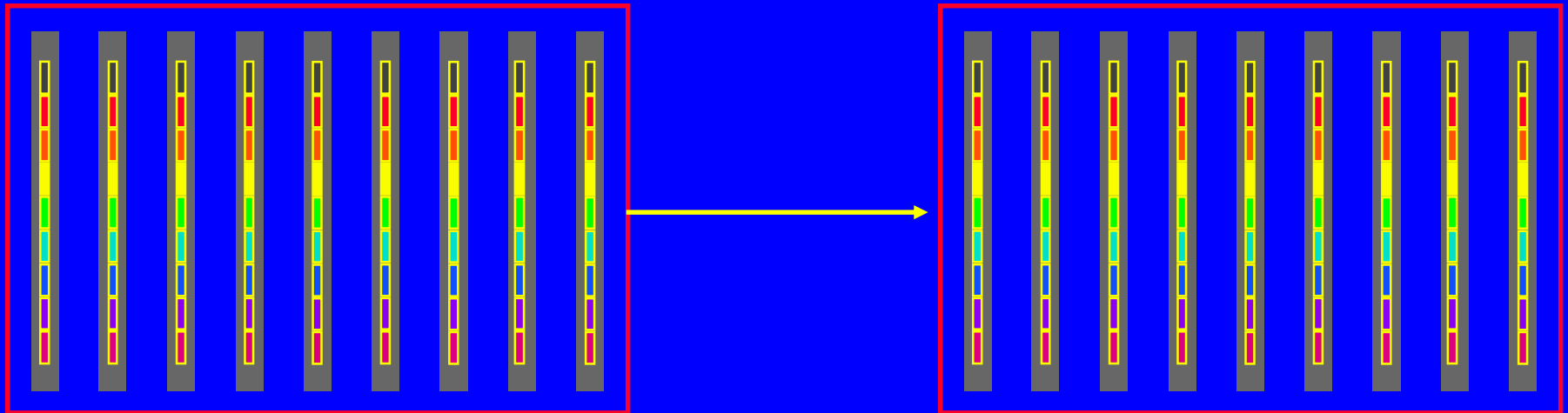
Cost of Reduce-scatter/Gather Reduce(-to-one)

- Assumption: power of two number of nodes

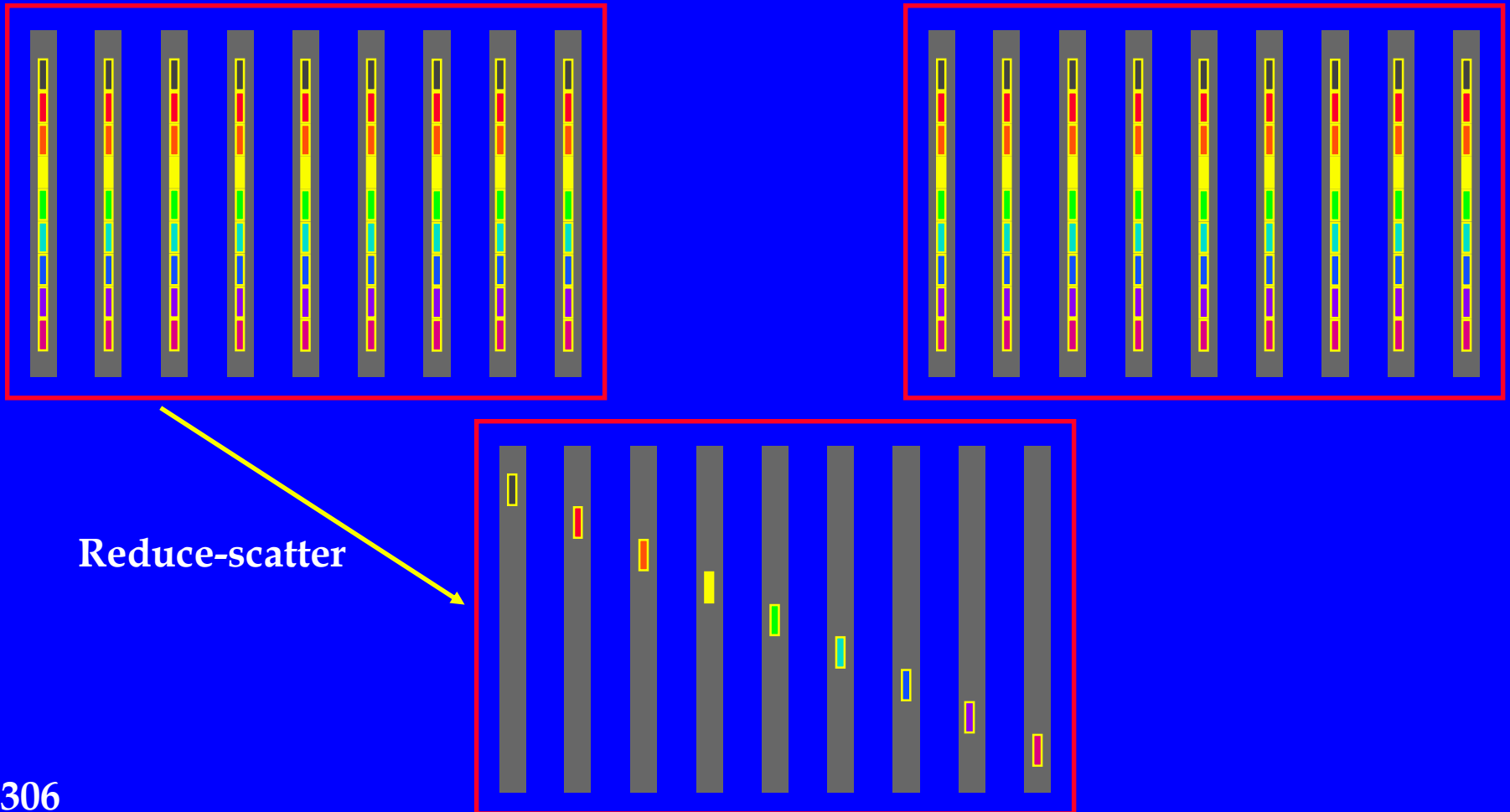
$$\begin{array}{l}
 \text{Reduce-scatter} \quad (p-1)\alpha + \frac{p-1}{p}n\beta + \frac{p-1}{p}n\gamma \\
 \text{gather} \quad \log(p)\alpha + \frac{p-1}{p}n\beta \\
 \hline
 (\log(p) + p-1)\alpha + 2\frac{p-1}{p}n\beta + \frac{p-1}{p}n\gamma
 \end{array}$$

Notice: attains within a factor of two of the lower bound for bandwidth and attains lower bound for computation

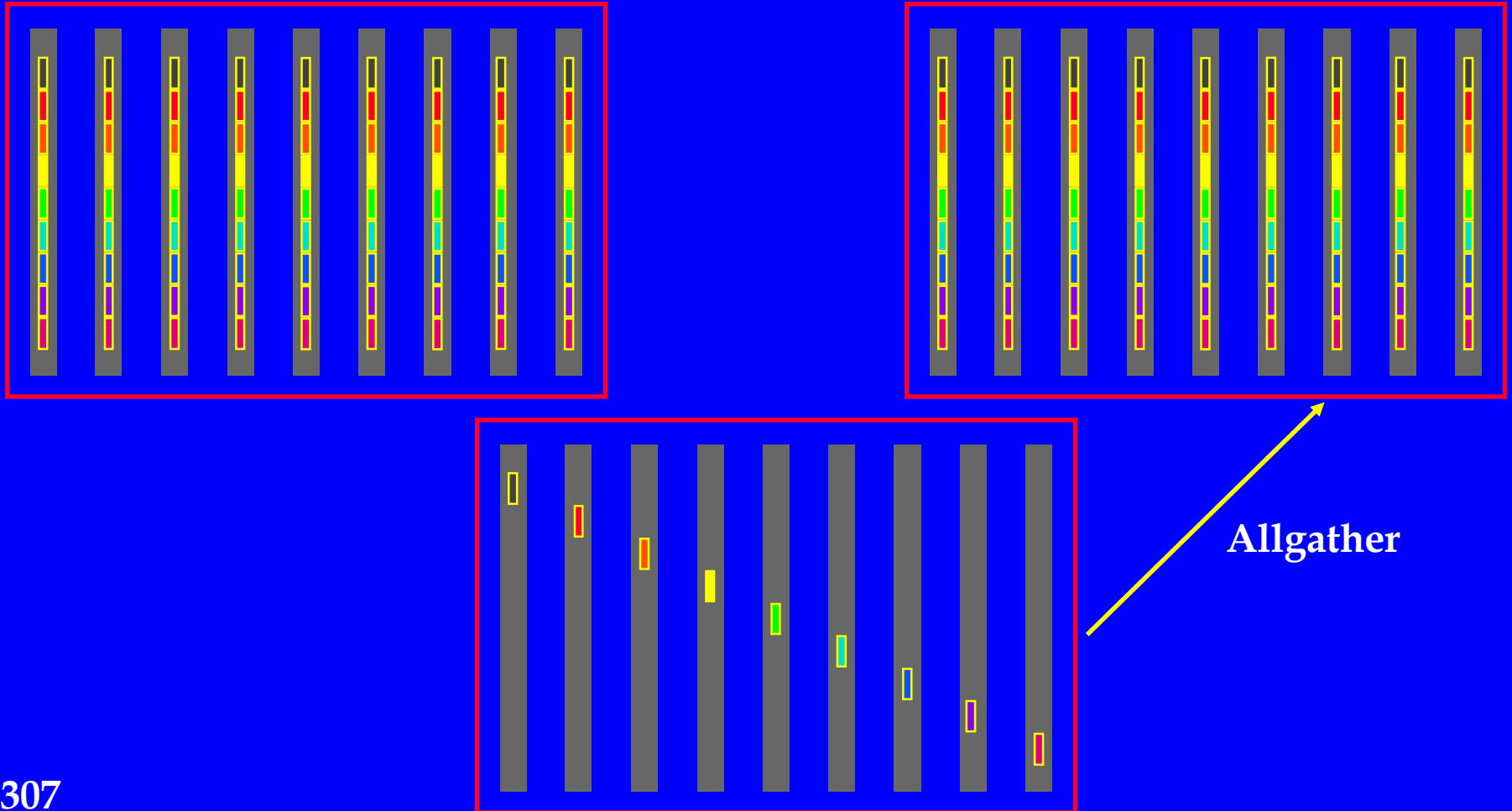
Allreduce (long vector)



Allreduce (long vector)



Allreduce (long vector)



Cost of Reduce-scatter/Allgather Allreduce

- Assumption: power of two number of nodes

Reduce-scatter	$(p-1)\alpha + \frac{p-1}{p}n\beta + \frac{p-1}{p}n\gamma$
Allgather	$(p-1)\alpha + \frac{p-1}{p}n\beta$
	$\frac{2(p-1)\alpha + 2\frac{p-1}{p}n\beta + \frac{p-1}{p}n\gamma}{2}$

Cost of Reduce-scatter/Allgather Allreduce

- Assumption: power of two number of nodes

Reduce-scatter	$(p-1)\alpha + \frac{p-1}{p}n\beta + \frac{p-1}{p}n\gamma$
Allgather	$(p-1)\alpha + \frac{p-1}{p}n\beta$
	<hr/>
	$2(p-1)\alpha + 2\frac{p-1}{p}n\beta + \frac{p-1}{p}n\gamma$

Notice: attains the lower bound for bandwidth and computation

Recap

Reduce-scatter

$$(p-1)\alpha + \frac{p-1}{p}n(\beta + \gamma)$$

Scatter

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Gather

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Allgather

$$(p-1)\alpha + \frac{p-1}{p}n\beta$$

Reduce(-to-one)

Allreduce

Broadcast

Recap

Reduce-scatter

$$(p-1)\alpha + \frac{p-1}{p}n(\beta + \gamma)$$

Scatter

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Gather

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Allgather

$$(p-1)\alpha + \frac{p-1}{p}n\beta$$

Reduce(-to-one)

$$(p-1 + \log(p))\alpha + \frac{p-1}{p}n(2\beta + \gamma)$$

Allreduce

Broadcast

Recap

Reduce-scatter

$$(p-1)\alpha + \frac{p-1}{p}n(\beta + \gamma)$$

Scatter

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Gather

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Allgather

$$(p-1)\alpha + \frac{p-1}{p}n\beta$$

Reduce(-to-one)

$$(p-1 + \log(p))\alpha + \frac{p-1}{p}n(2\beta + \gamma)$$

Allreduce

$$2(p-1)\alpha + \frac{p-1}{p}n(2\beta + \gamma)$$

Broadcast

$$(\log(p) + p-1)\alpha + 2\frac{p-1}{p}n\beta$$

Recap

Reduce-scatter

$$(p-1)\alpha + \frac{p-1}{p}n(\beta + \gamma)$$

Scatter

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Gather

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Allgather

$$(p-1)\alpha + \frac{p-1}{p}n\beta$$

Reduce(-to-one)

$$(p-1 + \log(p))\alpha + \frac{p-1}{p}n(2\beta + \gamma)$$

Allreduce

$$2(p-1)\alpha + \frac{p-1}{p}n(2\beta + \gamma)$$

Broadcast

Recap

Reduce-scatter

$$(p-1)\alpha + \frac{p-1}{p}n(\beta + \gamma)$$

Scatter

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Gather

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Allgather

$$(p-1)\alpha + \frac{p-1}{p}n\beta$$

Reduce(-to-one)

$$(p-1 + \log(p))\alpha + \frac{p-1}{p}n(2\beta + \gamma)$$

Allreduce

$$2(p-1)\alpha + \frac{p-1}{p}n(2\beta + \gamma)$$

Broadcast

$$(\log(p) + p-1)\alpha + 2\frac{p-1}{p}n\beta$$

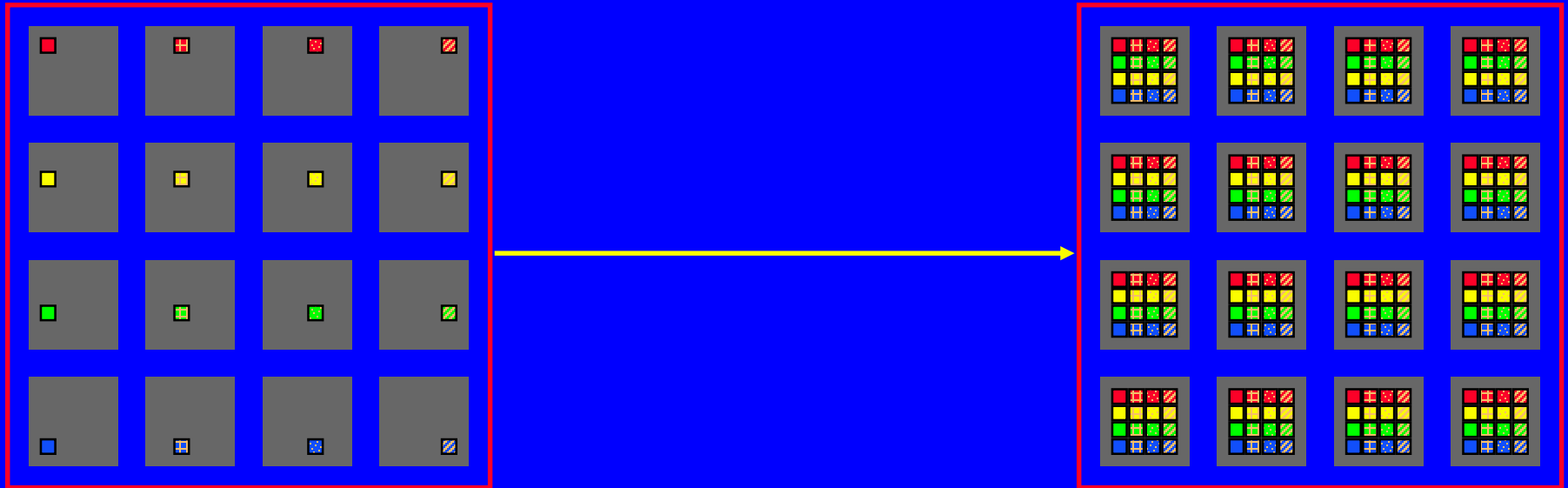
Advanced Techniques:

Taking advantage of higher dimensions

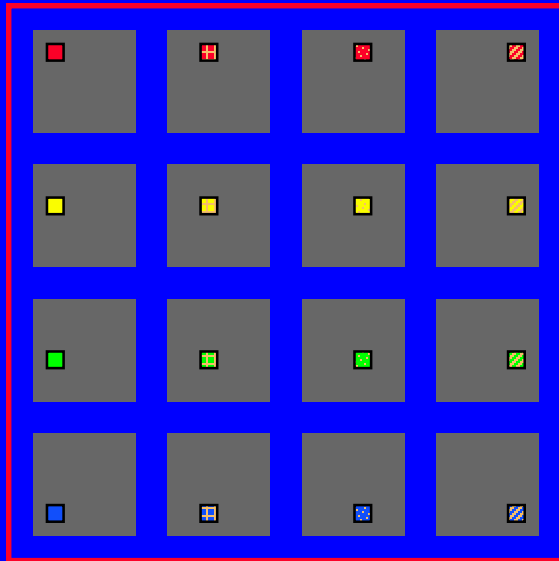
Physical 2D meshes

- **Simple solution: embed logical linear array**
 - problem: large p implies high latency for bucket algorithms
- **Advanced solution: perform operation in each dimension**
 - collect:
collect within rows, followed by collect within columns
 - distributed combine:
same, in reverse

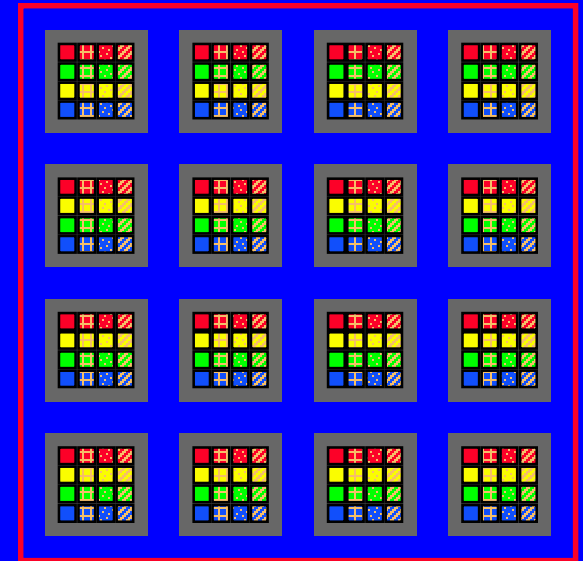
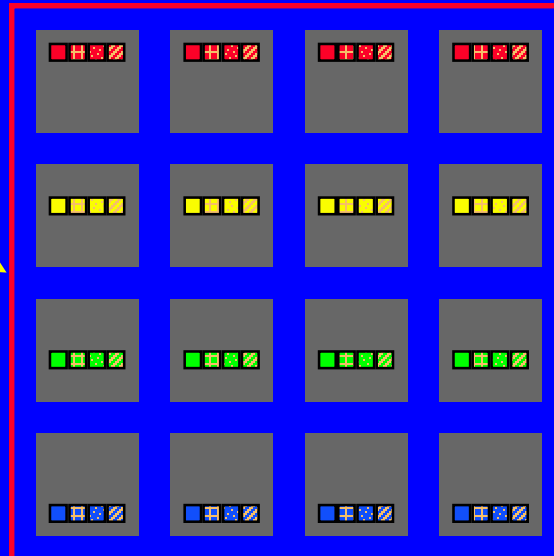
Example: 2D Allgather



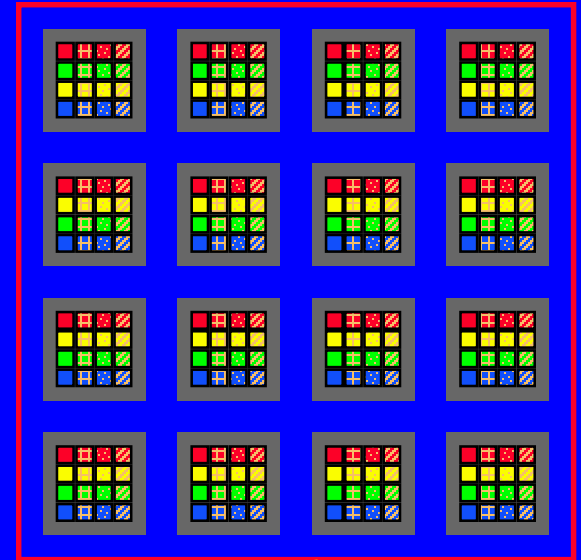
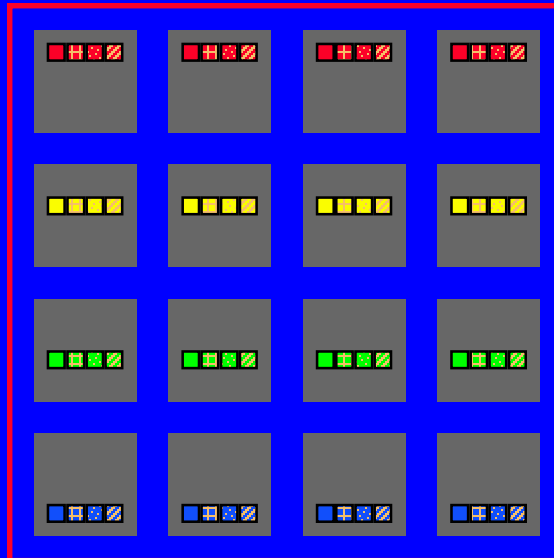
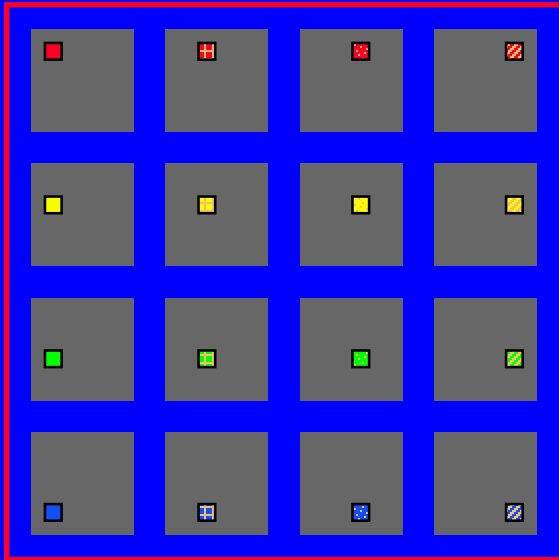
Example: 2D Allgather



Allgather in rows



Example: 2D Collect



Allgather
in columns

Cost of 2D Allgather

row Allgather

$$(c-1)\alpha + (c-1)\frac{n}{p}\beta$$

column Allgather

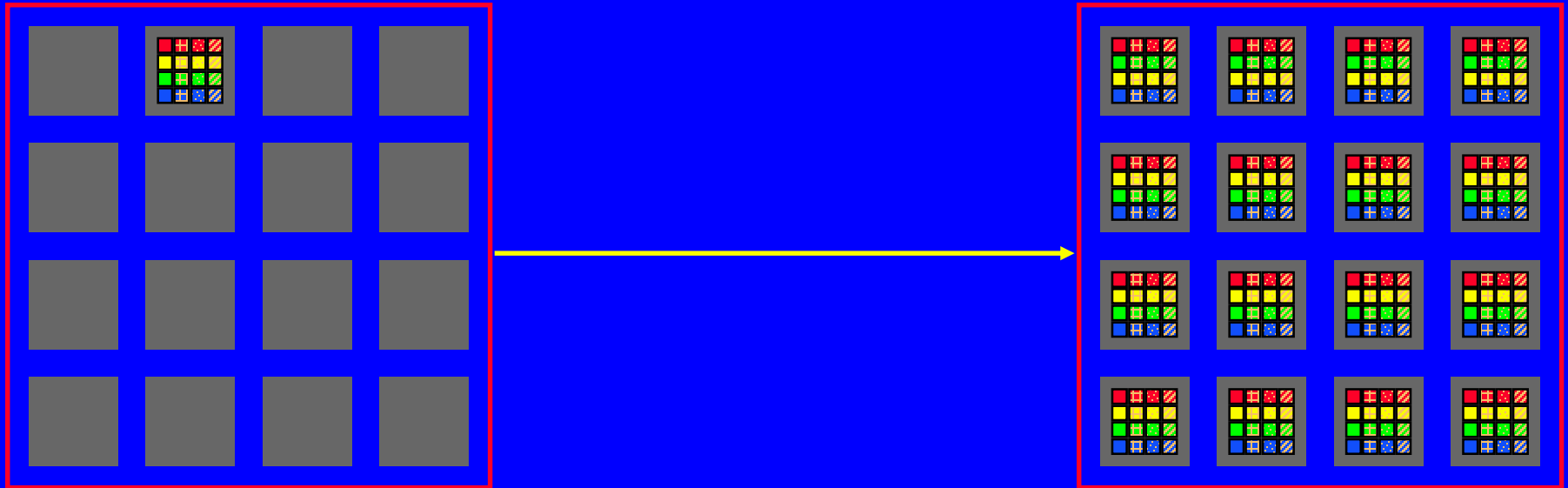
$$(r-1)\alpha + (r-1)\frac{c}{p}n\beta$$

$$(r+c-2)\alpha + \frac{p-1}{p}n\beta$$

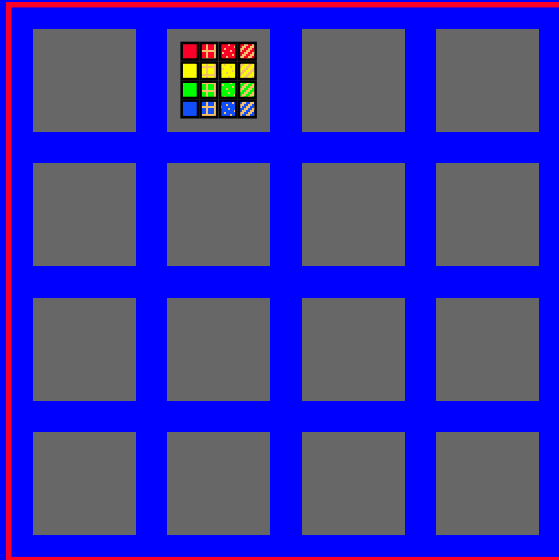
latency term is
reduced

bandwidth term
is unaffected

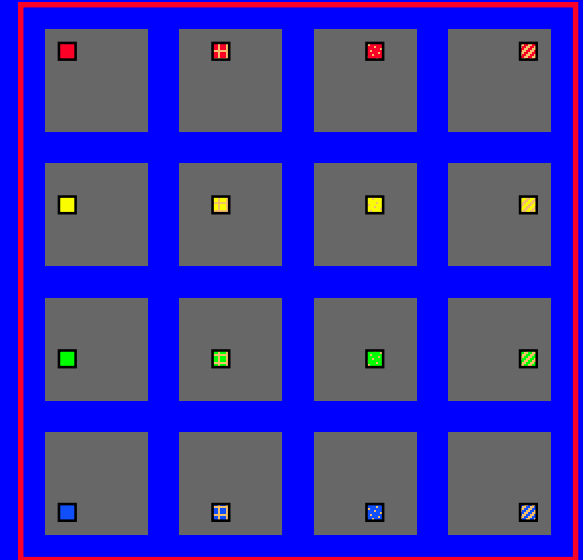
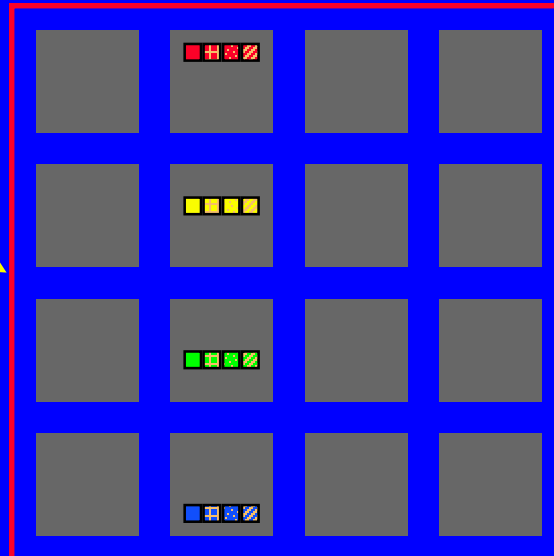
Example: 2D Scatter/Allgather Broadcast



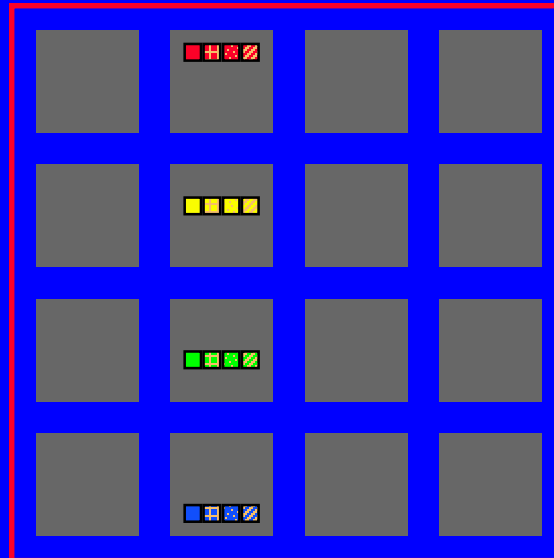
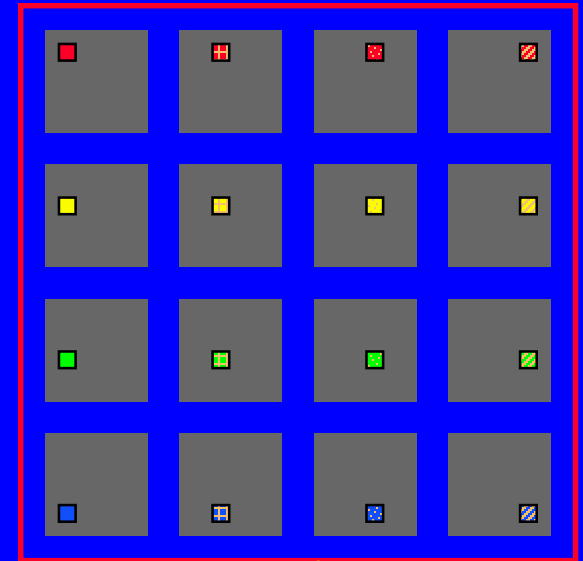
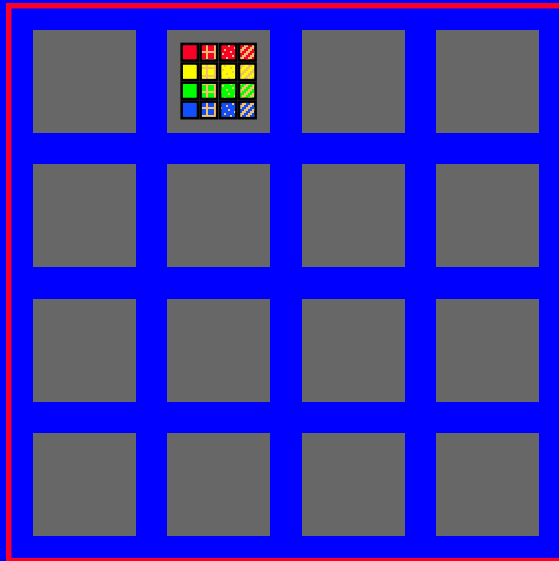
Example: 2D Scatter/Allgather Broadcast



scatter in columns

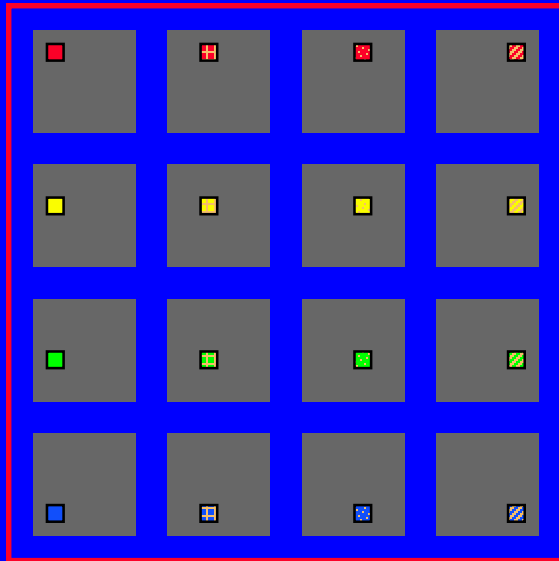


Example: 2D Scatter/ Allgather Broadcast

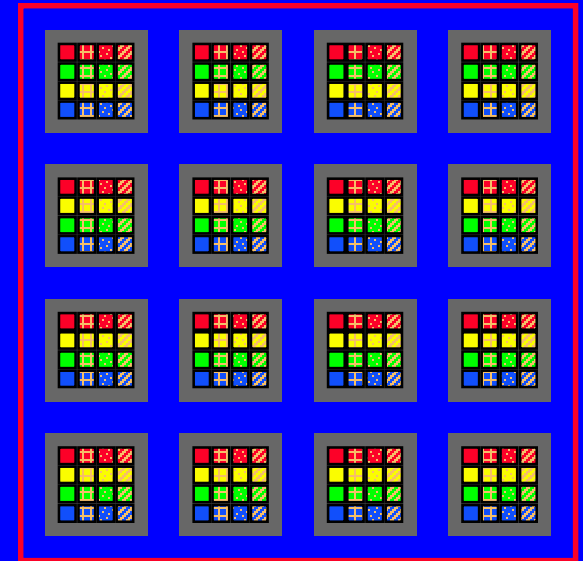
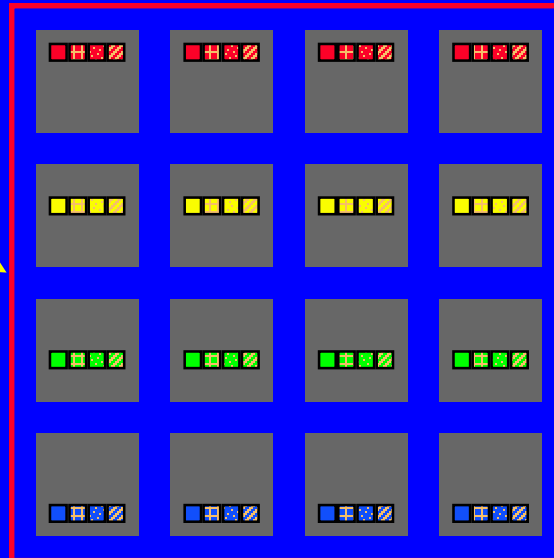


Scatter in rows

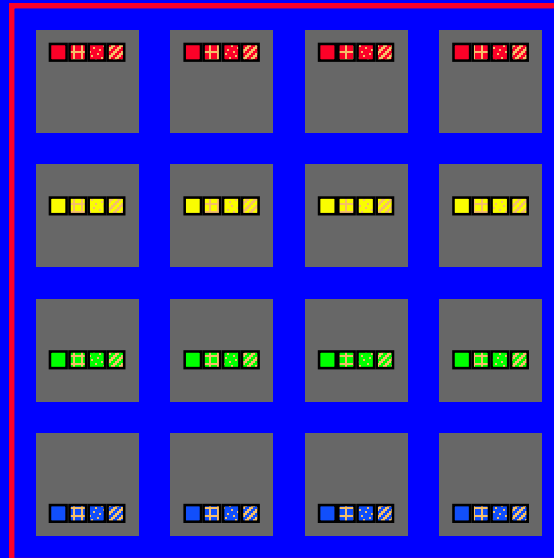
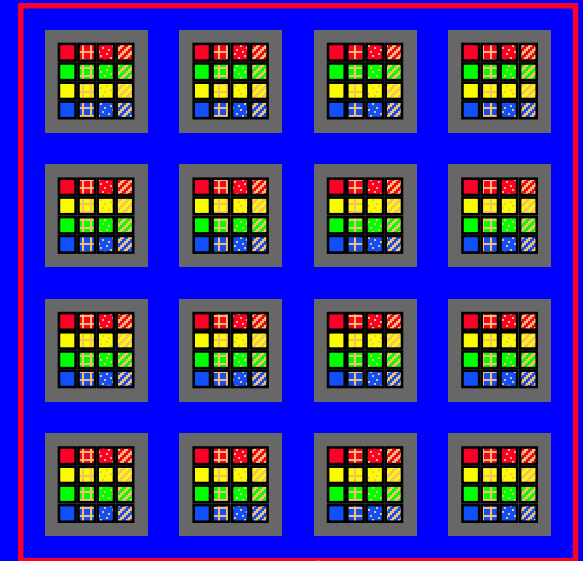
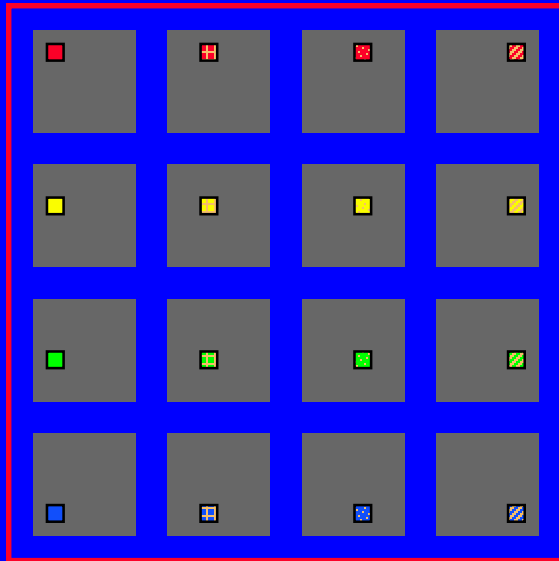
Example: 2D Scatter/ Allgather Broadcast



Allgather in rows



Example: 2D Scatter/Collect Broadcast



Allgather
in columns

Cost of 2D scatter/Allgather broadcast

$$(\log(p) + r + c - 2)\alpha + 2\frac{p-1}{p}n\beta$$

A building block approach to library implementation

- Short vector case
- Long vector case
- Hybrid algorithms

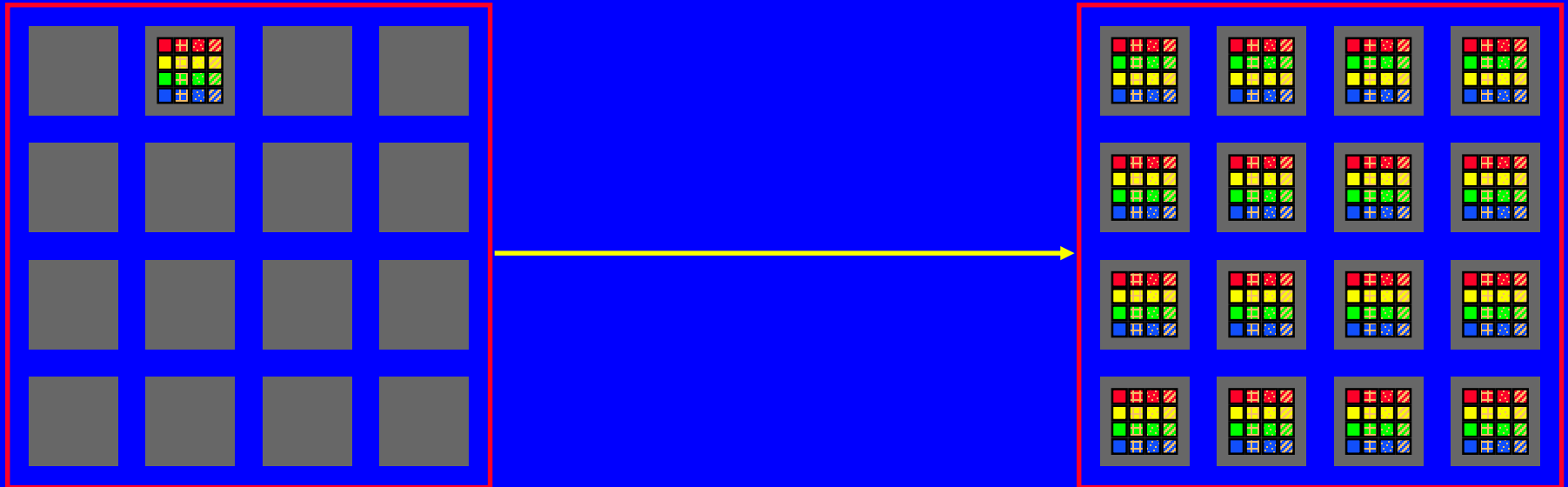
Hybrid algorithms (intermediate length case)

- algorithms must balance latency, cost due to vector length, and network conflicts

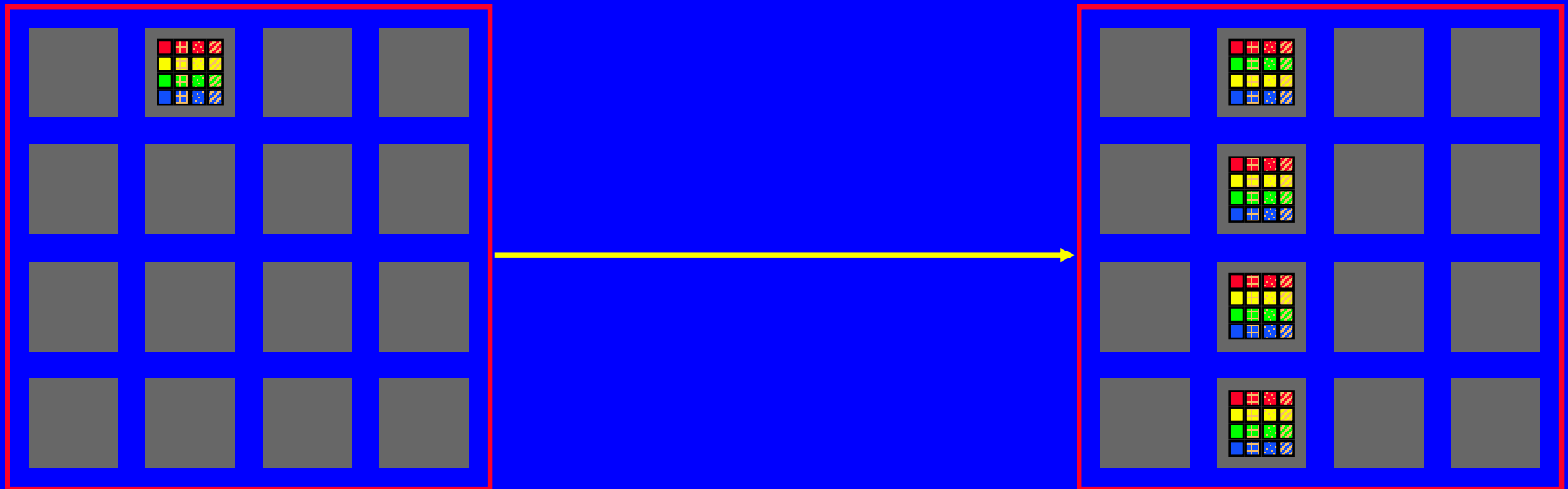
Example

- We will illustrate the techniques using the broadcast as an example
 - short vector: minimum spanning tree broadcast

Example: 2D Broadcast

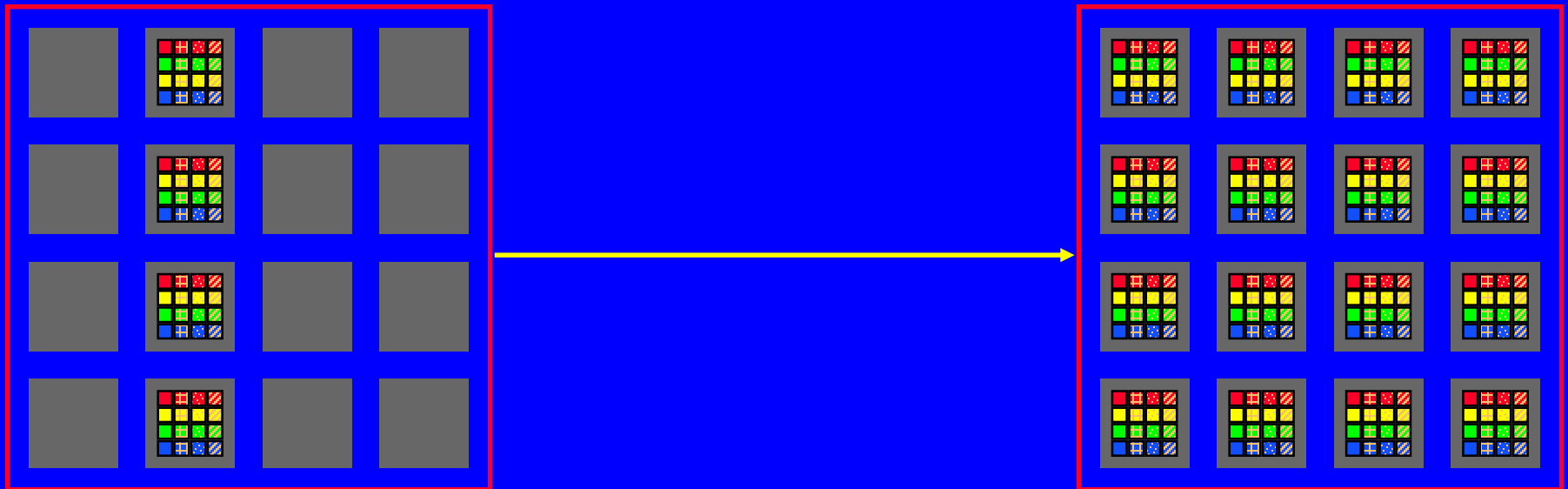


Example: 2D Broadcast



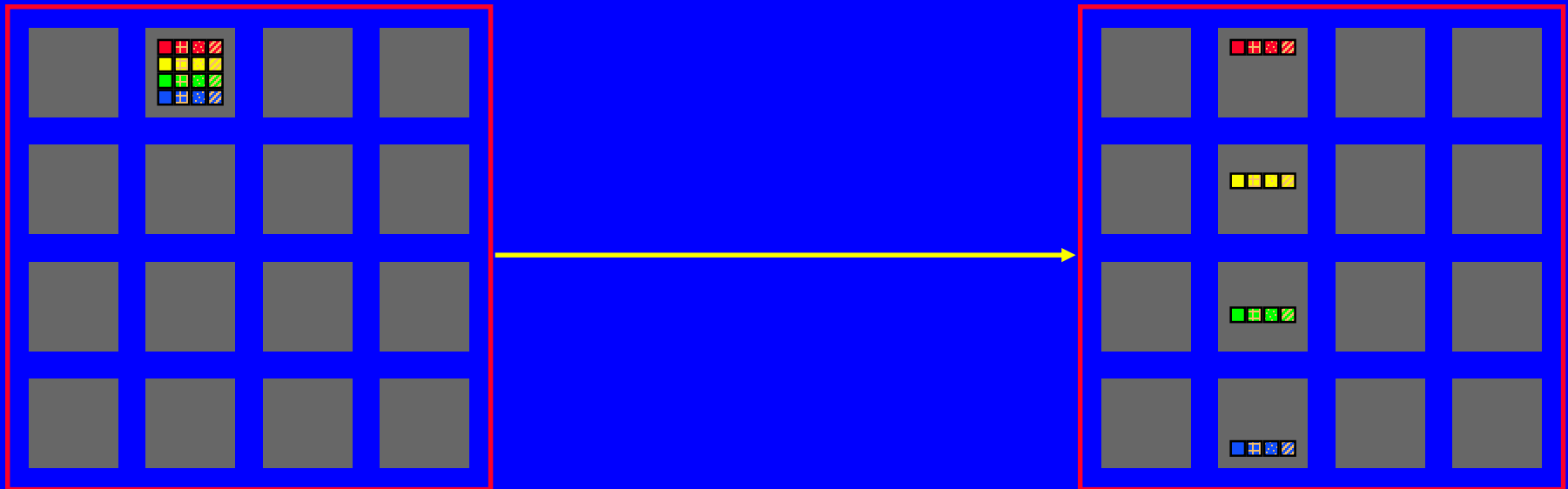
- Option 1:
 - **MST broadcast in column**
 - MST broadcast in rows

Example: 2D Broadcast



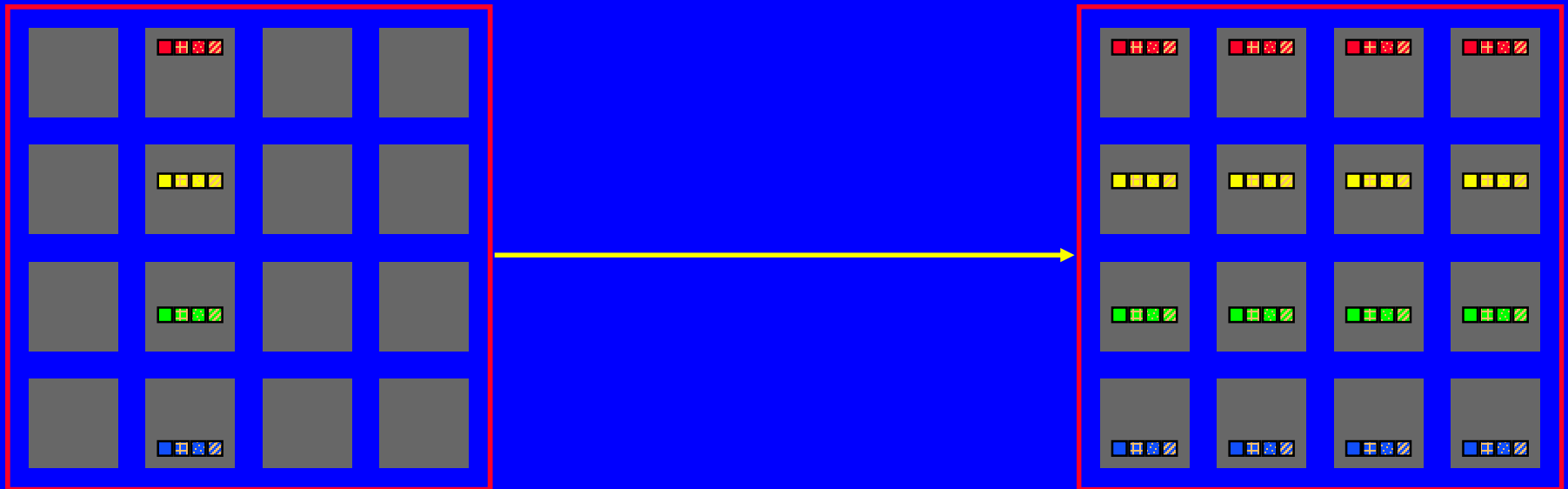
- Option 1:
 - MST broadcast in column
 - **MST broadcast in rows**

Example: 2D Broadcast



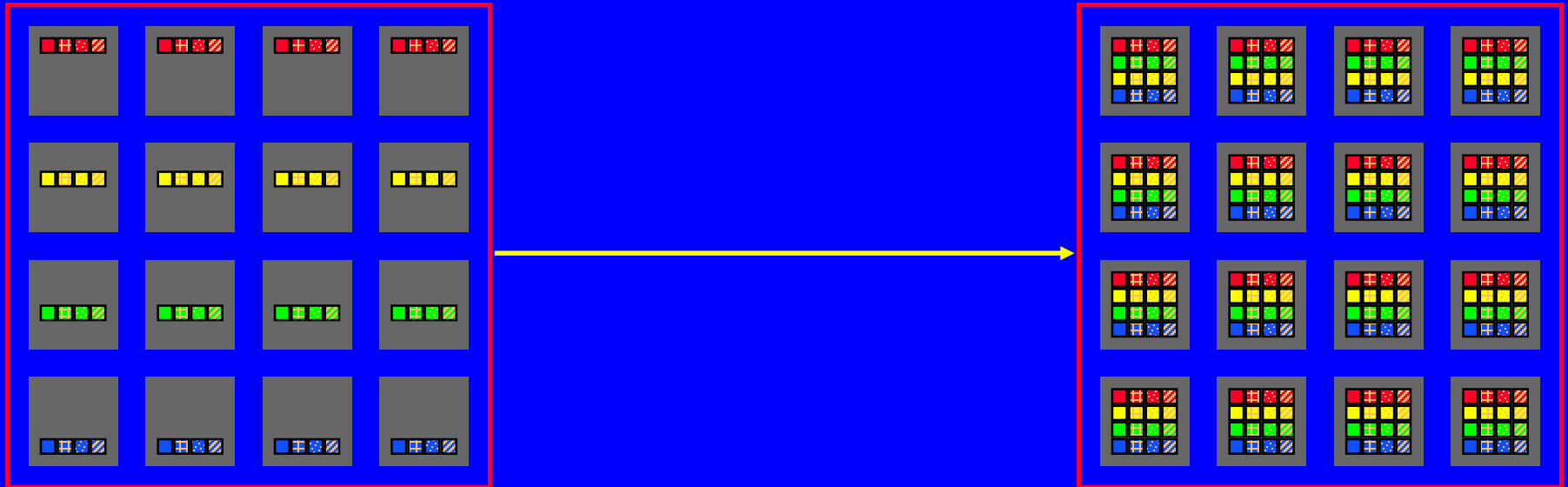
- Option 2:
 - **Scatter in column**
 - MST broadcast in rows
 - Allgather in columns

Example: 2D Broadcast



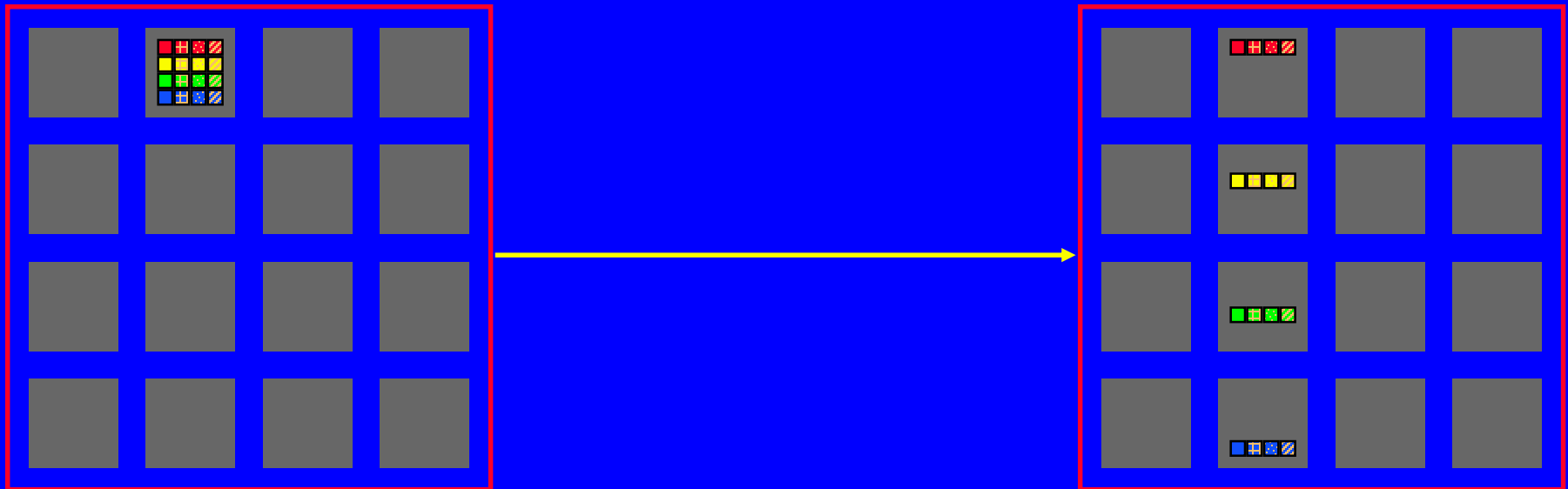
- Option 2:
 - Scatter in column
 - **MST broadcast in rows**
 - Allgather in columns

Example: 2D Broadcast



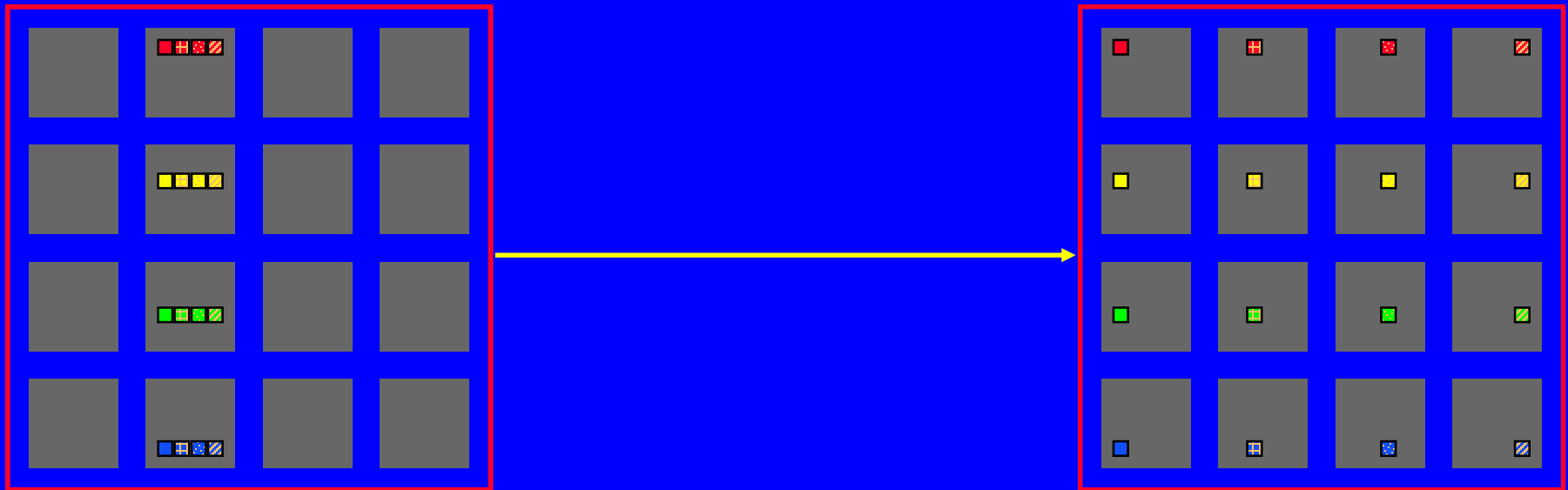
- Option 2:
 - Scatter in column
 - MST broadcast in rows
 - Allgather **in columns**

Example: 2D Broadcast



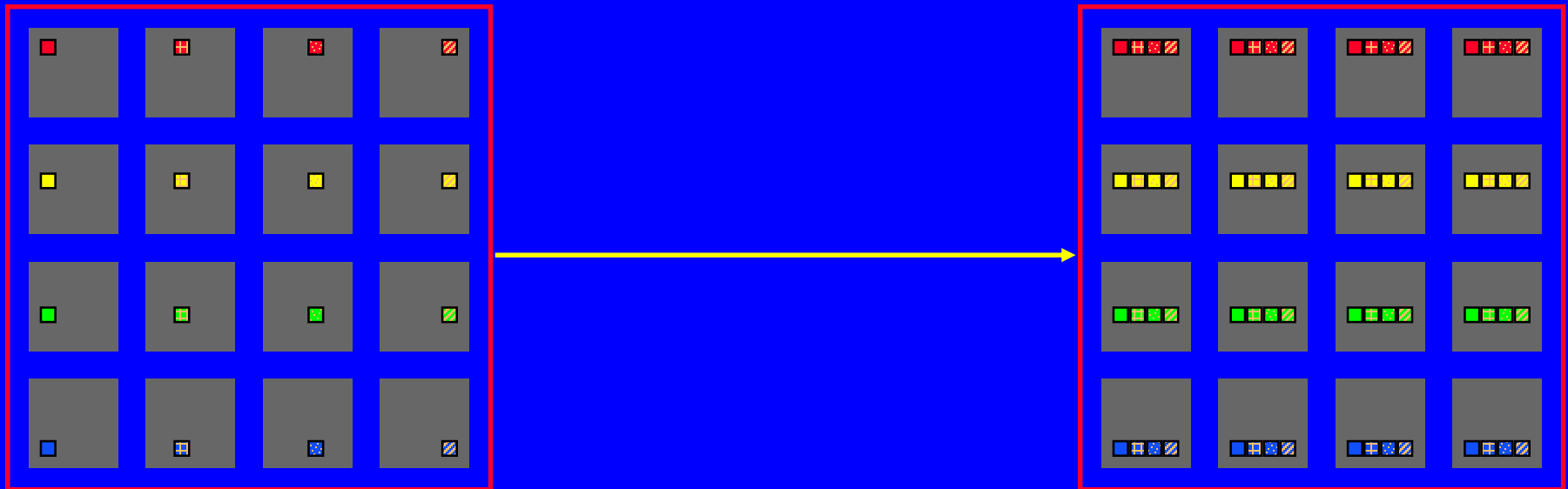
- Option 3:
 - Scatter in column
 - Scatter in rows
 - Allgather in rows
 - Allgather in columns

Example: 2D Broadcast



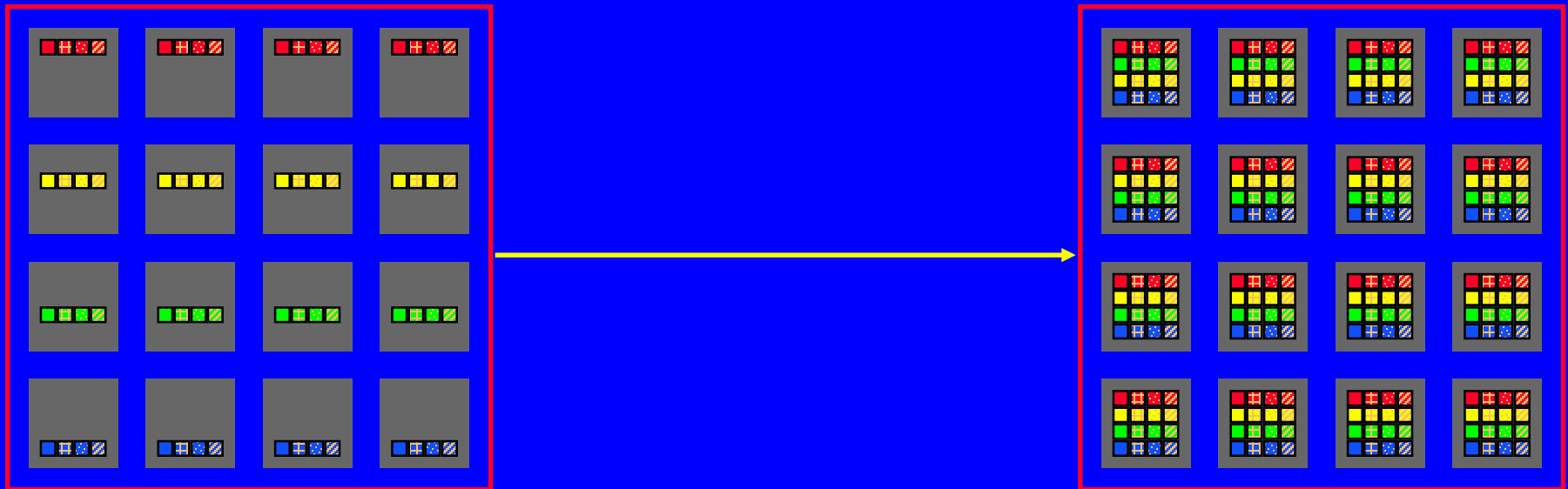
- Option 3:
 - Scatter in column
 - **Scatter in rows**
 - Allgather in rows
 - Allgather in columns

Example: 2D Broadcast



- Option 3:
 - Scatter in column
 - Scatter in rows
 - Allgather **in rows**
 - Allgather in columns

Example: 2D Broadcast



- Option 3:
 - Scatter in column
 - Scatter in rows
 - Allgather in rows
 - Allgather **in columns**

Cost comparison

- **Option 1:**
 - MST broadcast in column
 - MST broadcast in rows
- **Option 2:**
 - Scatter in column
 - MST broadcast in rows
 - Allgather in columns
- **Option 3:**
 - Scatter in column
 - Scatter in rows
 - Allgather in rows
 - Allgather in columns

$$\frac{\log(c)\alpha + \log(c)n\beta}{\log(r)\alpha + \log(r)n\beta}$$
$$\log(p)\alpha + \log(p)n\beta$$

Cost comparison

- Option 1:
 - MST broadcast in column
 - MST broadcast in rows
- Option 2:
 - Scatter in column
 - MST broadcast in rows
 - Allgather in columns
- Option 3:
 - Scatter in column
 - Scatter in rows
 - Allgather in rows
 - Allgather in columns

$$\begin{array}{c}
 \log(c)\alpha + \frac{c-1}{c}n\beta \\
 \log(r)\alpha + \log(r)\frac{n}{c}\beta \\
 (c-1)\alpha + \frac{c-1}{c}n\beta \\
 \hline
 (\log(p) + c - 1)\alpha + \left(2\frac{c-1 + \log(r)}{c}\right)n\beta
 \end{array}$$

Cost comparison

- Option 1:
 - MST broadcast in column
 - MST broadcast in rows
- Option 2:
 - Scatter in column
 - MST broadcast in rows
 - Allgather in columns
- Option 3:
 - Scatter in column
 - Scatter in rows
 - Allgather in rows
 - Allgather in columns

$$\begin{array}{l}
 \log(c)\alpha + \frac{c-1}{c}n\beta \\
 \log(r)\alpha + \frac{r-1}{r}\frac{n}{c}\beta \\
 (r-1)\alpha + \frac{r-1}{r}\frac{n}{c}\beta \\
 (c-1)\alpha + \frac{c-1}{c}n\beta \\
 \hline
 (\log(p) + r + c - 2)\alpha + 2\frac{p-1}{p}n\beta
 \end{array}$$

Cost comparison

- Option 1:

- MST broadcast in column
- MST broadcast in rows

$$\log(p)\alpha + \log(p)n\beta$$

- Option 2:

- Scatter in column
- MST broadcast in rows
- Allgather in columns

$$(\log(p) + c - 1)\alpha + \left(2\frac{c - 1 + \log(r)}{c}\right)n\beta$$

- Option 3:

- Scatter in column
- Scatter in rows
- Allgather in rows
- Allgather in columns

$$(\log(p) + r + c - 2)\alpha + 2\frac{p - 1}{p}n\beta$$

Higher dimensions

- This technique can be extended by viewing one- and two-dimensional meshes logically as higher dimensions
 - reduces latency
 - incurs network conflicts
 - can be used to create faster short vector implementations
- Details require more time that is available today

Other techniques

- Pipelined algorithms
 - can be used to further reduce the cost of broadcast and combine-to-one for long vectors
 - very effective on hypercubes
 - » (Ho and Johnsson)
 - effective on meshes with low latency
 - » (Watts and van de Geijn)
 - complicated to implement, analyze and explain

Outline

Part I: Theory

- Model of computation
- Collective communications
- A building block approach to library implementation

Part II: Practice

- Implementation on the Paragon
- Performance results
- Applications

Outline

Part I: Theory

- Model of computation
- Collective communications
- A building block approach to library implementation

Part II: Practice

- **Implementation on the Paragon**
- Performance results
- Applications

Theory is nice, but how does it work in practice?

- Paragon does not match our model
 - Bad news:
 - » sending and receiving more complex than the model indicates
 - » forced messages vs. unforced messages
 - » preposted messages vs. nonpreposted messages
 - » etc.
 - Good news:
 - » excess bandwidth in the network

Interprocessor Collective Communication (InterCom) Project

Implementation on the Paragon

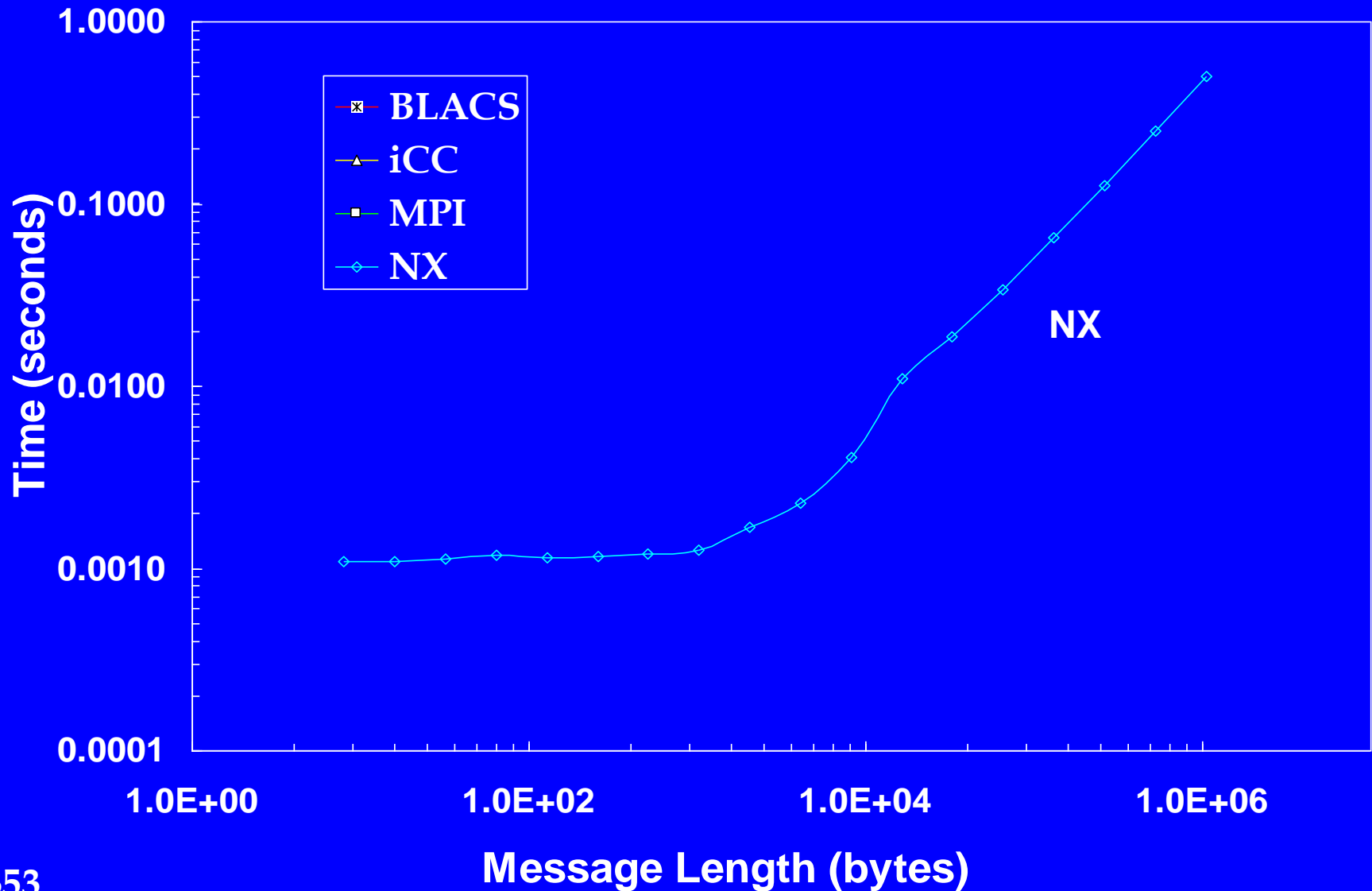
- Short vector building blocks
 - reduce latency by *not* preposting and synchronizing
- Long vector building blocks
 - improve bandwidth by preposting and synchronizing
- Incorporate more complex issues into model
 - various startups, bandwidths, depending on situation
- Use simple heuristic to choose hybrid strategy
 - because of excess bandwidth, the mesh acts more like a hypercube, for which some solid theory exists
 - » (van de Geijn)
 - details go beyond this tutorial.

Performance

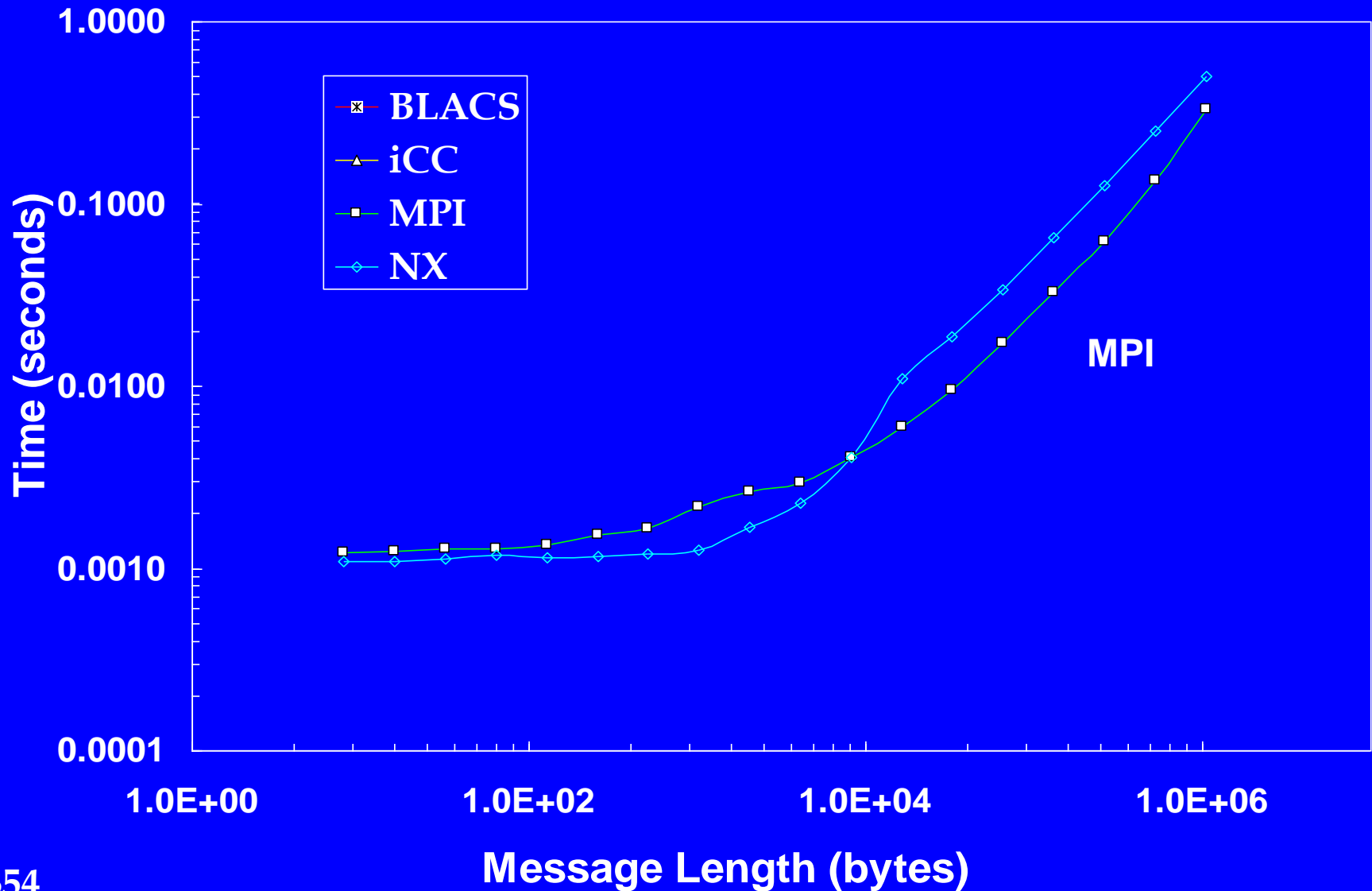
Performance comparison

- **NX collective communication**
- **Message Passing Interface (MPI)**
 - Reference implementation from ANL and MSU
 - Bill Gropp, Rusty Lusk, and Tony Skjellum
- **Basic Linear Algebra Communication Subprograms (BLACS)**
 - Communication library of ScaLAPACK
 - Reference implementation from the Univ. of TN
 - Jack Dongarra and Clint Whaley
- **Interprocessor Collective Communication (iCC) Library**
 - High performance implementation by the InterCom team

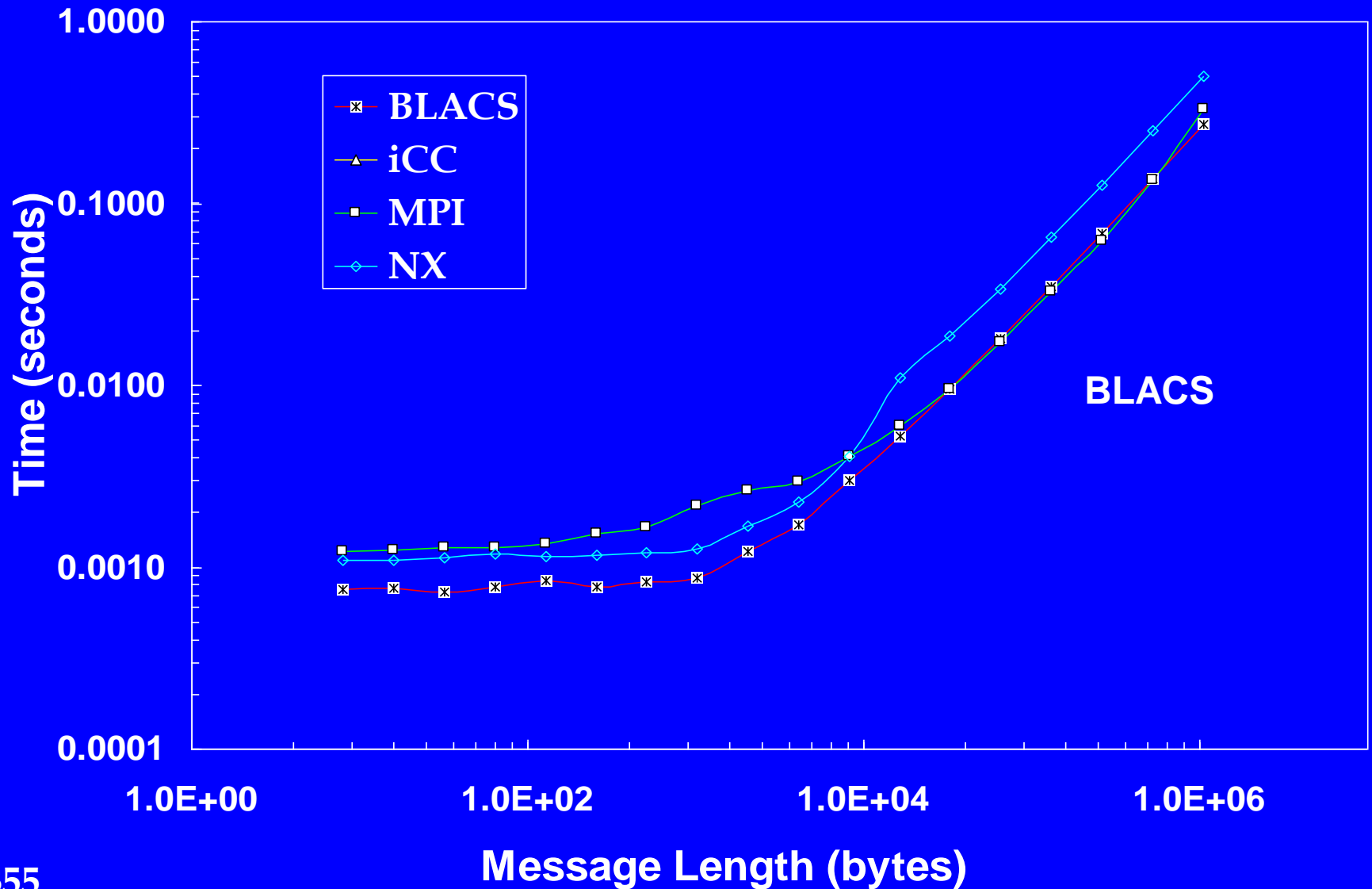
Broadcast on 16 x 32 mesh Paragon



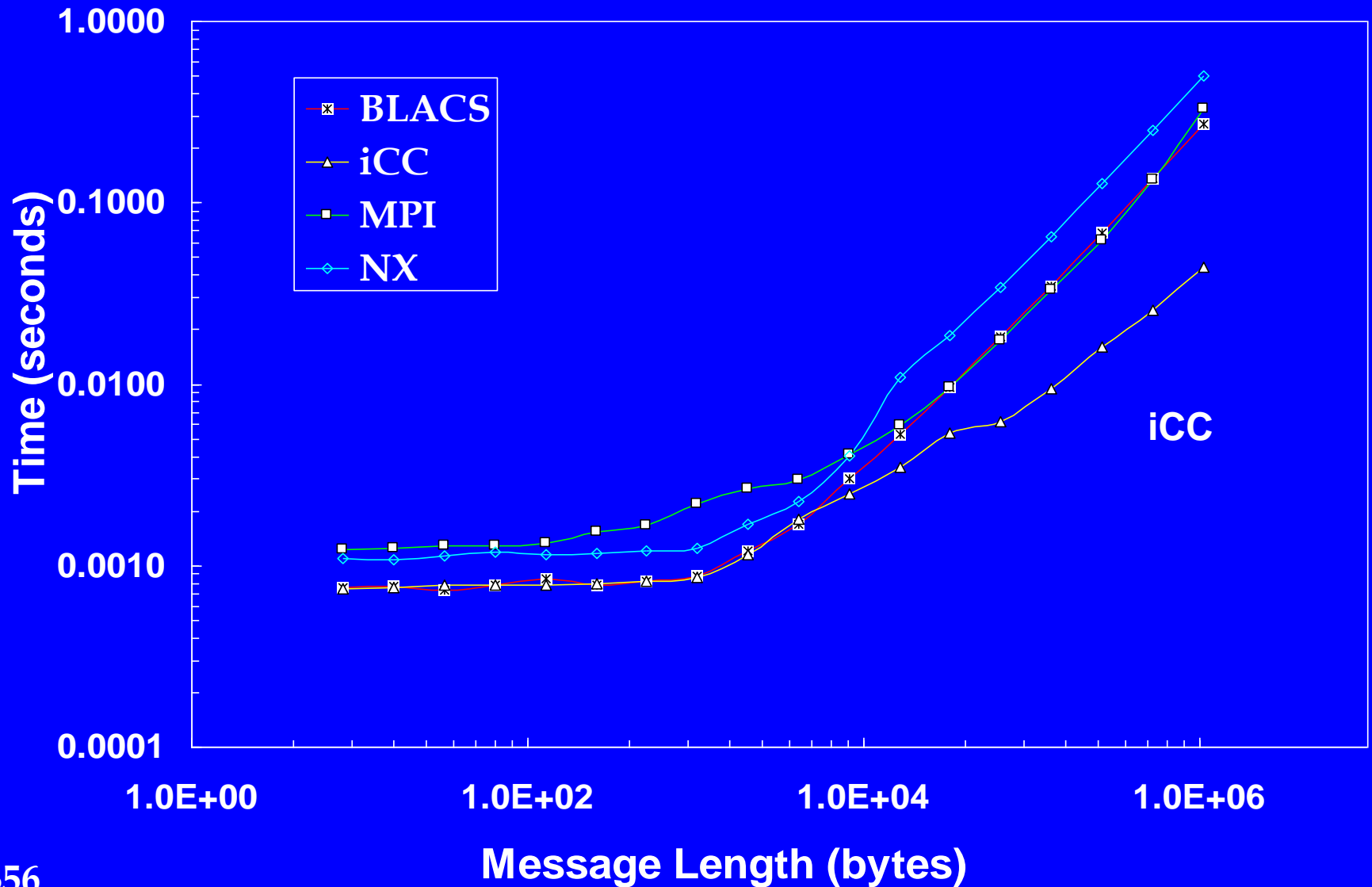
Broadcast on 16 x 32 mesh Paragon



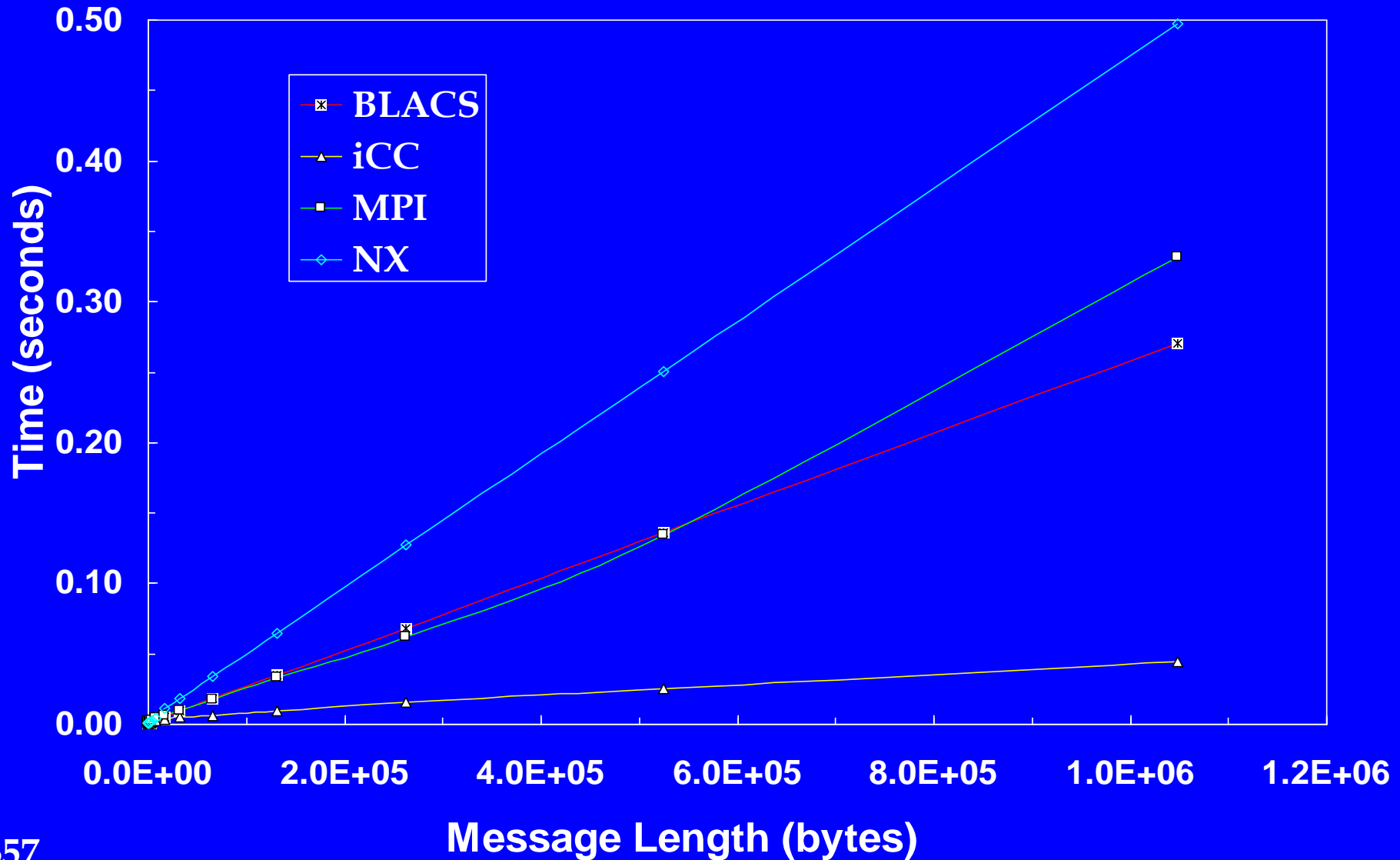
Broadcast on 16 x 32 mesh Paragon



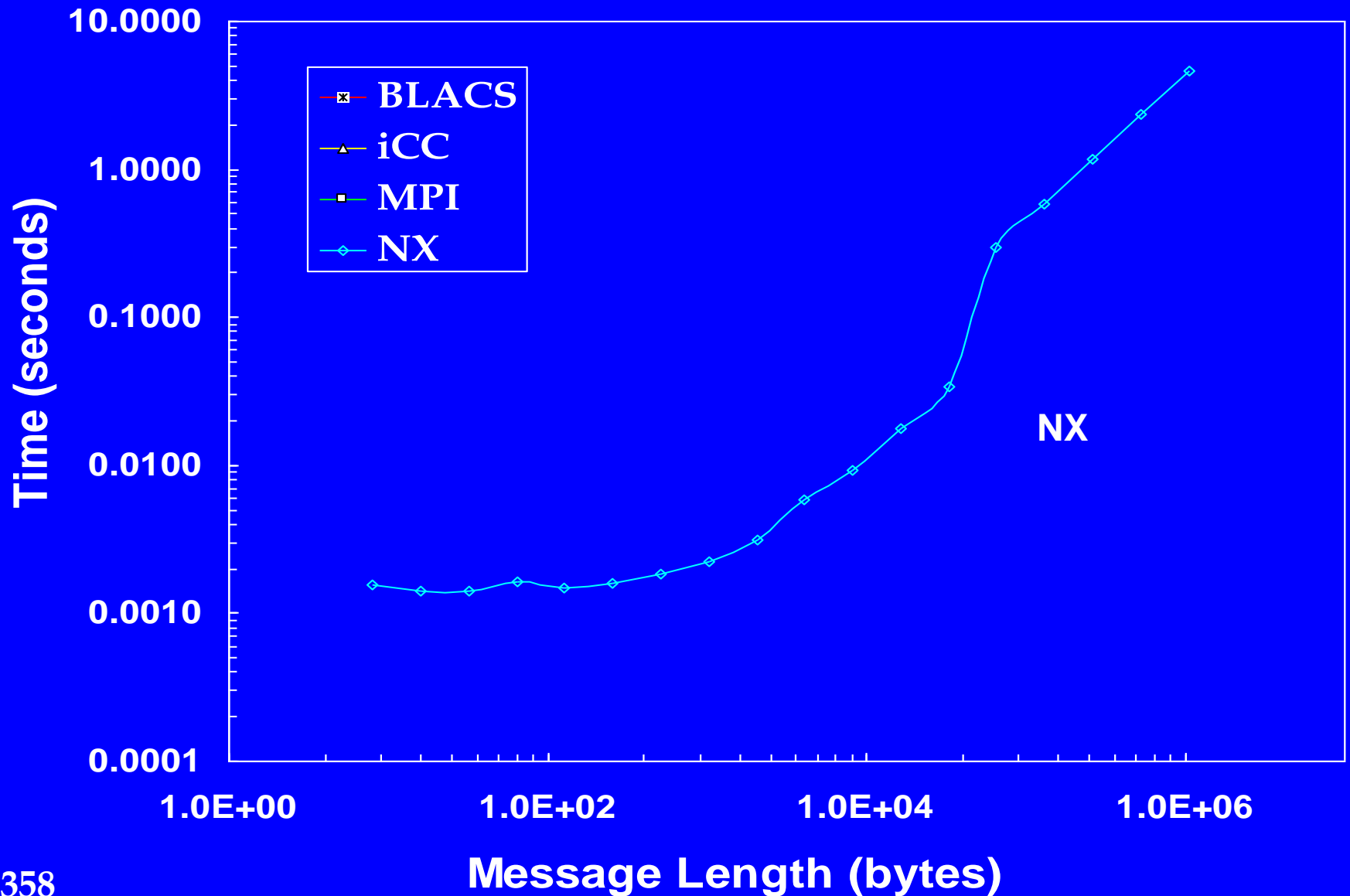
Broadcast on 16 x 32 mesh Paragon



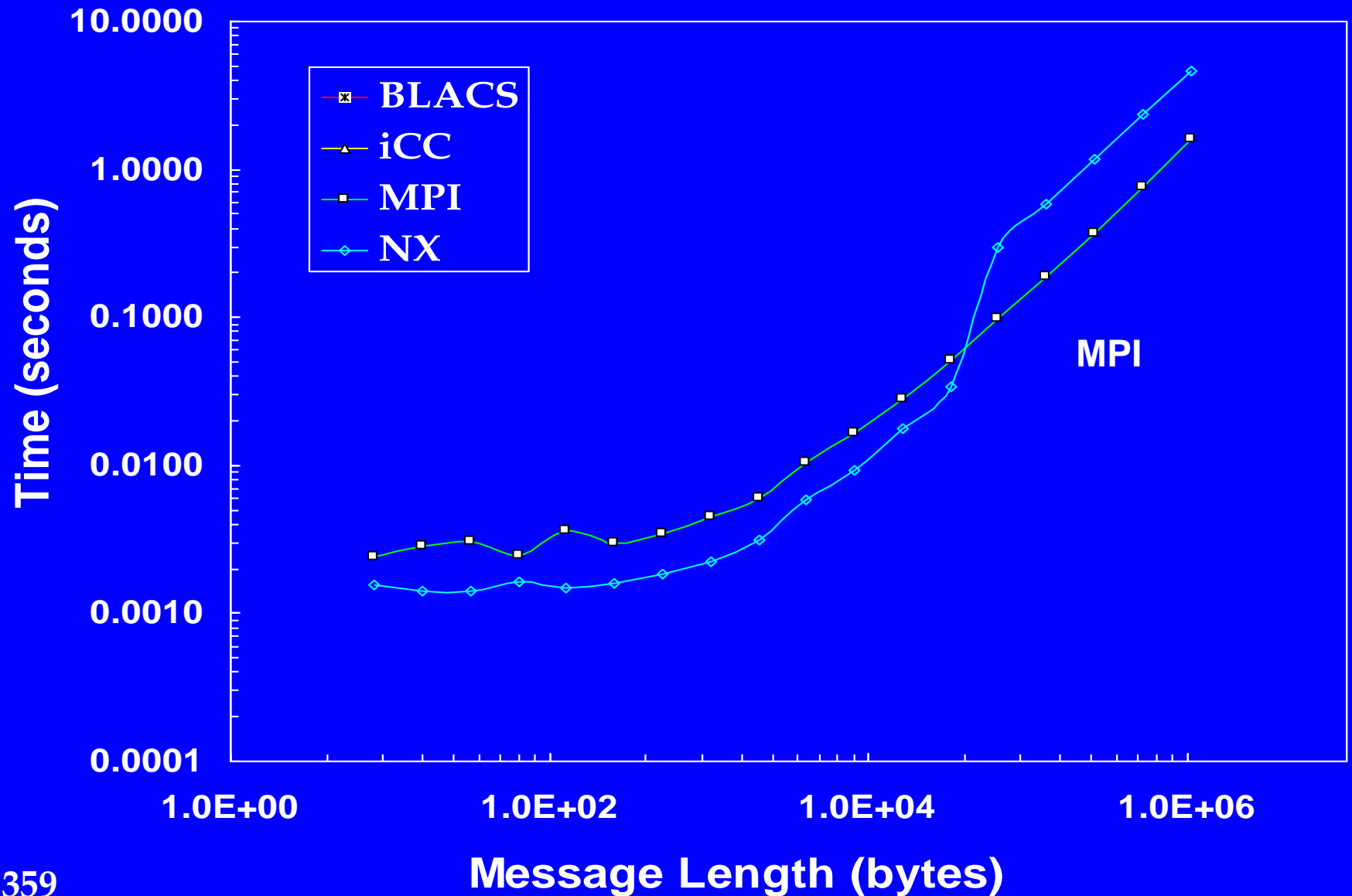
Broadcast on 16 x 32 mesh Paragon



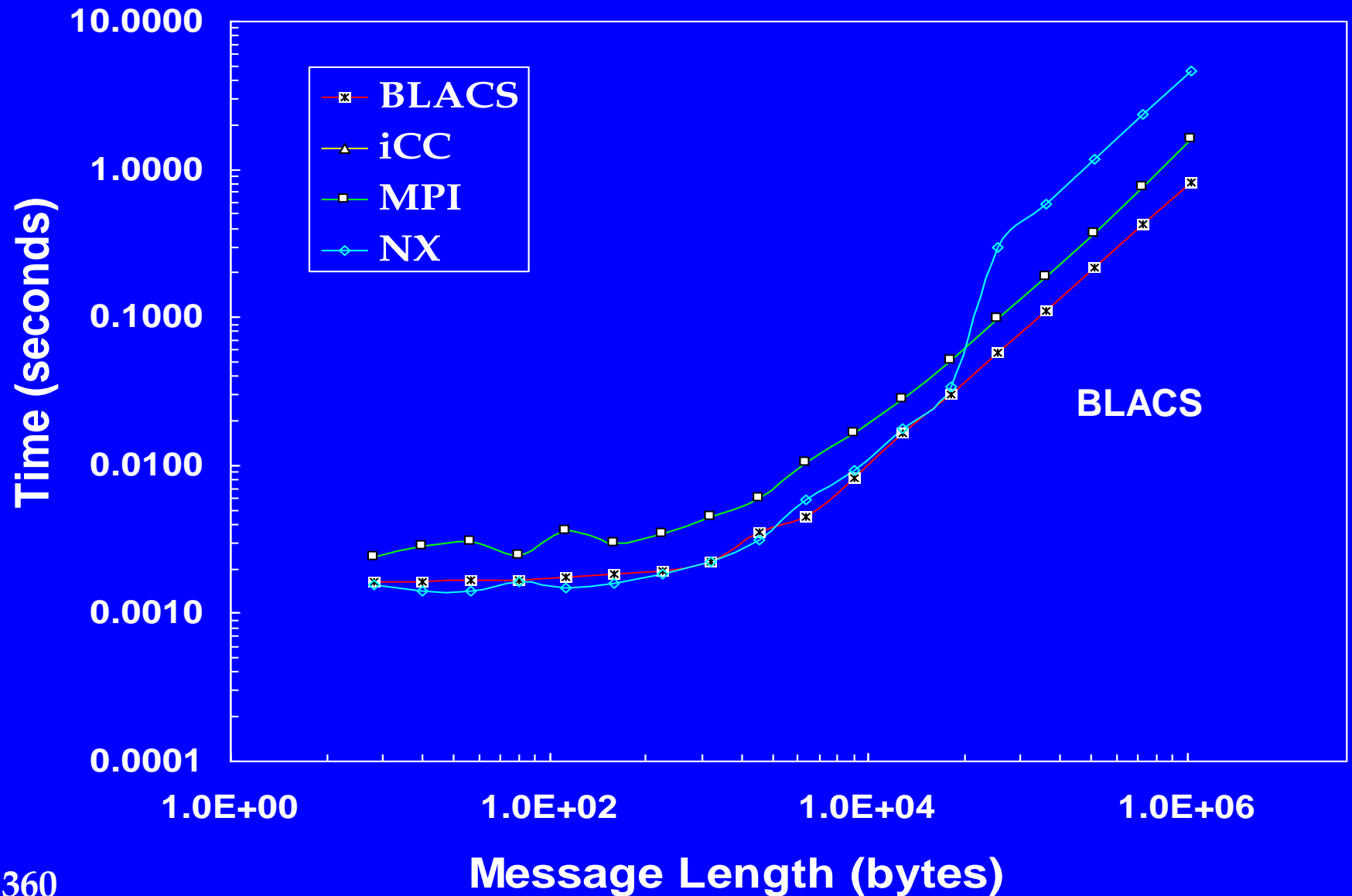
Allgather on 16 x 32 mesh Paragon



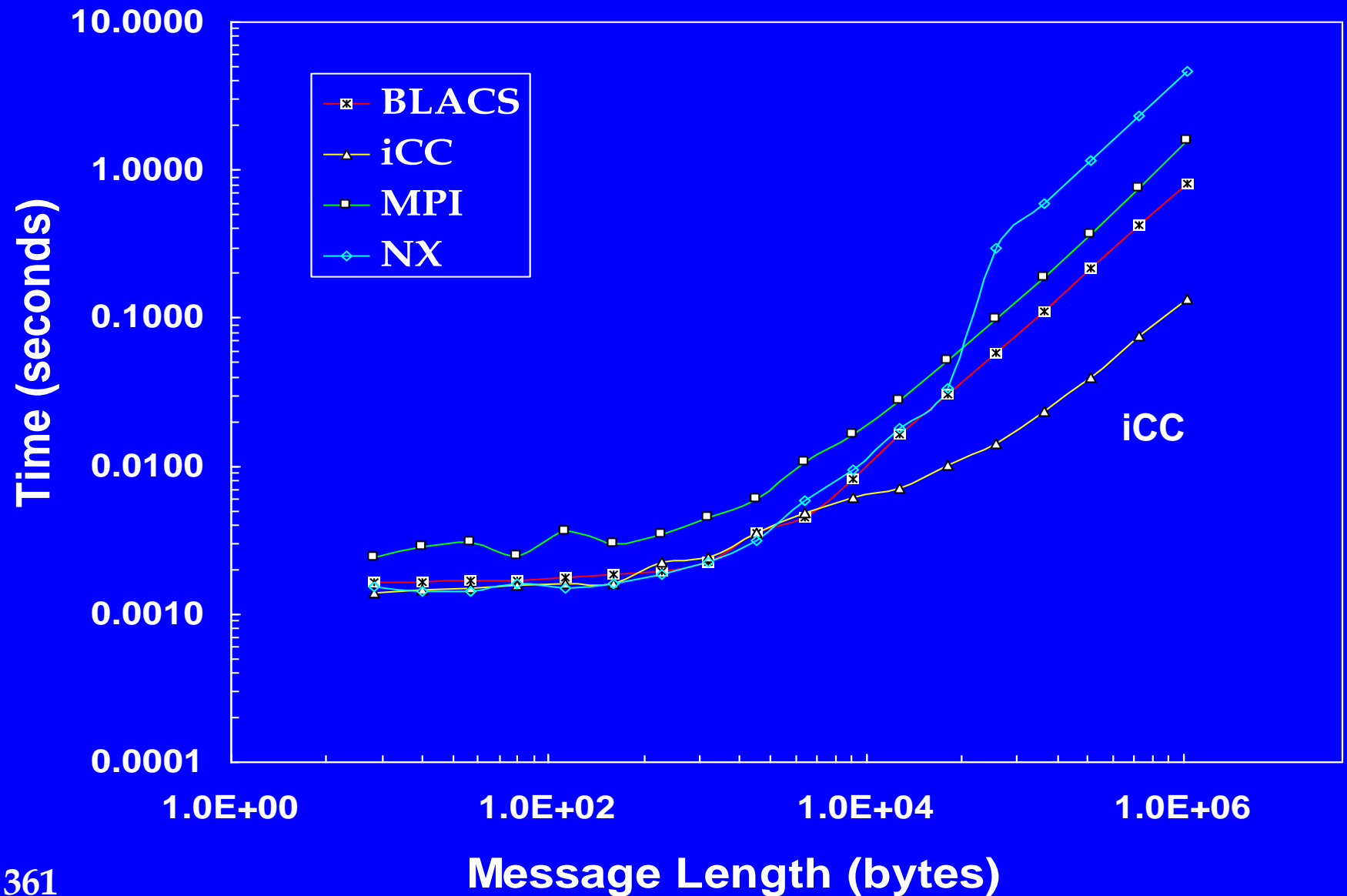
Allgather on 16 x 32 mesh Paragon



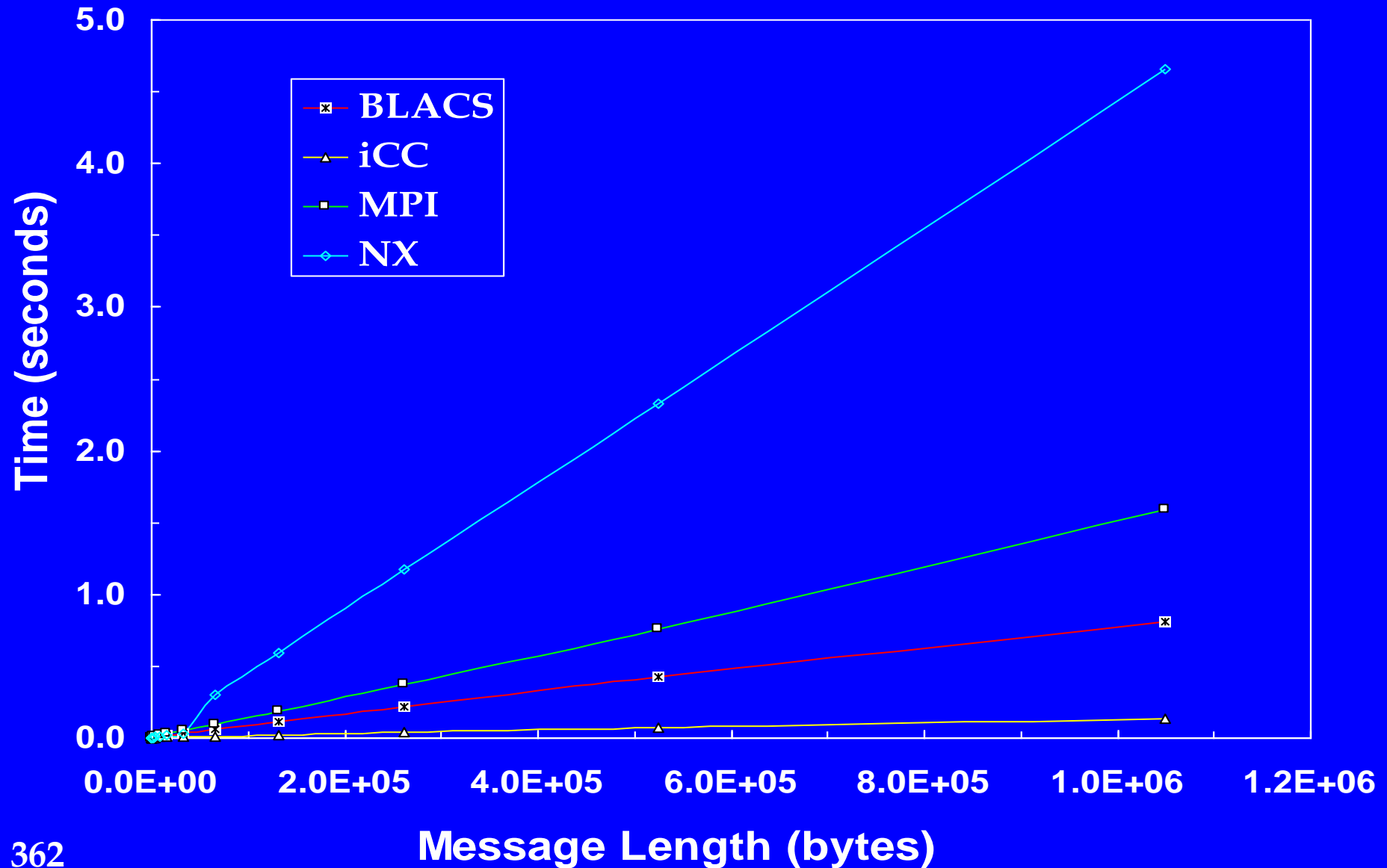
Allgather on 16 x 32 mesh Paragon



Allgather on 16 x 32 mesh Paragon



Allgather on 16 x 32 mesh Paragon



This PowerPoint presentation may be copied for nonprofit educational purposes. Credit should be given to the InterCom project.

For information, contact

rvdg@cs.utexas.edu

CollMark: Collective Communication Benchmark

A look at the current state-of-the-art
(spring 2000)

How to measure the quality of an implementation

- Architecture independent measure of the quality of the implementation:

$$\frac{T_{comm}(n, p)}{T_{p2p}(n)}$$

- Ideally:

$$\frac{T_{comm}(n, p)}{T_{p2p}(n)} \xrightarrow{n \rightarrow \infty} 1 \quad \text{or} \quad 2$$

