Beamer **n**amed **ove**rlay **s**pecifications

- This rich presentation is made with Beamer
- Visual effects will appear only on supporting viewers (like Acrobat Reader)
- 3 parts:
  1. an example with great photos of animals.
  2. How to declare named overlay specifications with `\Beanoves`,
  3. How to use `?(...)` queries in overlay specifications.
- Some basic knowledge of standard beamer overlay specifications is required.

Beamer uses overlay specification aware commands to associate material to slides. The command

```
\only<2-4>{TEXT}
```

will display “TEXT” on slides 2, 3, and 4 only.

Using explicit slide numbers is sometimes difficult, incremental specifications like

```
\item<+>{TEXT}
\item<+(-1)-+>{TEXT}
```

may help in linear situations, but this does not fit to next simple example, to which suit **named overlay specifications**.

Beanoves example about animals: Simple items

Slide 1

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

Beanoves example about animals: Simple items

Slide 2

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

Let us add some dynamism and uncover items one by one

Beanoves example: Uncovered items

Slide 1

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

Beanoves example: Uncovered items

Slide 2

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

Beanoves example: Uncovered items

Slide 3

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

Beanoves example: Uncovered items

Slide 4

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

Beanoves example: Uncovered items

Slide 5

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

We jumped directly to the last slide...

Beanoves example: Uncovered items

Slide 6

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

```
\begin{itemize}
  \item Air
  \begin{itemize}
    \item
    \only<-+>{\transparent{0.3}}
    Chameleo
    \item
    \only<-+>{\transparent{0.3}}
    Gannet
    \end{itemize}
  \item
  \only<-+>{\transparent{0.3}}
  Water
  ...
\end{itemize}
```

... and the source code reads:

Beanoves example: Uncovered items

Slide 7

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

```
\begin{itemize}
  \item Air
  \begin{itemize}
    \item
    \only<-+>{\transparent{0.3}}
    Chameleo
    \item
    \only<-+>{\transparent{0.3}}
    Gannet
    \end{itemize}
  \item
  \only<-+>{\transparent{0.3}}
  Water
```

Notice the very handy incremental overlay specifications in `\only<-+>{...}`.

Beanoves example: Uncovered items

Slide 8

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

```
\begin{itemize}
  \item Air
  \begin{itemize}
    \item
    \only<-+>{\transparent{0.3}}
    Chameleo
    \item
    \only<-+>{\transparent{0.3}}
    Gannet
    \end{itemize}
  \item
  \only<-+>{\transparent{0.3}}
  Water
```

Notice the very handy incremental overlay specifications in `\only<-+>{...}`.

Beanoves example: Uncovered items

Slide 9

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

```
\begin{itemize}
  \item Air
  \begin{itemize}
    \item
    \only<-+>{\transparent{0.3}}
    Chameleo
    \item
    \only<-+>{\transparent{0.3}}
    Gannet
    \end{itemize}
  \item
  \only<-+>{\transparent{0.3}}
  Water
```

- Notice the very handy incremental overlay specifications in `\only<-+>{...}`.
- Unfortunately, they won't help when different layers are involved.

Beanoves example: Uncovered items

Slide 10

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

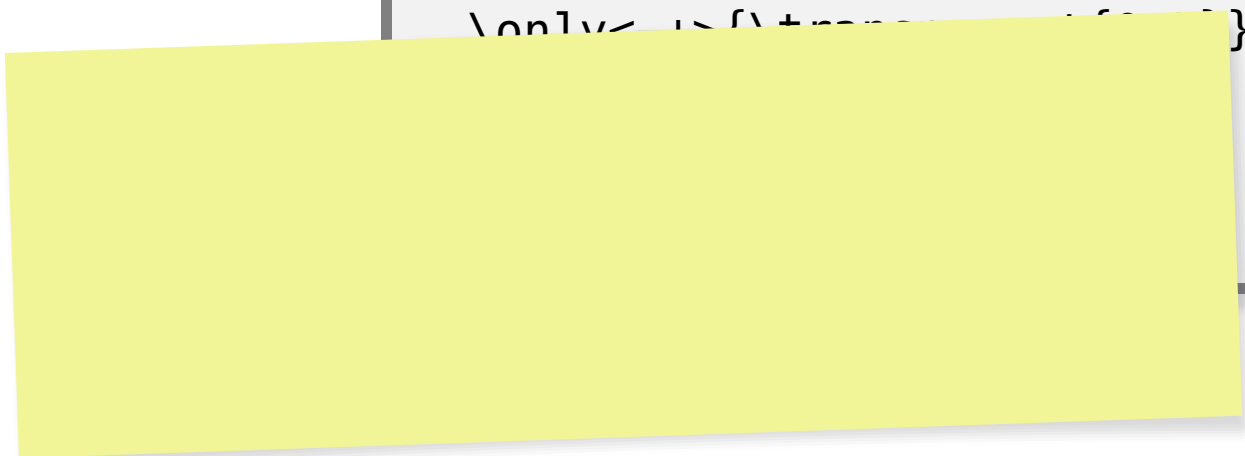
```
\begin{itemize}
  \item Air
  \begin{itemize}
    \item
    \only<-+>{\transparent{0.3}}
    Chameleo
    \item
    \only<-+>{\transparent{0.3}}
    Gannet
    \end{itemize}
  \item
```

Beanoves example: Uncovered items

Slide 11

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

```
\begin{itemize}
  \item Air
  \begin{itemize}
    \item
    \only<-+>{\transparent{0.3}}
    Chameleo
    \item
    \only<-+>{\transparent{0.3}}
    Gannet
    \end{itemize}
  \item
```



Beanoves example: Uncovered items

Slide 12

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

```
\begin{itemize}
  \item Air
  \begin{itemize}
    \item
    \only<-+>{\transparent{0.3}}
    Chameleo
    \item
    \only<-+>{\transparent{0.3}}
    Gannet
    \end{itemize}
  \item
```

Beanoves example: Uncovered items

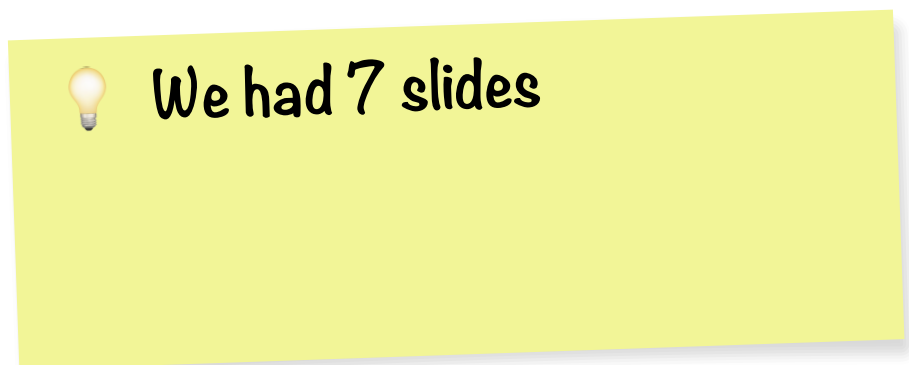
Slide 13

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

```
\begin{itemize}
  \item Air
  \begin{itemize}
    \item
    \only<-+>{\transparent{0.3}}
    Chameleo
    \item
    \only<-+>{\transparent{0.3}}
    Gannet
    \end{itemize}
  \item
  \only<-+>{\transparent{0.3}}
  Water
  ...
\end{itemize}
```

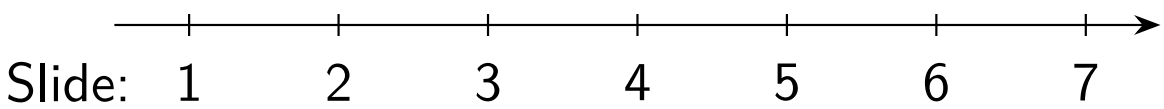
Beanoves example: Uncovered items

Chronology



Beanoves example: Uncovered items

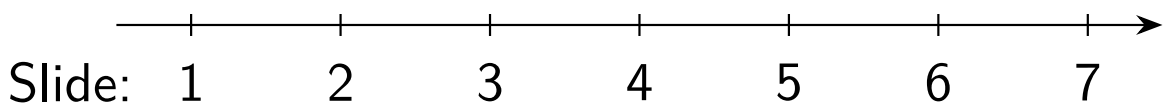
Chronology



We had 7 slides

Beanoves example: Uncovered items

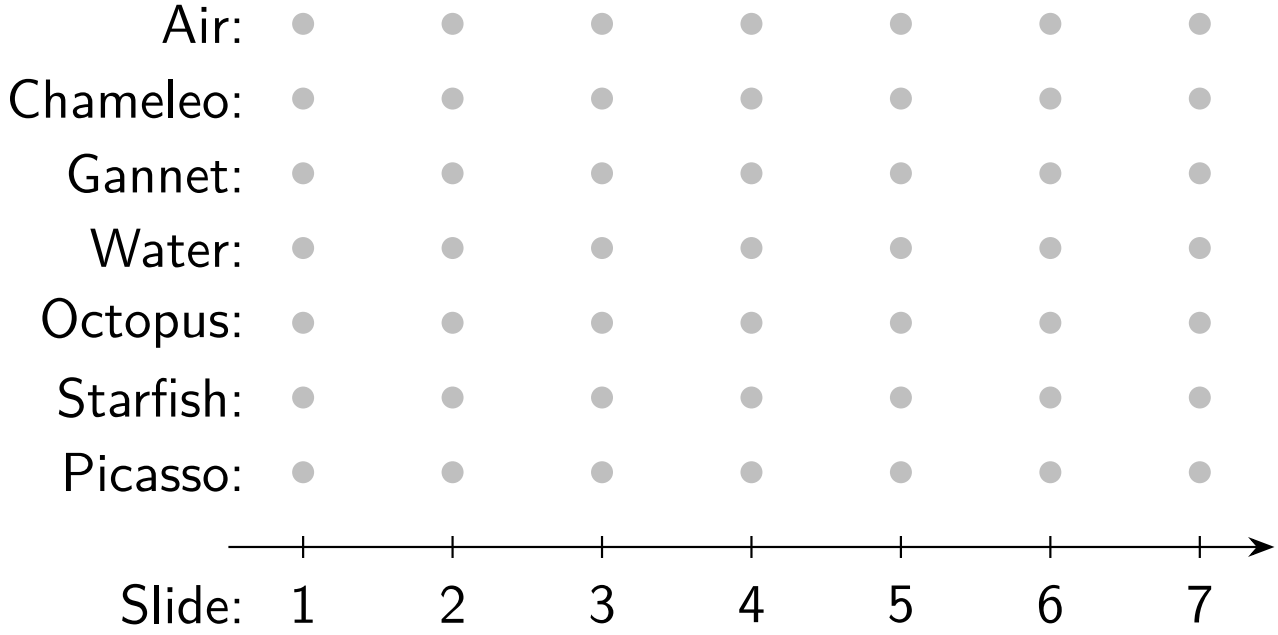
Chronology



We had 7 slides for 7 texts

Beanoves example: Uncovered items

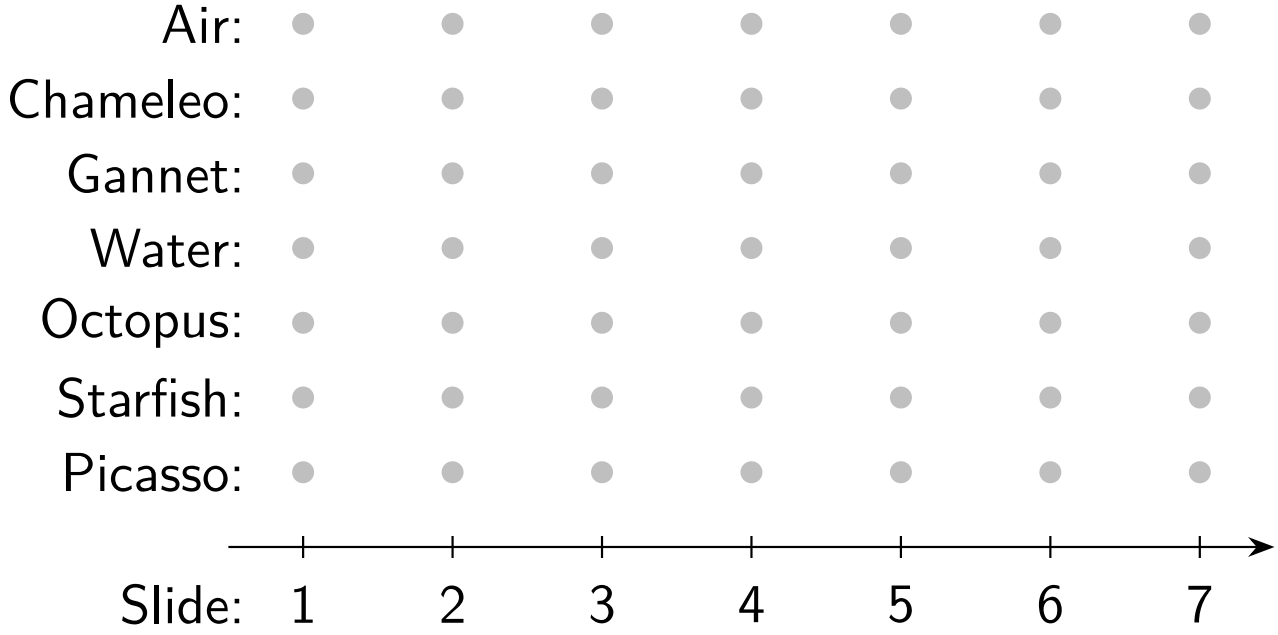
Chronology



We had 7 slides for 7 texts

Beanoves example: Uncovered items

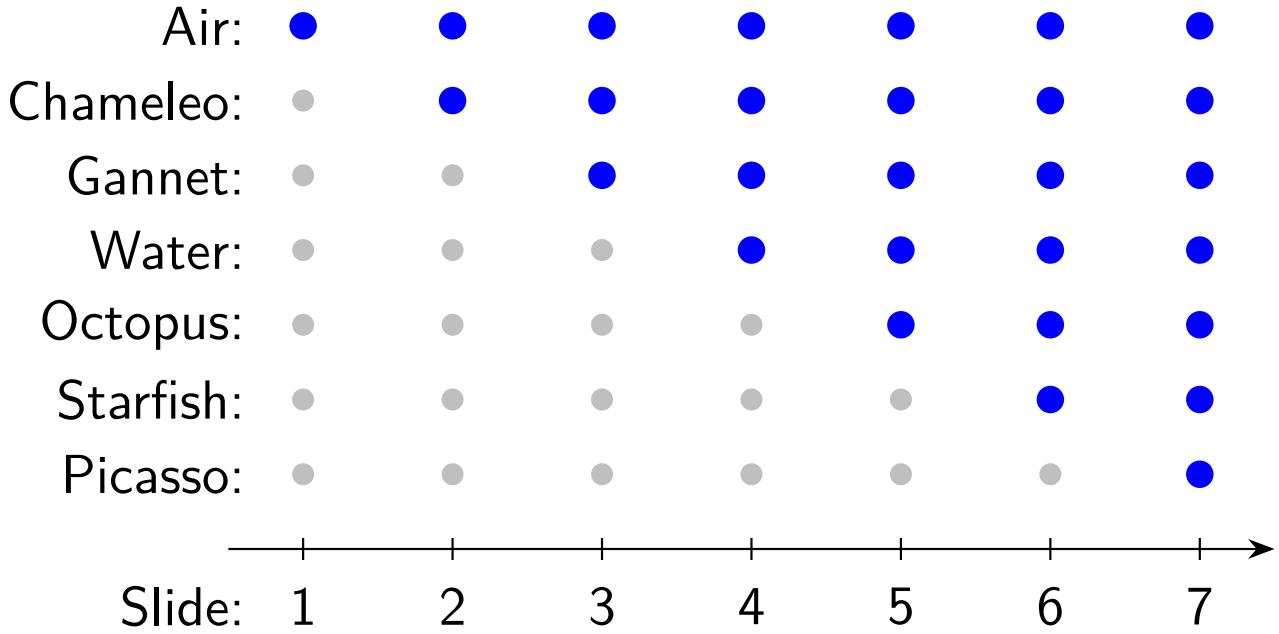
Chronology



- We had 7 slides for 7 texts
- Blue dots for uncovered text

Beanoves example: Uncovered items

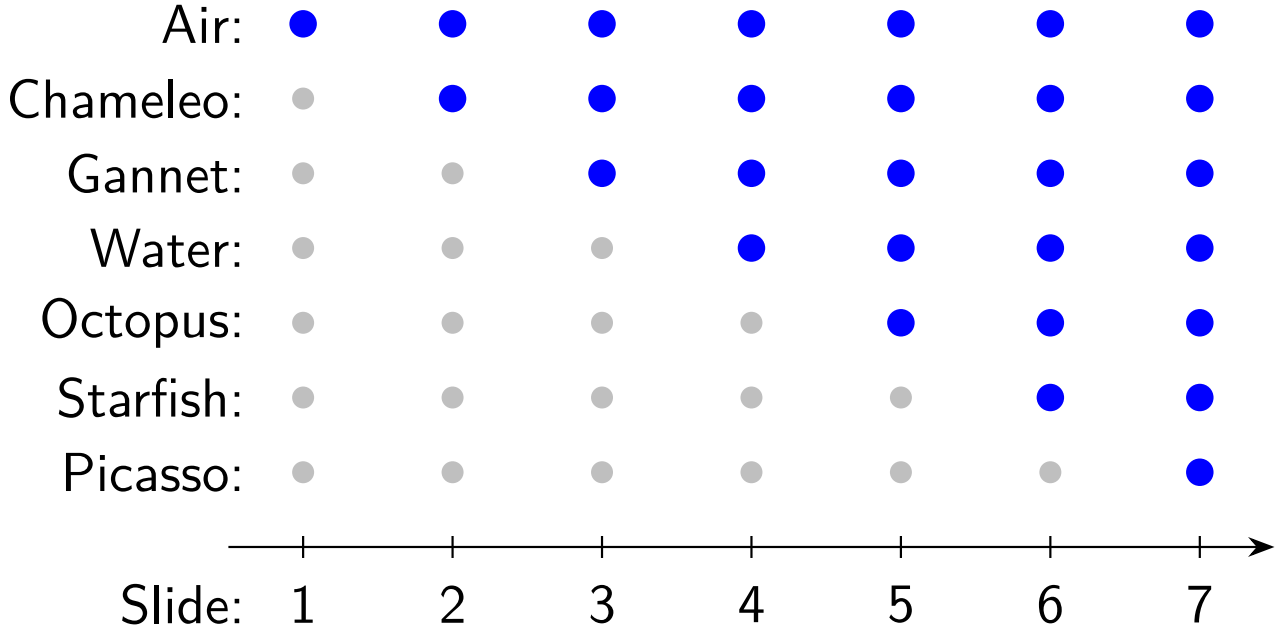
Chronology



- We had 7 slides for 7 texts
- Blue dots for uncovered text

Beanoves example: Uncovered items

Chronology



- We had 7 slides for 7 texts
- Blue dots for uncovered text
- Very regular behavior

Beanoves example: Uncovered items + images

Slide 1

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

Beanoves example: Uncovered items + images

Slide 2

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

Beanoves example: Uncovered items + images

Slide 3

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

Beanoves example: Uncovered items + images

Slide 4

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

Notice how the chameleon image is transparent

Beanoves example: Uncovered items + images

Slide 5

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

Beanoves example: Uncovered items + images

Slide 6

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

Beanoves example: Uncovered items + images

Slide 7

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

Beanoves example: Uncovered items + images

Slide 8

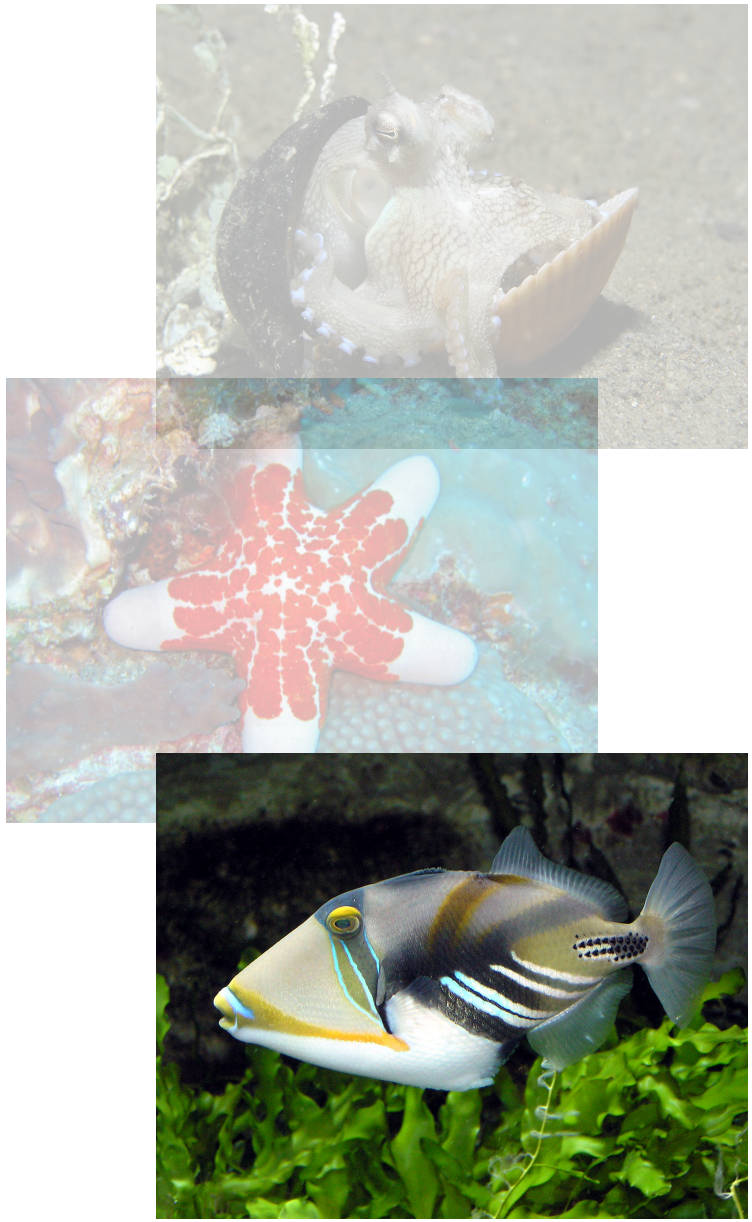
- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

Overlap allows for larger images

Beanoves example: Uncovered items + images

Slide 9

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish



Beanoves example: Uncovered items + images

Slide 10

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

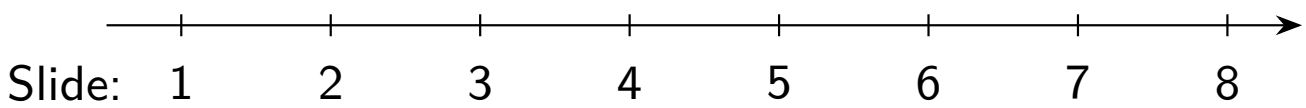
Beanoves example: Uncovered items + images

Chronology

- We had 8 slides

Beanoves example: Uncovered items + images

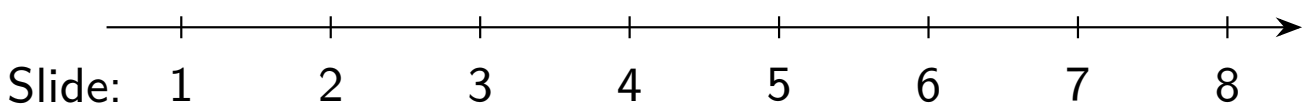
Chronology



- We had 8 slides

Beanoves example: Uncovered items + images

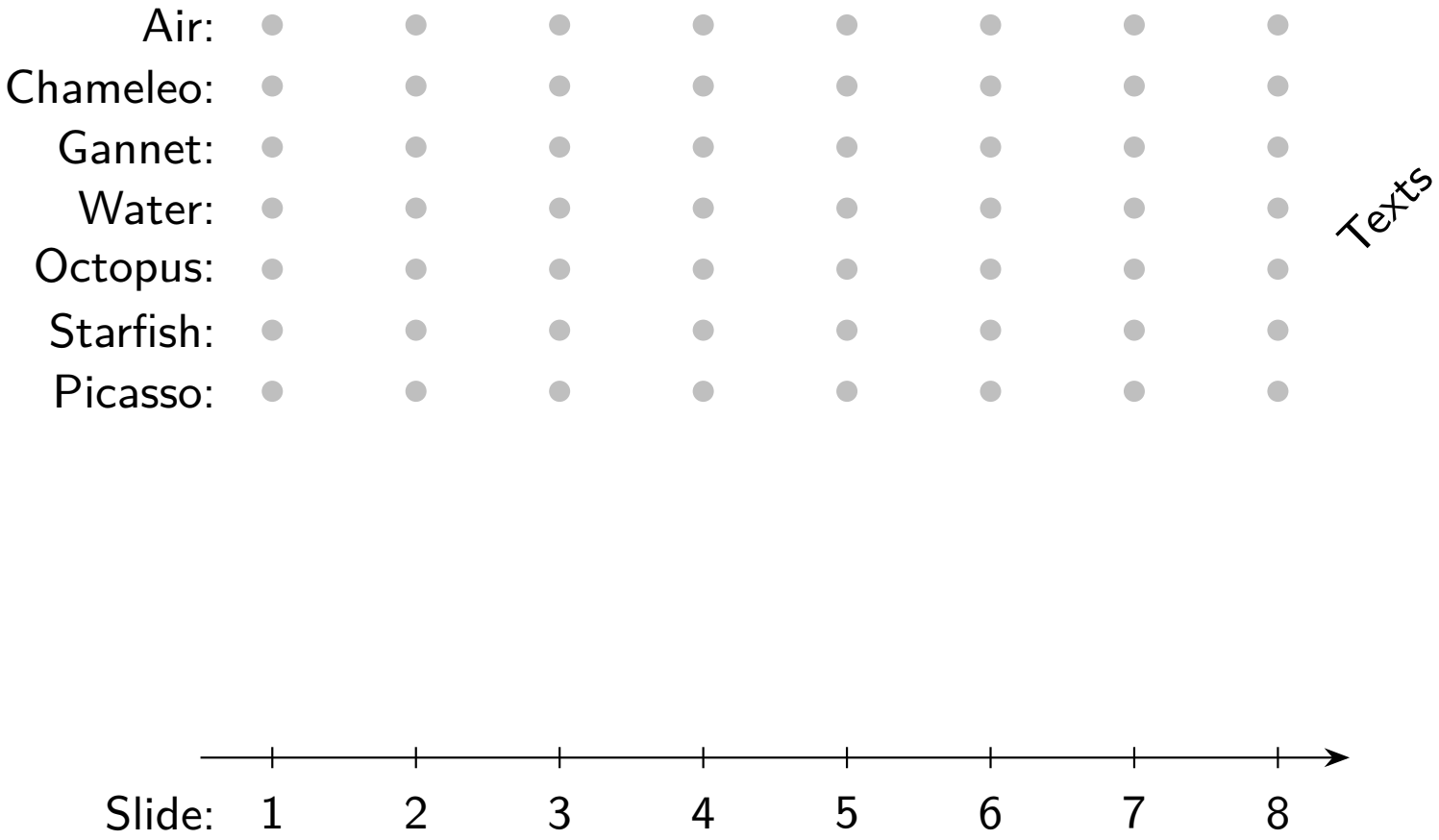
Chronology



- We had 8 slides for 7 texts

Beanoves example: Uncovered items + images

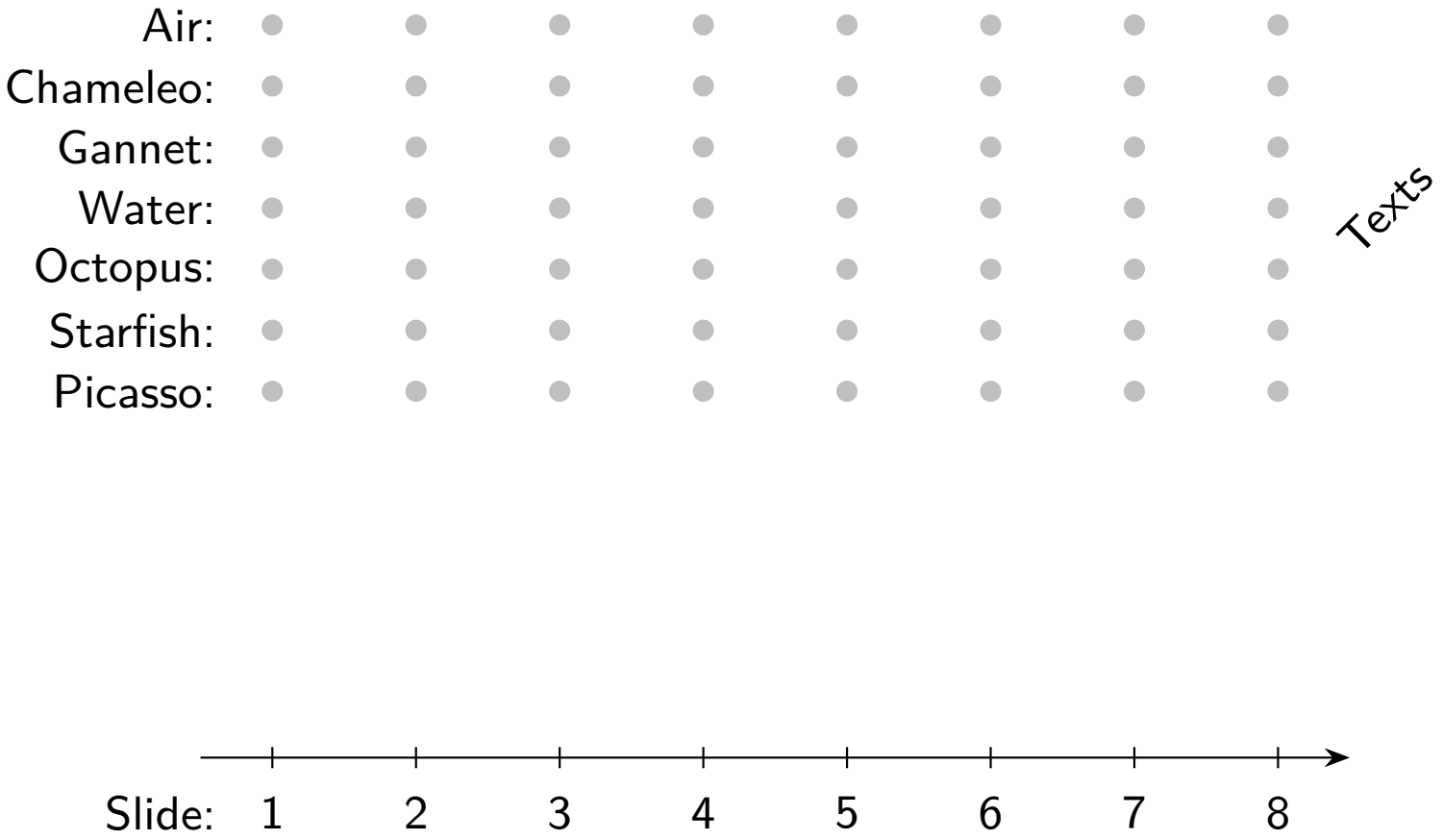
Chronology



- We had 8 slides for 7 texts

Beanoves example: Uncovered items + images

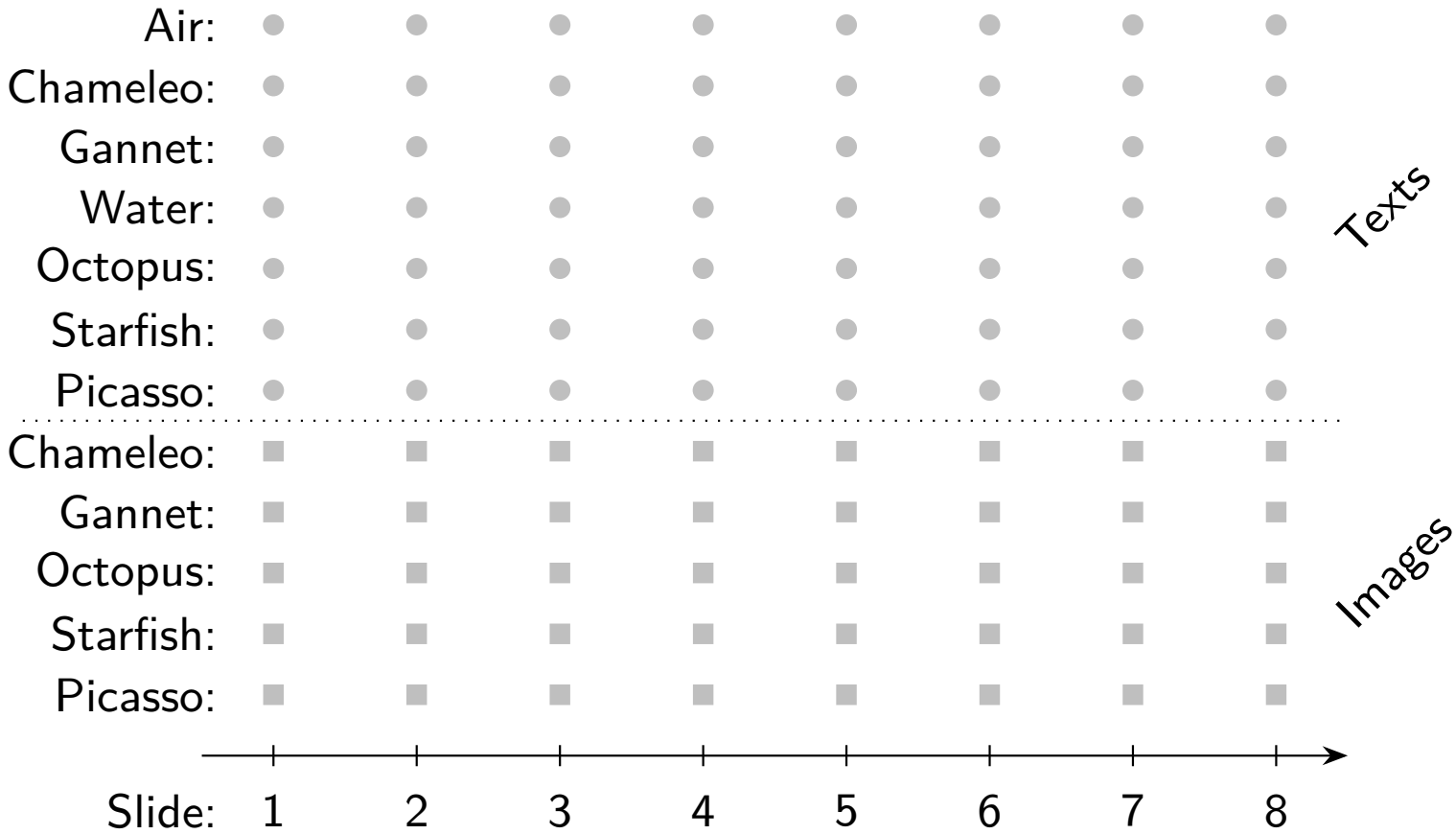
Chronology



- We had 8 slides for 7 texts and 5 images

Beanoves example: Uncovered items + images

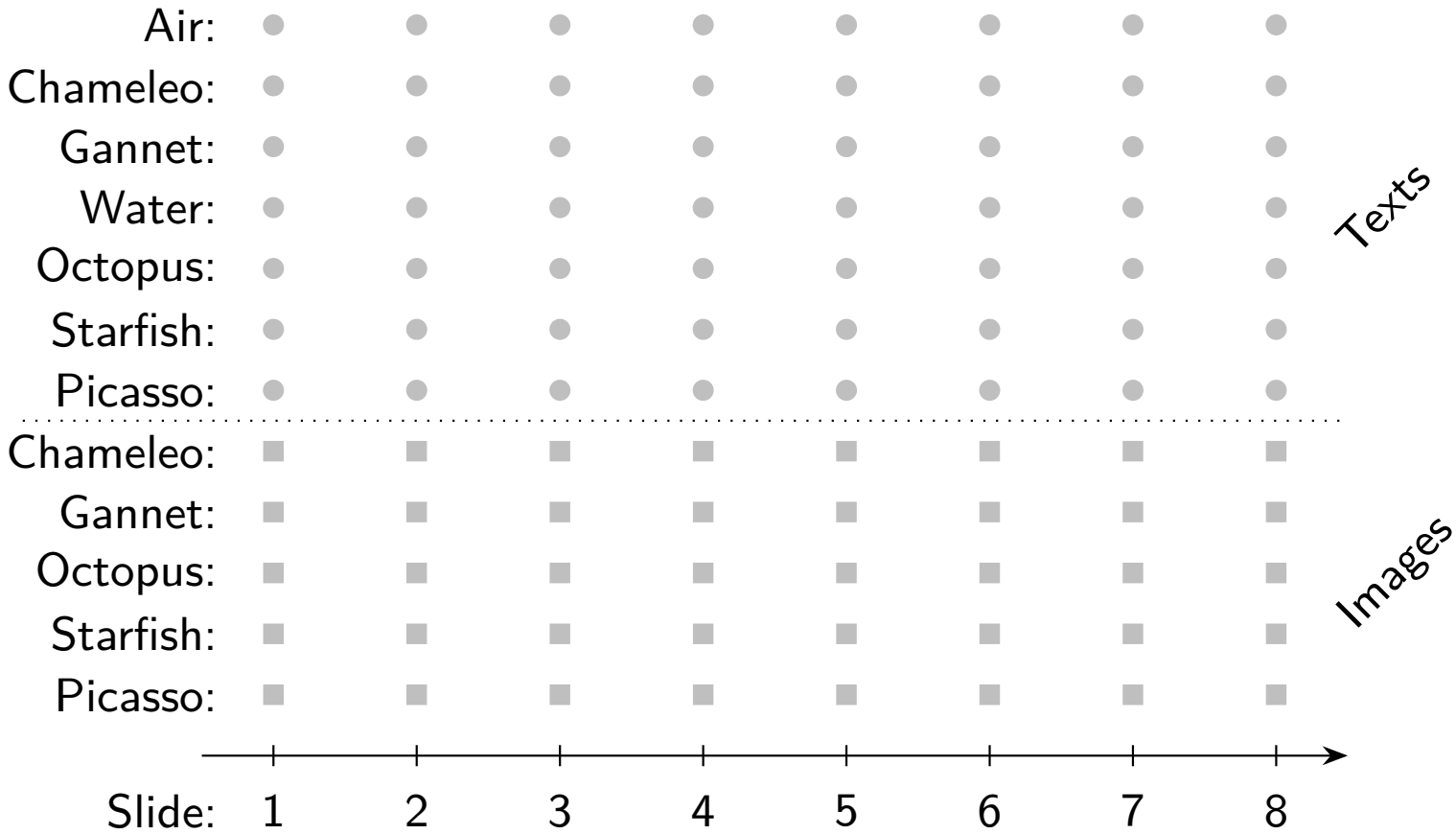
Chronology



- We had 8 slides for 7 texts and 5 images

Beanoves example: Uncovered items + images

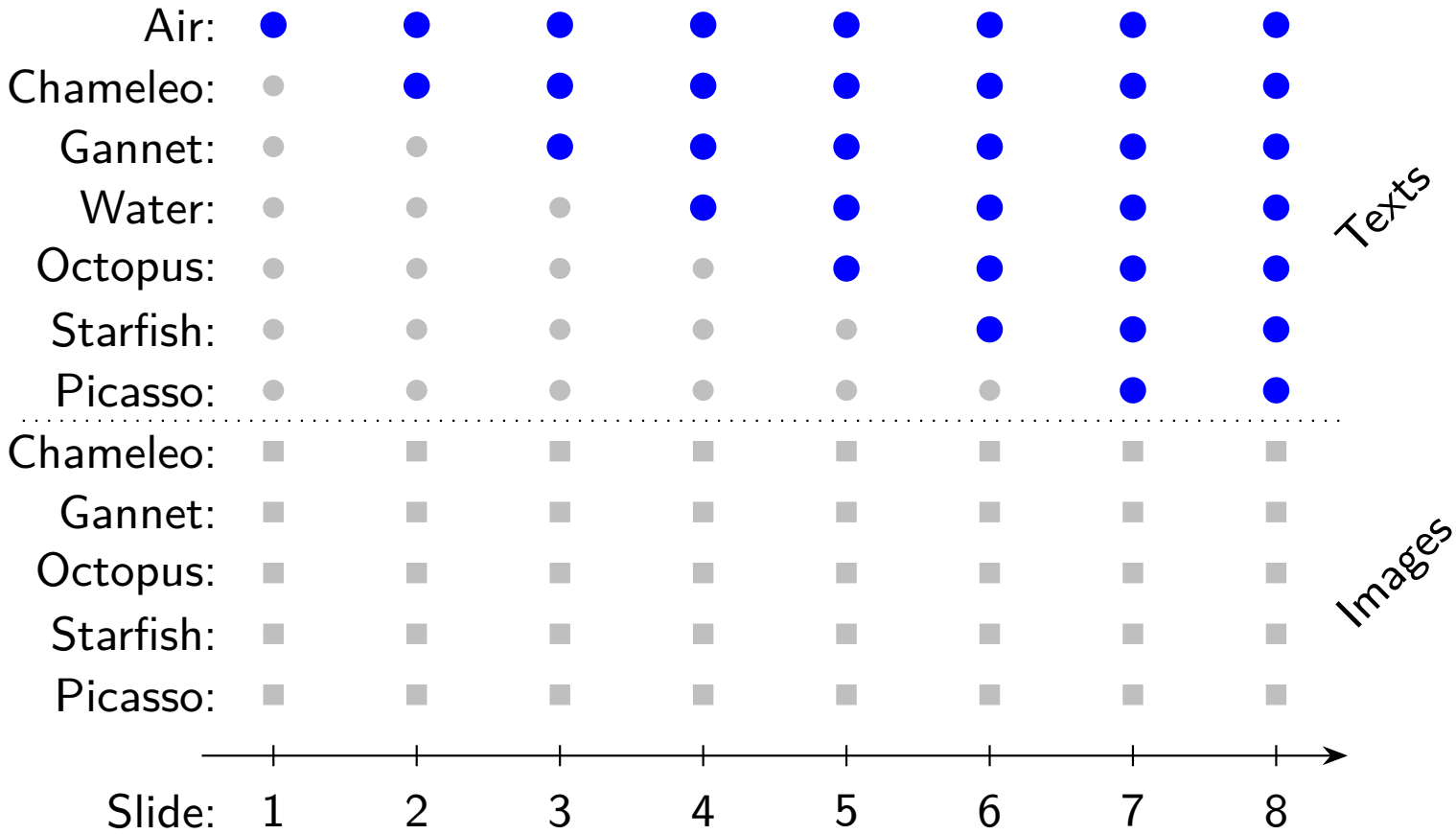
Chronology



- Blue dots for uncovered text

Beanoves example: Uncovered items + images

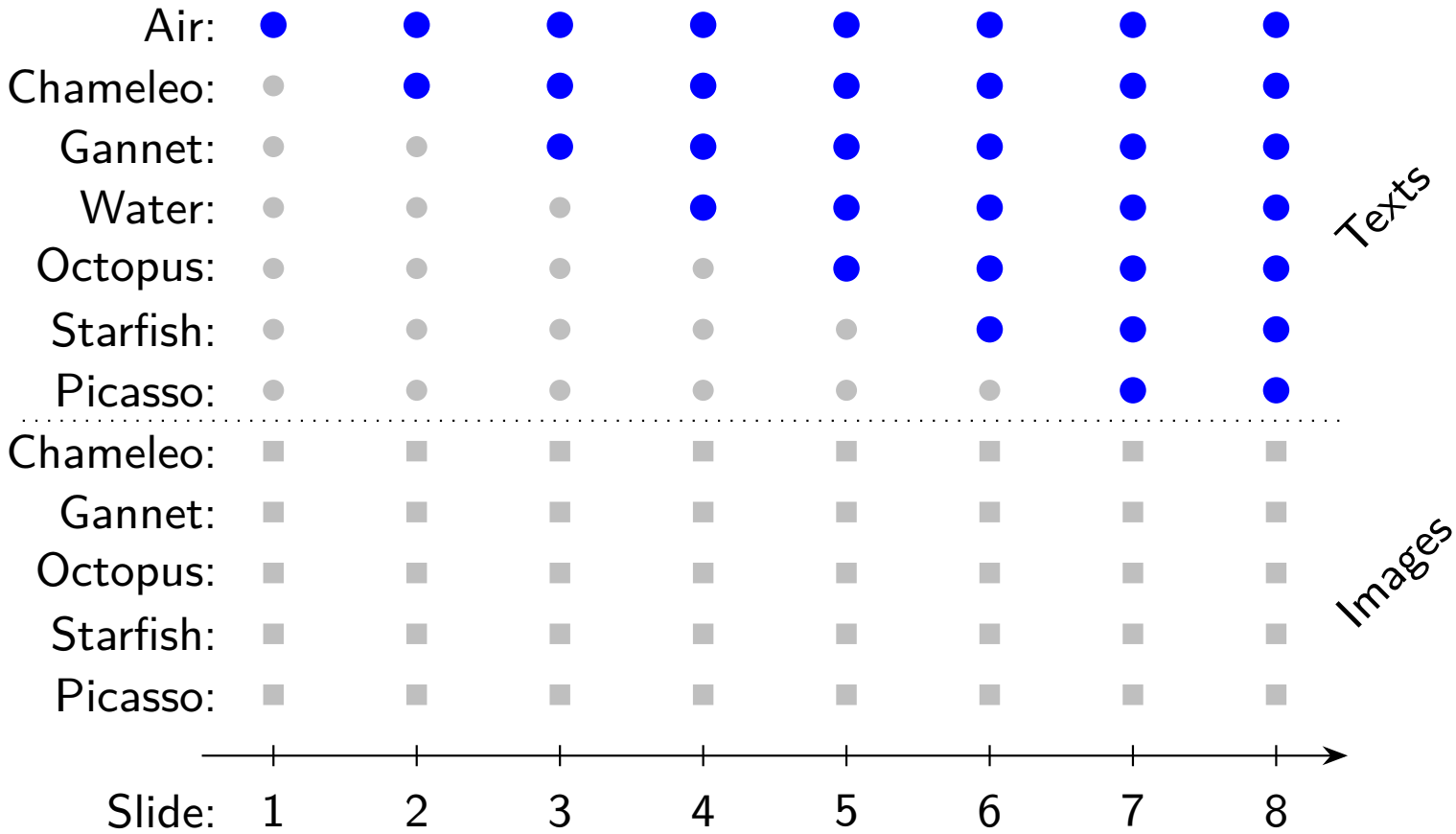
Chronology



- Blue dots for uncovered text

Beanoves example: Uncovered items + images

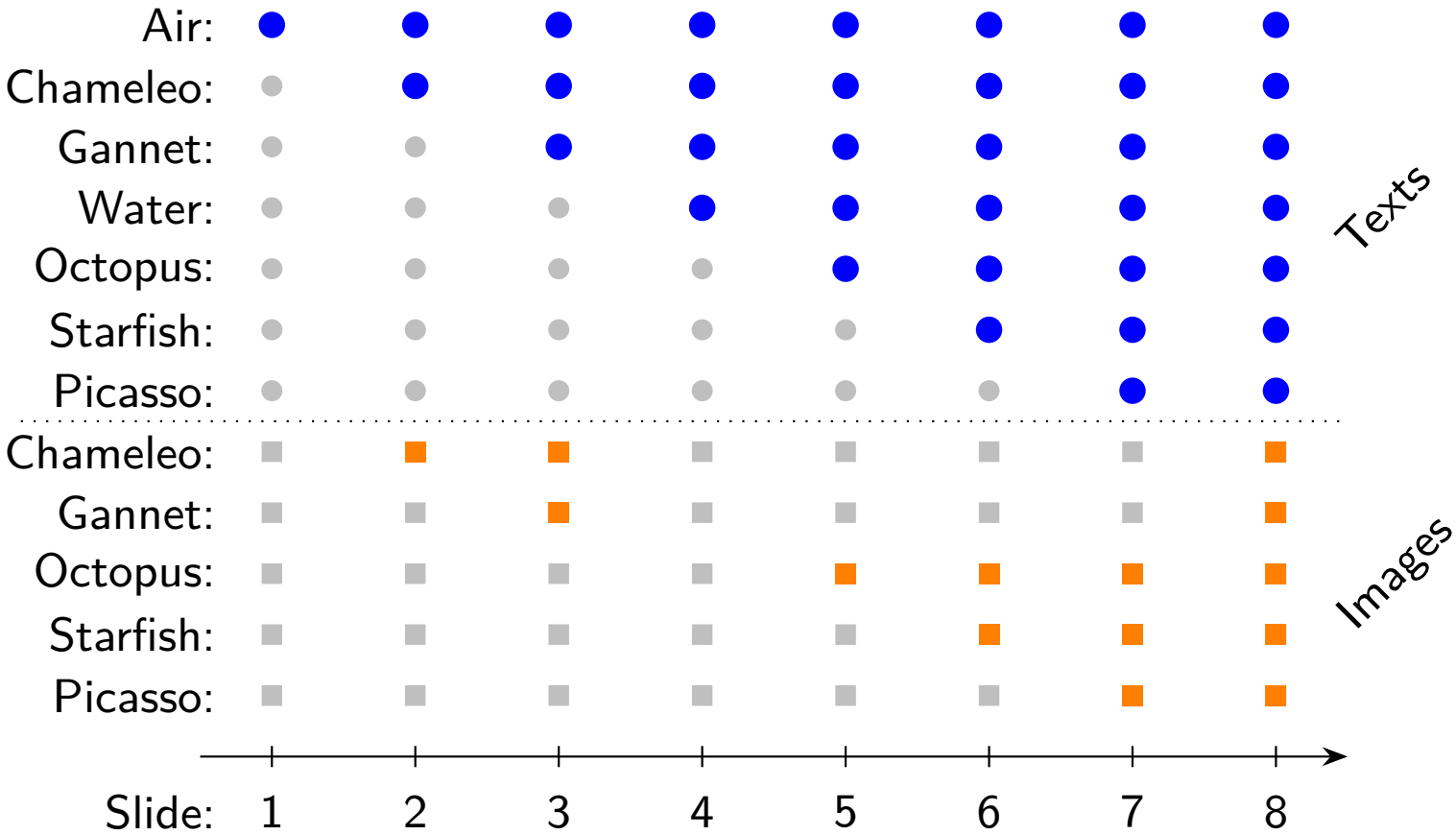
Chronology



- orange squares for uncovered images

Beanoves example: Uncovered items + images

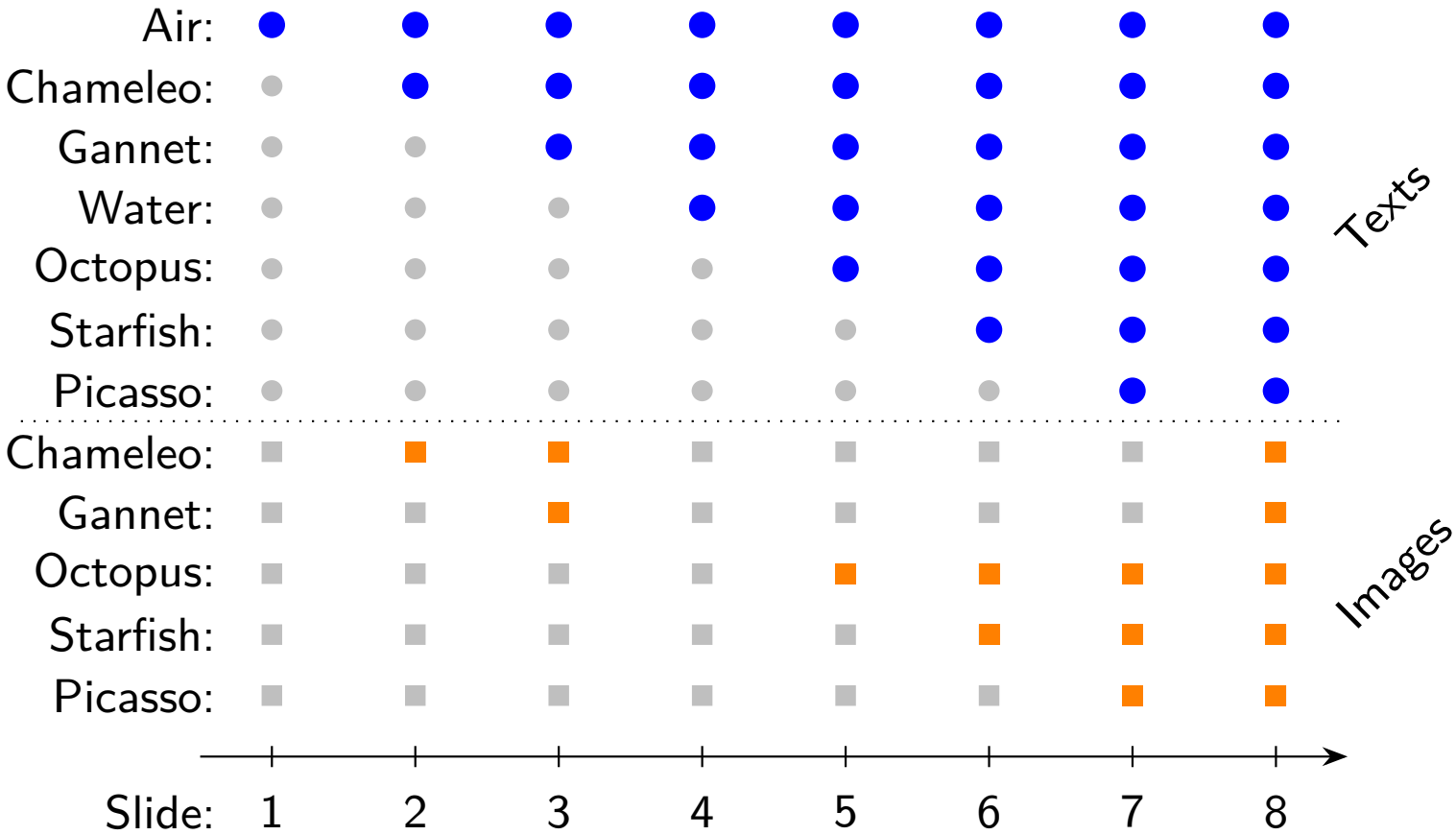
Chronology



- orange squares for uncovered images

Beanoves example: Uncovered items + images

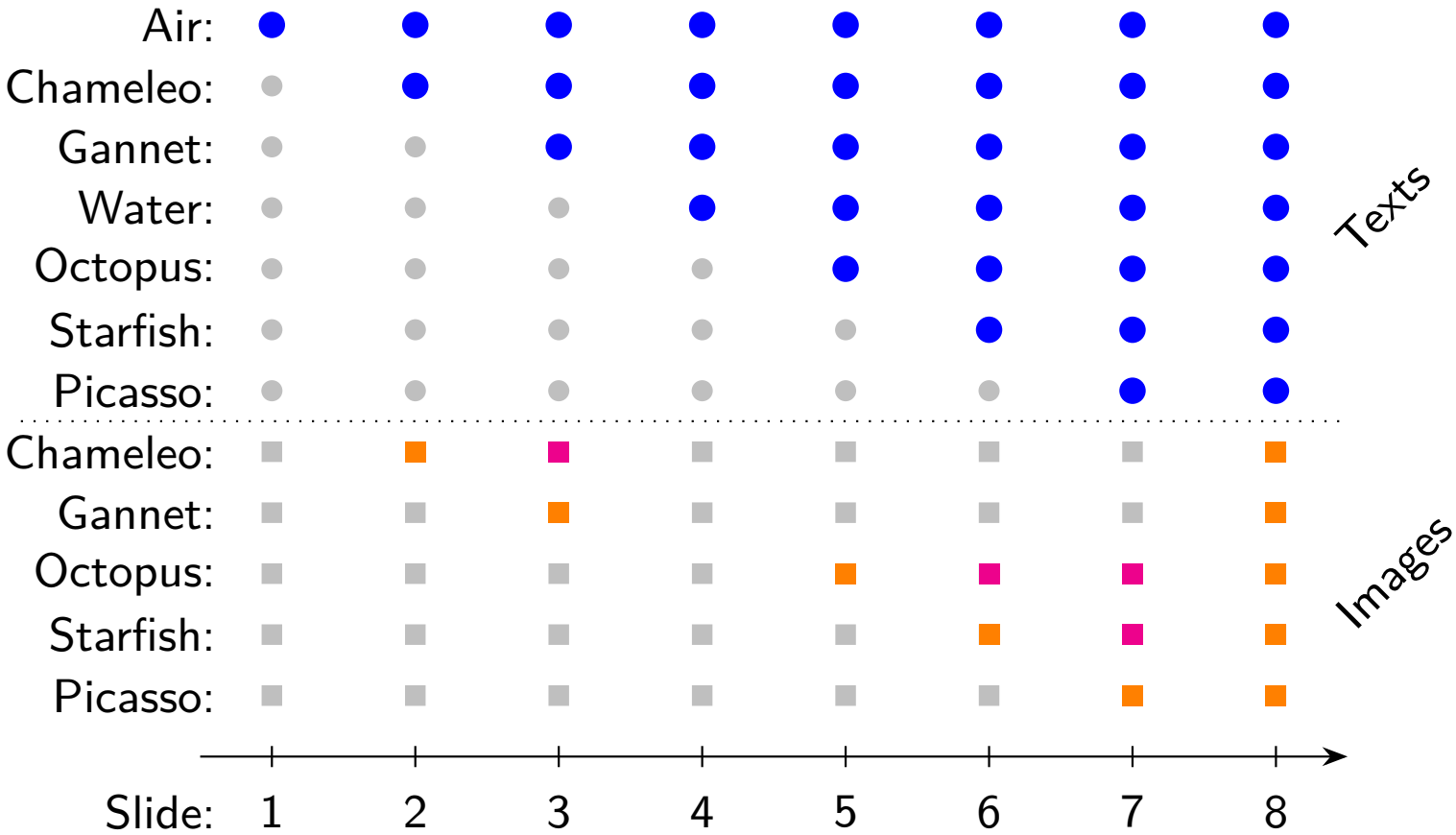
Chronology



- magenta squares for uncovered transparent images

Beanoves example: Uncovered items + images

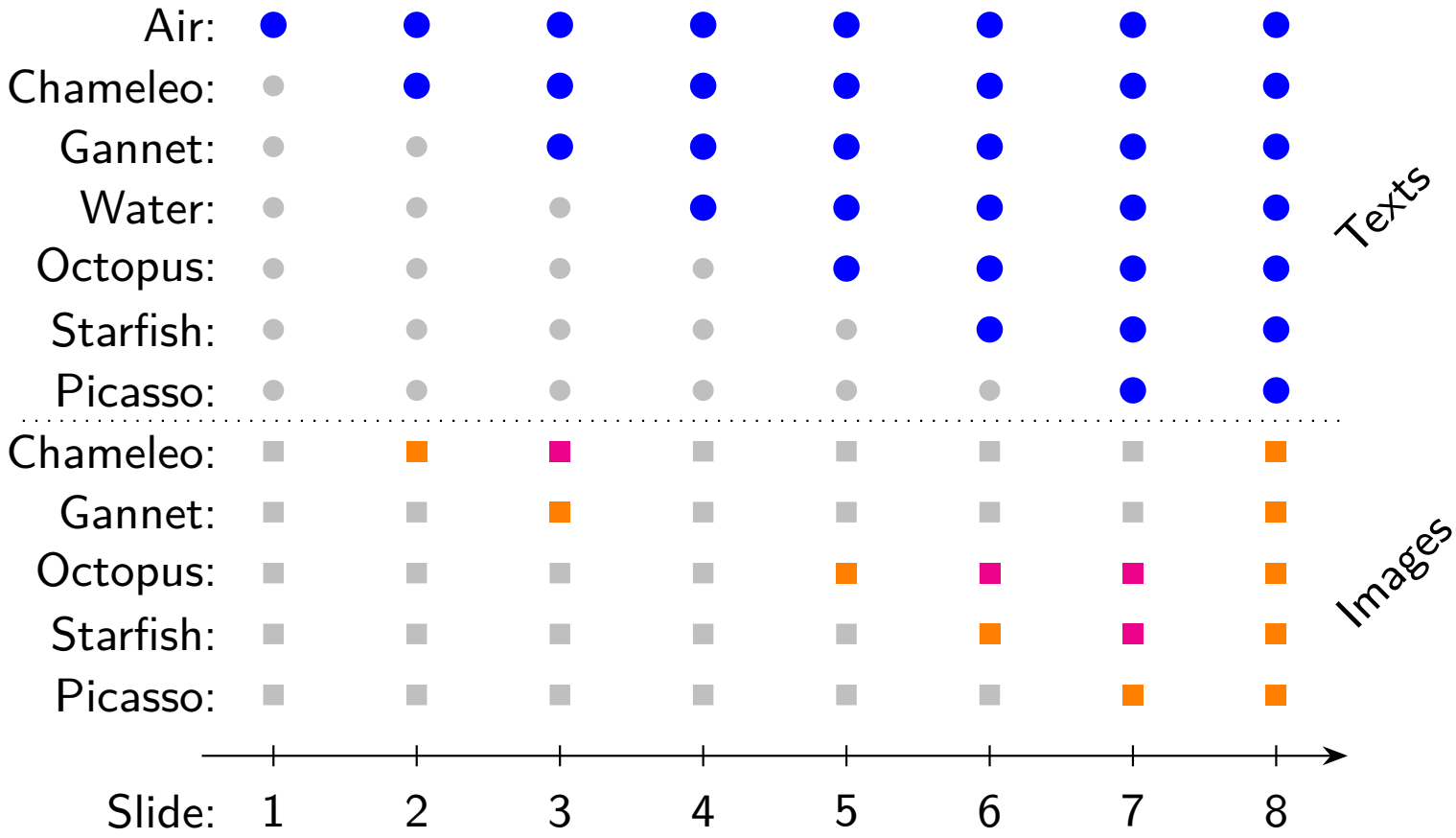
Chronology



- magenta squares for uncovered transparent images

Beanoves example: Uncovered items + images

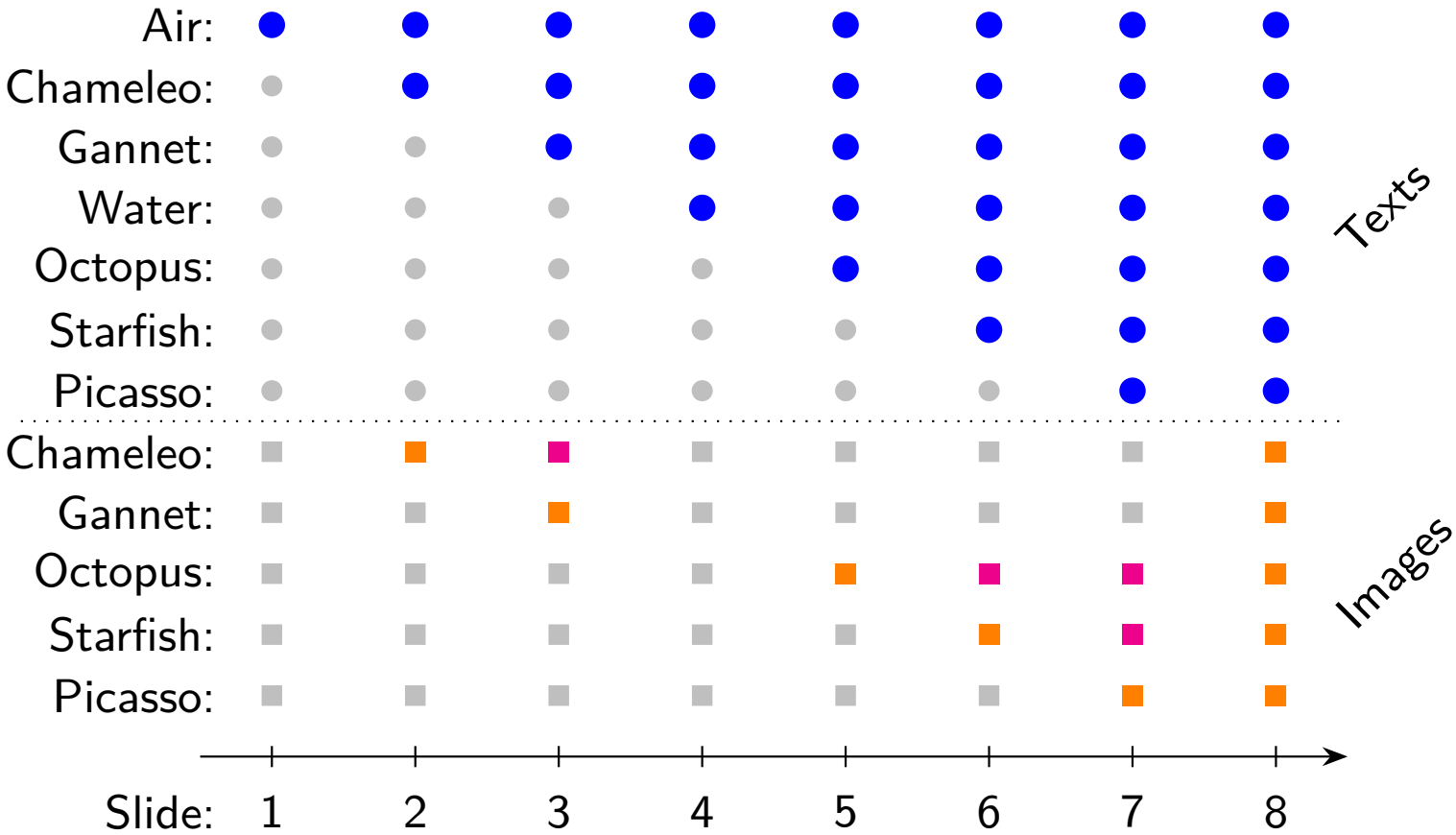
Chronology



Let us organize all this

Beanoves example: Uncovered items + images

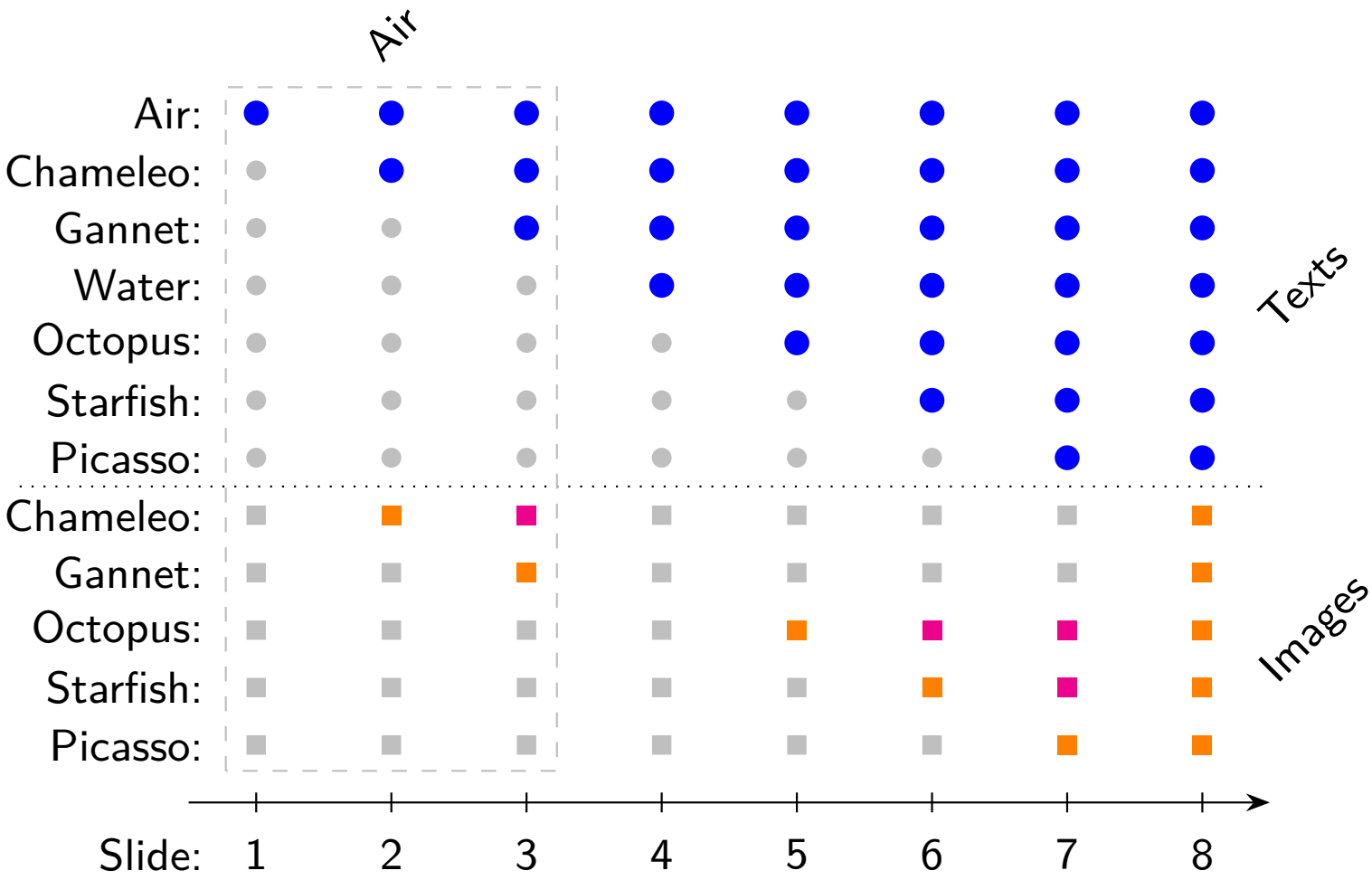
Chronology



- 3 slides for the Air, 4 for the Water, 1 to summarize

Beanoves example: Uncovered items + images

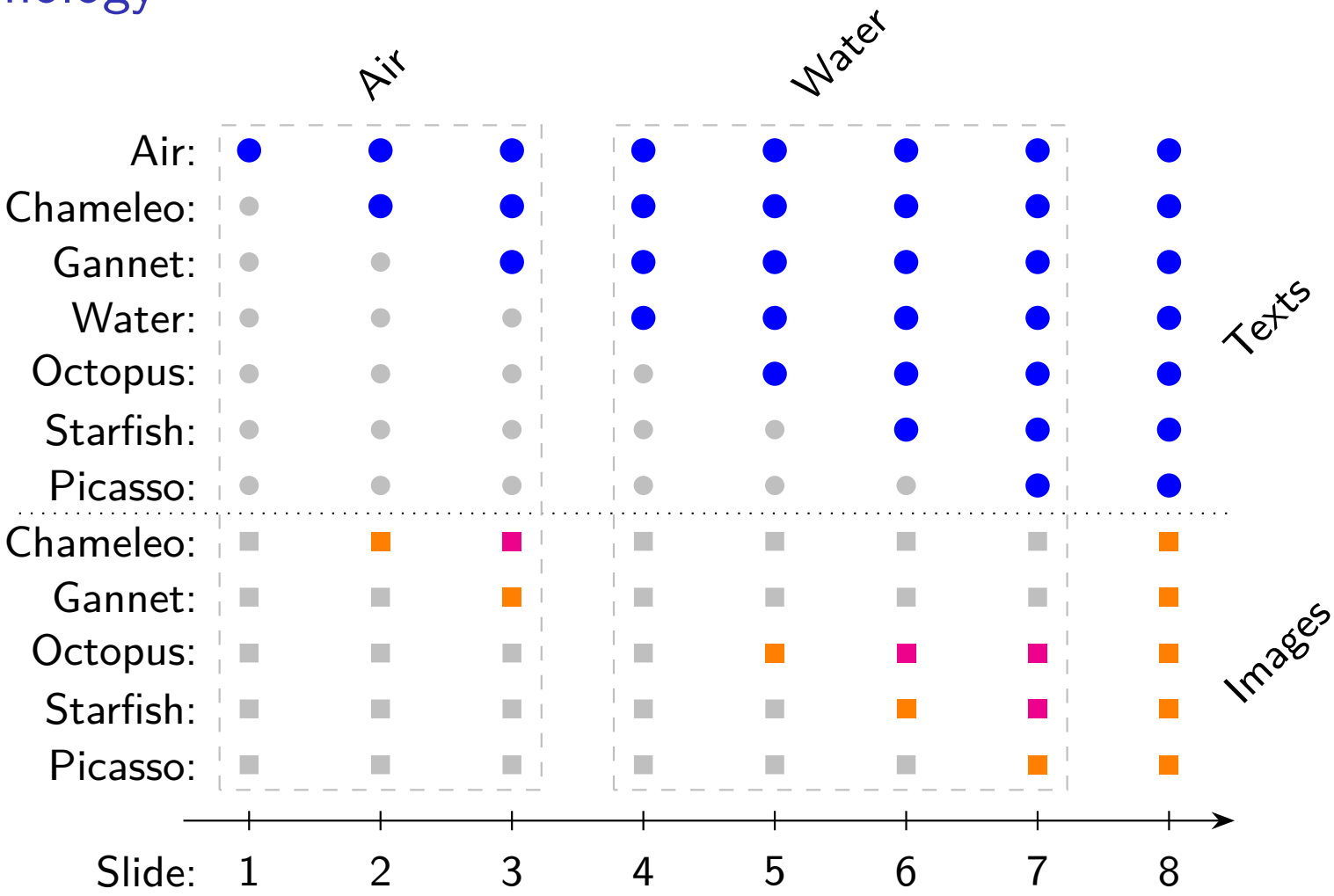
Chronology



- 3 slides for the Air, 4 for the Water, 1 to summarize

Beanoves example: Uncovered items + images

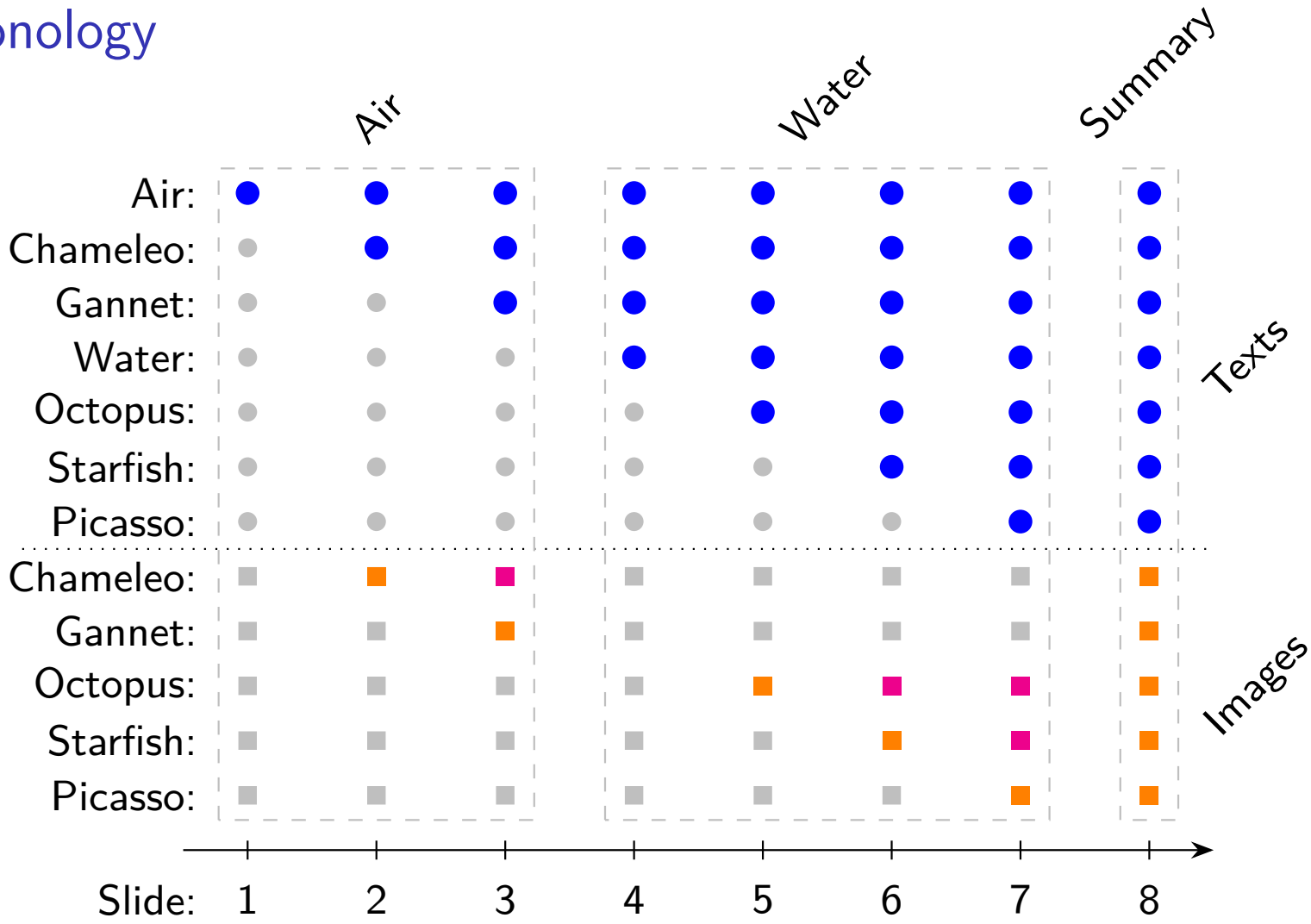
Chronology



- 3 slides for the Air, 4 for the Water, 1 to summarize

Beanoves example: Uncovered items + images

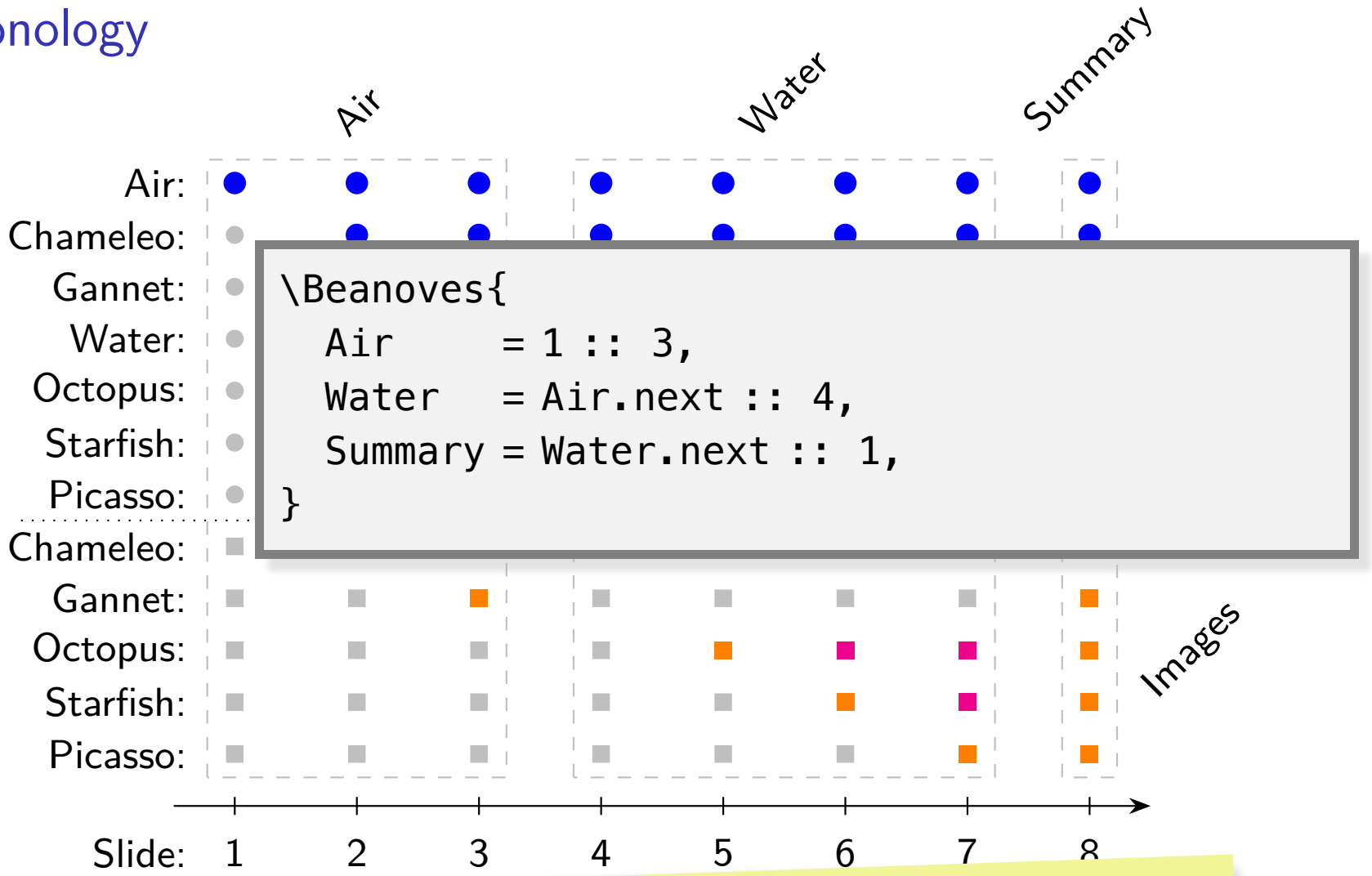
Chronology



- 3 slides for the Air, 4 for the Water, 1 to summarize

Beanoves example: Uncovered items + images

Chronology

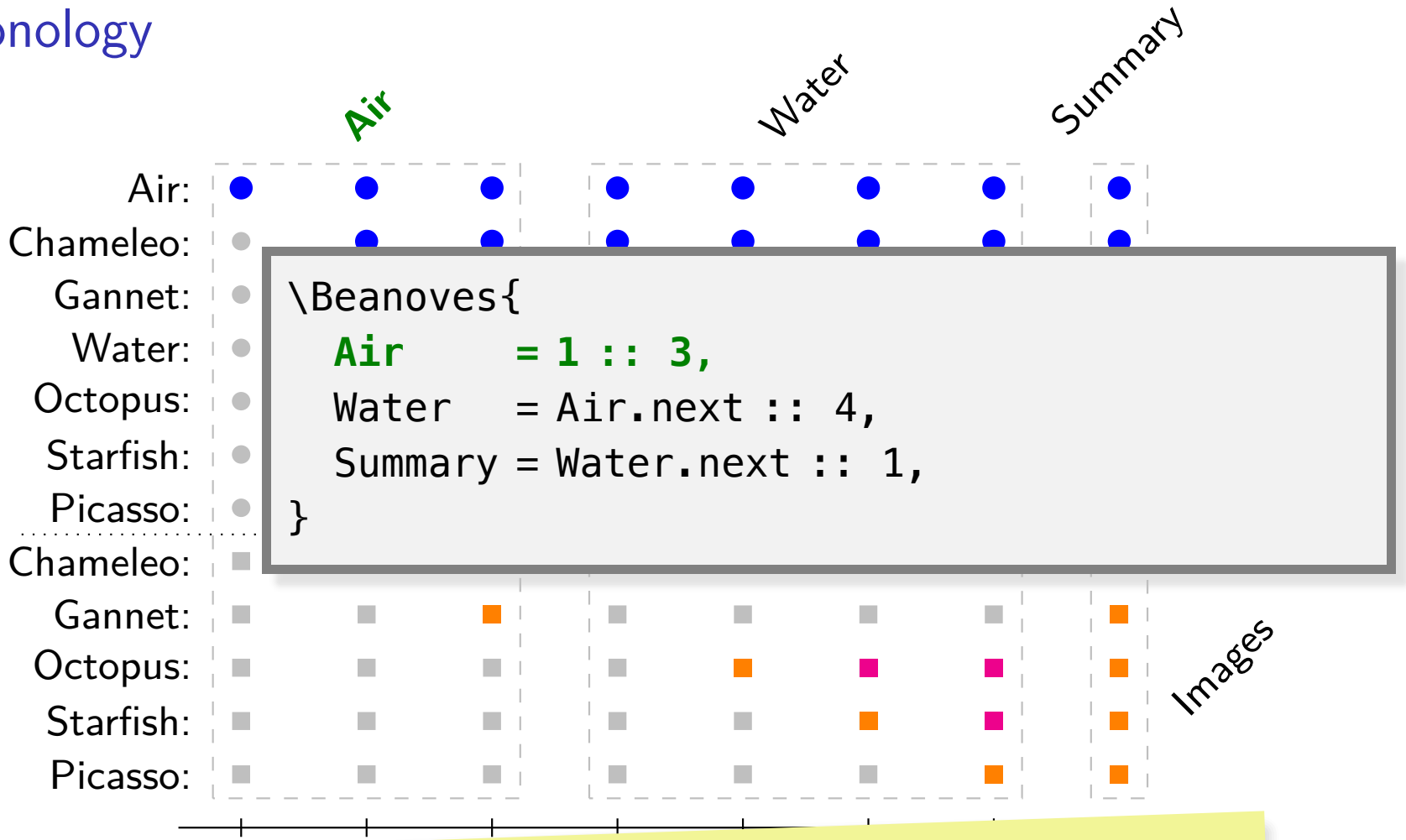



```
\Beanoves{
  Air     = 1 :: 3,
  Water   = Air.next :: 4,
  Summary = Water.next :: 1,
}
```

- This code declares the logical overlay ranges
- Before the frame source

Beanoves example: Uncovered items + images

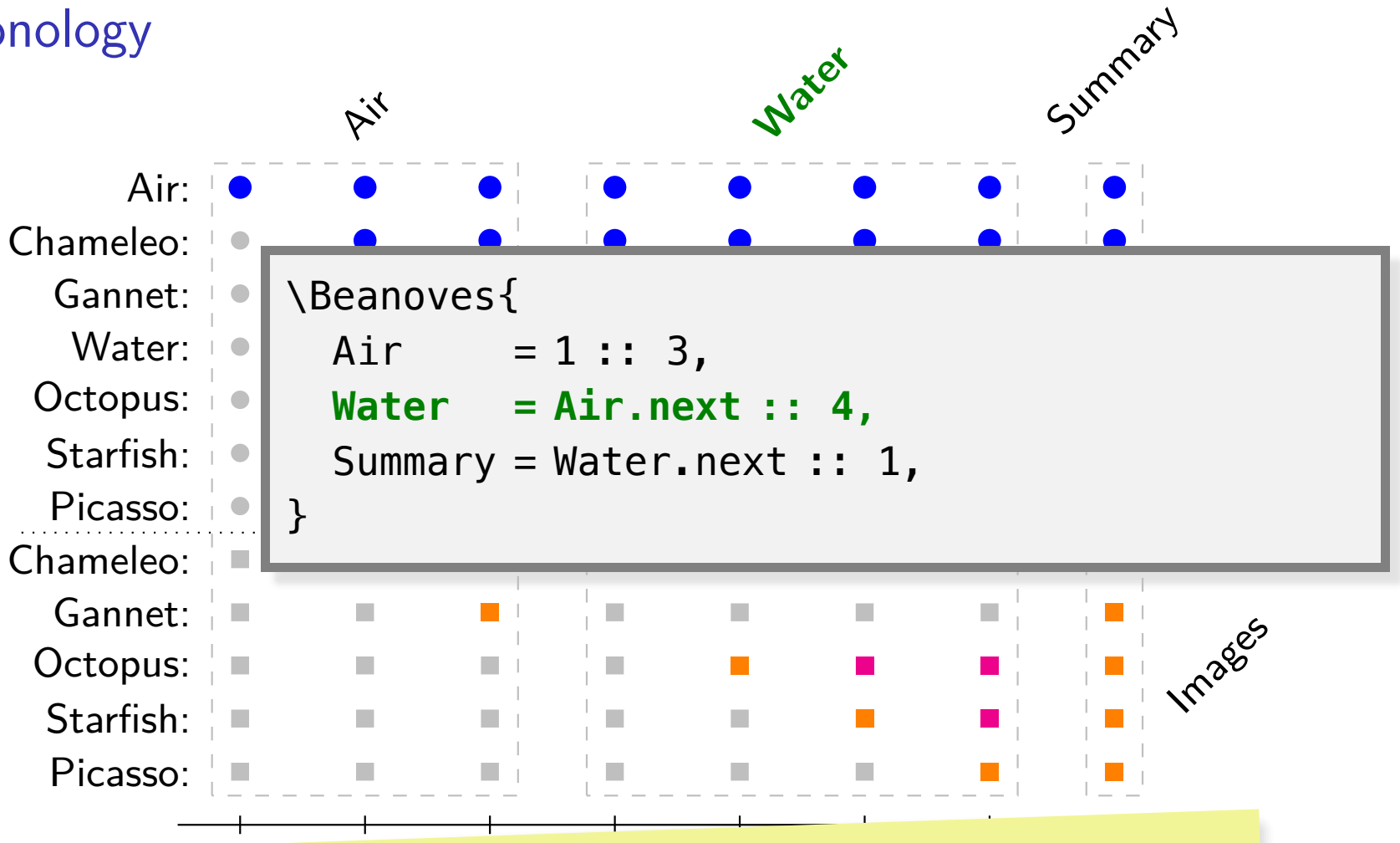
Chronology



- `Air` is a variable like name
- `1` is the position of the first slide
- `3` is the duration (or length) of the range

Beanoves example: Uncovered items + images

Chronology

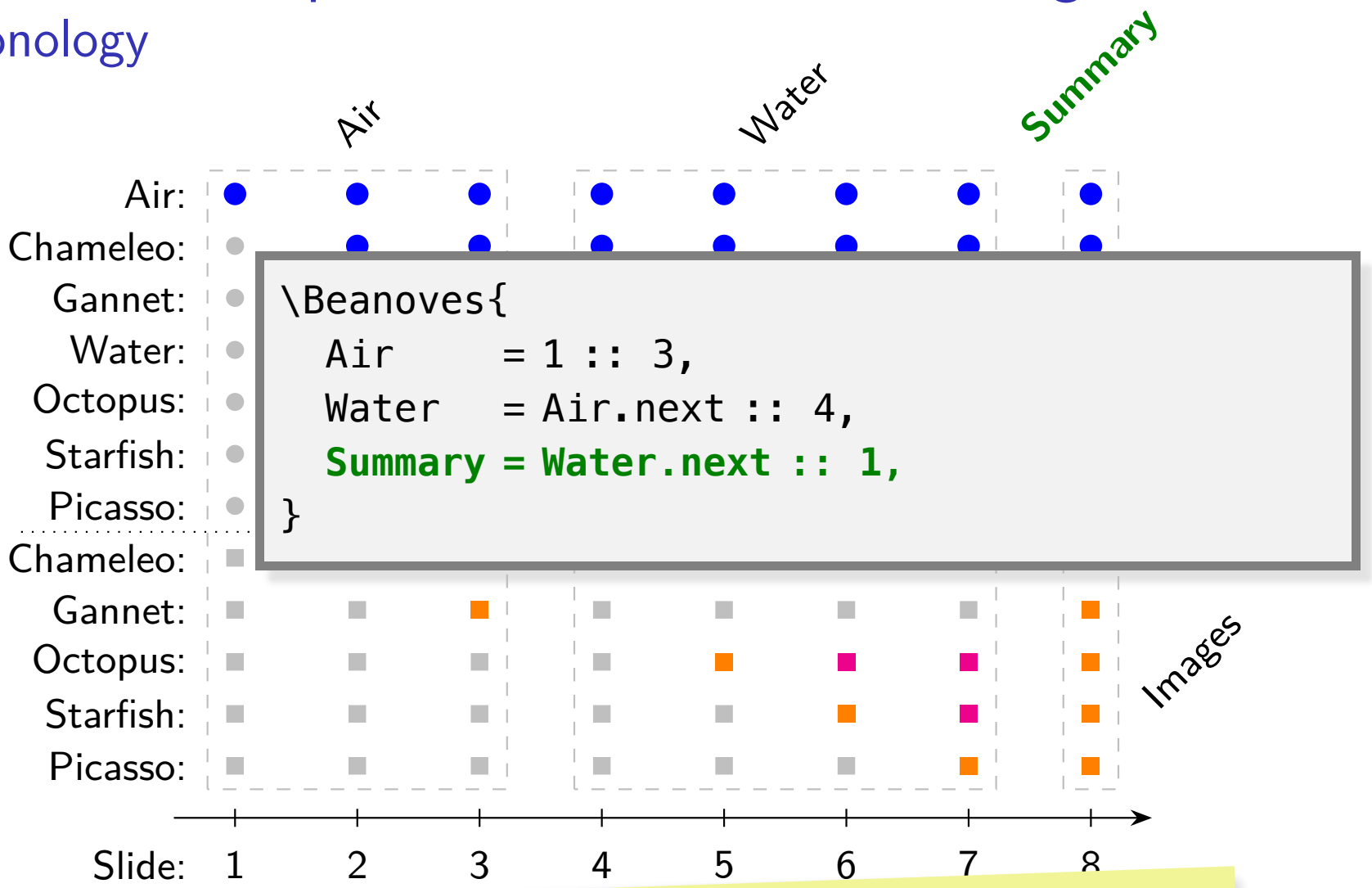



```
\Beanoves{
  Air     = 1 :: 3,
  Water   = Air.next :: 4,
  Summary = Water.next :: 1,
}
```

- The "Water" topic follows the "Air" topic:
- `Air.next` is the first slide after the `Air` range
- `4` is the duration (or length)

Beanoves example: Uncovered items + images

Chronology

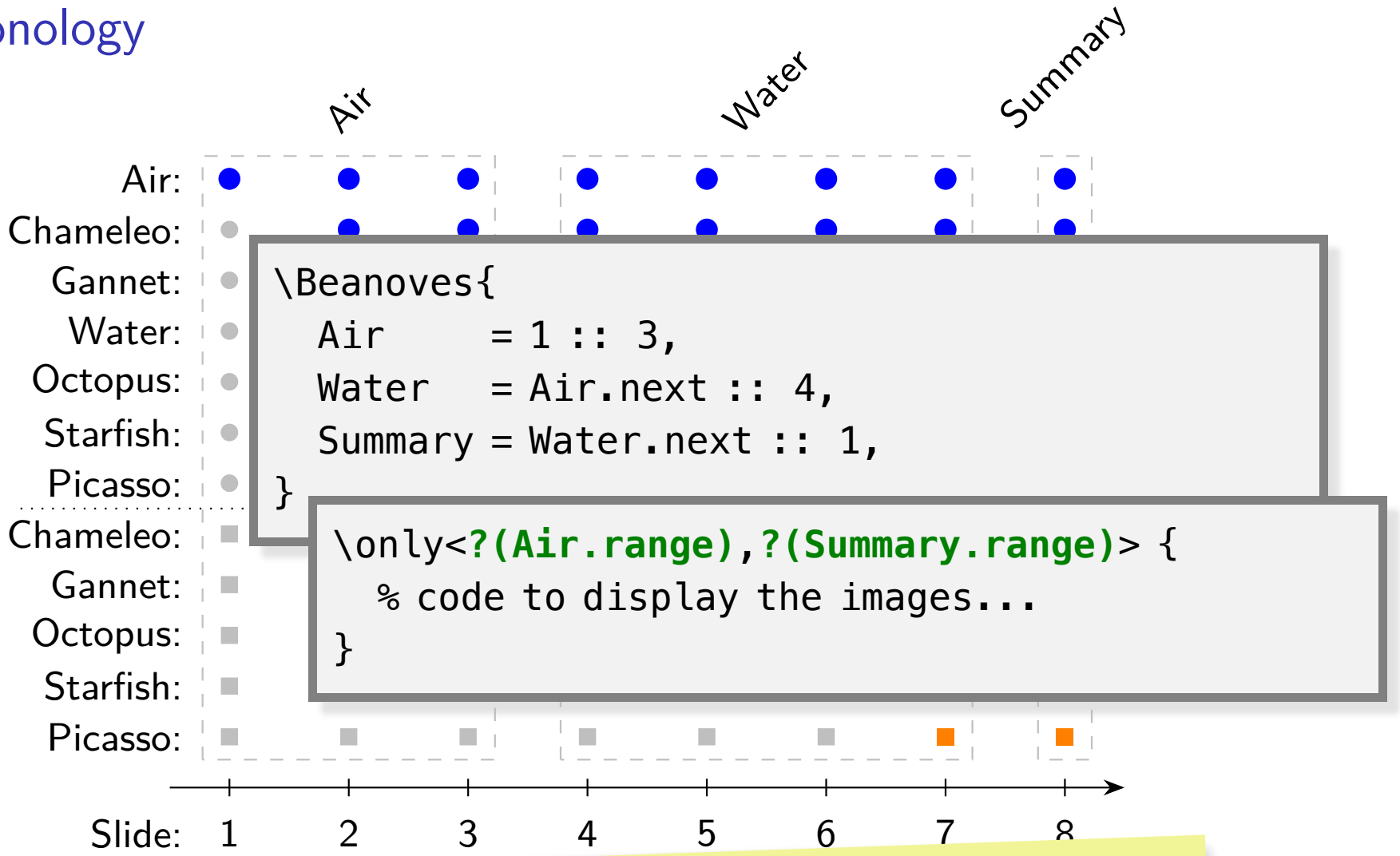



```
\Beanoves{
  Air     = 1 :: 3,
  Water   = Air.next :: 4,
  Summary = Water.next :: 1,
}
```

- The "Summary" follows the "Water" topic:
- `Water.next` is the first slide after the `Water` range

Beanoves example: Uncovered items + images

Chronology



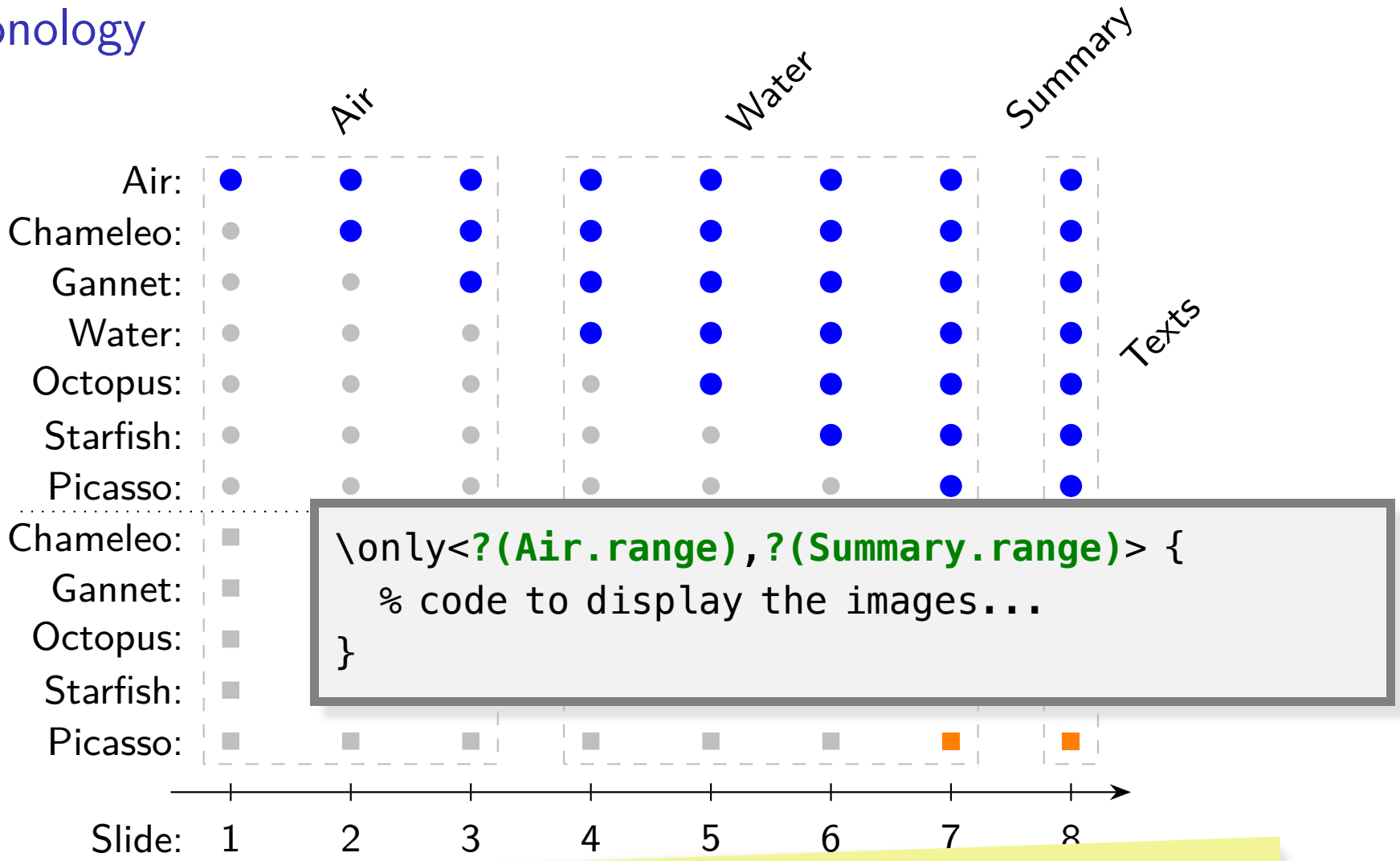

```
\Beanoves{
  Air     = 1 :: 3,
  Water   = Air.next :: 4,
  Summary = Water.next :: 1,
}
```

```
\only<?(Air.range),?(Summary.range)> {
  % code to display the images...
}
```

To limit "Air" topic images to the `Air` and `Summary` ranges, use a logical overlay specification...

Beanoves example: Uncovered items + images

Chronology




```
\only<?(Air.range),?(Summary.range)> {
  % code to display the images...
}
```

- `?(Air.range)` is automatically replaced by `1-3`
- `?(Summary.range)` is automatically replaced by `8-8`

Beanoves example: Uncovered items + images

Step back

- Air
  - Chameleo
  - Gannet
- Water
  - Octopus
  - Starfish
  - Picasso fish

- On next slide, the "Octopus" image is expected to be transparent
- Let us see the source code

Beanoves example: Uncovered items + images

Step back

- Air
  - Chameleo
  - Gannet
- Water
  -
  -
  -

```
\visible< ?(Octopus.1) - > {
  \only < ?(Starfish.1) - > {\transparent{0.3}}
  % code to display the Octopus...
}
```

- `?(Octopus.1)` is replaced by `5` and `?(Starfish.1)` by `6`
- The octopus is displayed from slide 5 and transparent from slide 6

Beanoves manual

Defining named overlay sets

```
\Beanoves{
   ⟨name_k⟩ = ⟨start_k⟩ :: ⟨length_k⟩,
  ...
}
```

Range starts and lengths are arithmetical expression involving raw integers as well as next ***named overlay references***.

Reference	$\longleftrightarrow$	Integer value
⟨$name_k$⟩.1	$\longleftrightarrow$	⟨$start_k$⟩
⟨$name_k$⟩.2	$\longleftrightarrow$	⟨$start_k$⟩+ 1
⟨$name_k$⟩.⟨i⟩	$\longleftrightarrow$	⟨$start_k$⟩+⟨i⟩- 1
⟨$name_k$⟩.length	$\longleftrightarrow$	⟨$length_k$⟩
⟨$name_k$⟩.next	$\longleftrightarrow$	⟨$start_k$⟩+⟨$length_k$⟩
⟨$name_k$⟩.last	$\longleftrightarrow$	⟨$start_k$⟩+⟨$length_k$⟩- 1
⟨$name_k$⟩.previous	$\longleftrightarrow$	⟨$start_k$⟩-1

Beanoves manual

Defining named overlay sets

```
\Beanoves{
   ⟨name_k⟩ = ⟨start_k⟩ :: ⟨length_k⟩,
  ...
}
```

Range starts and lengths are arithmetical expression involving raw integers as well as next ***named overlay references***.

Reference	$\longleftrightarrow$	Integer value
$\langle name_k \rangle$.1	$\longleftrightarrow$	$\langle start_k \rangle$
$\langle name_k \rangle$.2	$\longleftrightarrow$	$\langle start_k \rangle$+ 1
$\langle name_k \rangle$.$\langle i \rangle$	$\longleftrightarrow$	$\langle start_k \rangle$+$\langle i$
$\langle name_k \rangle$.length	$\longleftrightarrow$	$\langle length_k \rangle$
$\langle name_k \rangle$.next	$\longleftrightarrow$	$\langle start_k \rangle$+$\langle l$
$\langle name_k \rangle$.last	$\longleftrightarrow$	$\langle start_k \rangle$+$\langle l$
$\langle name_k \rangle$.previous	$\longleftrightarrow$	$\langle start_k \rangle$−1

Revisit the example...

Beanoves manual

Defining named overlay sets

```
\Beanoves{
  Air   = 1         :: 2,
  Water = Air.next :: 3,
}
```

Range starts and lengths are arithmetical expression involving raw integers as well as next ***named overlay references***.

Reference	$\longleftrightarrow$	Integer value
`Air.1`	$\longleftrightarrow$	1
`Air.2`	$\longleftrightarrow$	2
`Air.`$\langle i \rangle$	$\longleftrightarrow$	$\langle i \rangle$
`Air.length`	$\longleftrightarrow$	2
`Air.next`	$\longleftrightarrow$	3
`Air.last`	$\longleftrightarrow$	2
`Air.previous`	$\longleftrightarrow$	0

Revisit the example...

Beanoves manual

Defining named overlay sets

```
\Beanoves{
  Air   = 1         :: 2,
  Water = Air.next :: 3,
}
```

Range starts and lengths are arithmetical expression involving raw integers as well as next ***named overlay references***.

Reference	$\longleftrightarrow$	Integer value
`Water.1`	$\longleftrightarrow$	3
`Water.2`	$\longleftrightarrow$	4
`Water.`$\langle i \rangle$	$\longleftrightarrow$	$\langle i \rangle$+ 2
`Water.length`	$\longleftrightarrow$	3
`Water.next`	$\longleftrightarrow$	6
`Water.last`	$\longleftrightarrow$	5
`Water.previous`	$\longleftrightarrow$	2

Revisit the example...

Beanoves manual

Defining named overlay sets

```
\Beanoves{
  Air   = 1         :: 2,
  Water = Air.next :: 3,
}
```

Range starts and lengths are arithmetical expression involving raw integers as well as next ***named overlay references***.

Reference	$\longleftrightarrow$	Integer value
`Water.1`	$\longleftrightarrow$	`3`
`Water.2`	$\longleftrightarrow$	`4`
`Water.`$\langle i \rangle$	$\longleftrightarrow$	$\langle i \rangle$`+ 2`
`Water.length`	$\longleftrightarrow$	`3`
`Water.next`	$\longleftrightarrow$	`6`
`Water.last`	$\longleftrightarrow$	`5`
`Water.previous`	$\longleftrightarrow$	`2`

Revisit the example...

```
Water.1 = Air.next
Air.last = Water.0
```

Beanoves manual

Defining named overlay sets

```
\Beanoves{
  Air   = 1         :: 2,
  Water = Air.next :: 3,
}
```

Range starts and lengths are arithmetical expression involving raw integers as well as next ***named overlay references***.

Reference	$\longleftrightarrow$	Integer value
`Water.1`	$\longleftrightarrow$	3
`Water.2`	$\longleftrightarrow$	4
`Water.`$\langle i \rangle$	$\longleftrightarrow$	$\langle i \rangle$+ 2
`Water.length`	$\longleftrightarrow$	3
`Water.next`	$\longleftrightarrow$	6
`Water.last`	$\longleftrightarrow$	5
`Water.previous`	$\longleftrightarrow$	2

If the duration of the Air section happens to change,

Beanoves manual

Defining named overlay sets

```
\Beanoves{
  Air   = 1          :: 3,
  Water = Air.next :: 3,
}
```

Range starts and lengths are arithmetical expression involving raw integers as well as next ***named overlay references***.

Reference	$\longleftrightarrow$	Integer value
`Water.1`	$\longleftrightarrow$	4
`Water.2`	$\longleftrightarrow$	5
`Water.`$\langle i \rangle$	$\longleftrightarrow$	$\langle i \rangle$+ 3
`Water.length`	$\longleftrightarrow$	4
`Water.next`	$\longleftrightarrow$	7
`Water.last`	$\longleftrightarrow$	6
`Water.previous`	$\longleftrightarrow$	3

If the duration of the Air section happens to change, all the integer value automatically update

Beanoves manual
Overlay specification query

- Simple specifications
- Incremental specifications
- **Specification queries**

```
\only < 4 >                {...}
\only < 1 - 3 >            {...}
```

```
\only < + >                {...}
\only < +(<i>) >           {...}
```

Beanoves manual

Overlay specification query

- Simple specifications

```
\only < 4 >          {...}
\only < 1 - 3 >      {...}
```

- Incremental specifications

```
\only < + >          {...}
\only < +(<i>) >     {...}
```

- **Specification queries**

```
\only < ?(<query>) > {...}
```

Beanoves manual
Overlay specification query

- Simple specifications

```
\only < 4 >          {...}
\only < 1 - 3 >      {...}
```

- Incremental specifications

```
\only < + >          {...}
\only < +(<i>) >     {...}
```

- **Specification queries**

```
\only < ?(<query>) > {...}
```

A query may be used in an overlay specification wherever an integer or a range can be.
\only may be replaced by any specification aware command.

Beanoves manual

Overlay specification query syntax

- Position specifications

```
?(⟨integer expression with aliases⟩)
```

- Explicit range specifications

```
?(⟨start expression⟩ :: ⟨length expression⟩)
```

Both integer expressions accept aliases.

- Logical range specifications with a ***range alias***:

$\langle name_k \rangle$`.range` $\longleftrightarrow$ $\langle name_k \rangle$`.1 - `$\langle name_k \rangle$`.last`

where "–" stands for a dash and not a minus sign.

```
?(⟨name_k⟩.range)
```

Beanoves manual

Overlay specification query syntax

- Position specifications

```
?(⟨integer expression with aliases⟩)
```

- Explicit range specifications

```
?(⟨start expression⟩ :: ⟨length expression⟩)
```

Both integer expressions accept aliases.

- Logical range specifications with a ***range alias***:

$\langle name_k \rangle$`.range` $\longleftrightarrow$ $\langle name_k \rangle$`.1 - `$\langle name_k \rangle$`.last`

where "–" stands for a dash and not a minus sign.

```
?(⟨name_k⟩.range)
```

Range queries and beamer ranges must not be combined like in **`?(Air.range)-10`**, leading to the incorrect syntax **`1-2-10`**.

Beanoves manual

Overlay specification query syntax

- Position specifications

```
?(⟨integer expression with aliases⟩)
```

- Explicit range specifications

```
?(⟨start expression⟩ :: ⟨length expression⟩)
```

Both integer expressions accept aliases.

- Lo...

⟨r...

wh...

?...

💡 The middle slide of the Air topic is

`?((Air.1+Air.last)/2).`

💡 What corresponds to next query?

`?(Water.0 :: Water.length + 2)`

Range ... be combined like in **?(Air.range)-10**, leading to the incorrect syntax **1-2-10**.

Beanoves manual

Incremental specifications

```
\begin {frame}
\Beanoves {
  ⟨name_k⟩ = ⟨start_k⟩ :: ⟨length_k⟩,
  ...
}
```

Each logical overlay range has a ***counter***, with dedicated alias and operations. Within a specification query:

- ⟨***name***$_k$⟩, with no trailing "**.**", is an alias for the ***counter***
- **++**⟨***name***$_k$⟩ stands for the ***counter*** once incremented by 1
- ⟨***name***$_k$⟩**+=**⟨***i***⟩ stands for the ***counter*** after being incremented by ⟨i⟩.
- ⟨***name***$_k$⟩**.reset** stands for the ***counter*** once reset to its initial value.

Beanoves manual

Incremental specifications

```
\begin {frame}
\Beanoves {
  ⟨name_k⟩ = ⟨start_k⟩ :: ⟨length_k⟩,
  ...
}
```

Each logical overlay range has a ***counter***, with dedicated alias and operations. Within a specification query:

- ⟨***name_k***⟩, with no trailing "**.**", is an alias for the ***counter***
- **++**⟨***name_k***⟩ stands for the ***counter*** once incremented by 1
- ⟨***name_k***⟩**+=**⟨***i***⟩ stands for the ***counter*** after being incremented by ⟨*i*⟩.
- ⟨***name_k***⟩**.reset** stands for the ***counter*** once reset to its initial value.

Beanoves manual

Incremental specifications

```
\begin {frame}
\Beanoves {
  ⟨name_k⟩ = ⟨start_k⟩ :: ⟨length_k⟩,
  ...
}
```

Each logical overlay range has a ***counter***, with dedicated alias and operations. Within a specification query:

- ⟨***name_k***⟩, with no trailing "**.**", is an alias for the ***counter***
- **++**⟨***name_k***⟩ stands for the ***counter*** once incremented by 1
- ⟨***name_k***⟩**+=**⟨***i***⟩ stands for the ***counter*** after being incremented by ⟨*i*⟩.
- ⟨***name_k***⟩**.reset** stands for the ***counter*** once reset to its initial value.

Beanoves manual

Incremental specifications

```
\begin {frame}
\Beanoves {
  ⟨name_k⟩ = ⟨start_k⟩ :: ⟨length_k⟩,
  ...
}
```

Each logical overlay range has a ***counter***, with dedicated alias and operations. Within a specification query:

- ⟨***name***$_k$⟩, with no trailing "**.**", is an alias for the ***counter***
- **++**⟨***name***$_k$⟩ stands for the ***counter*** once incremented by 1
- ⟨***name***$_k$⟩**+=**⟨***i***⟩ stands for the ***counter*** after being incremented by ⟨i⟩.
- ⟨***name***$_k$⟩**.reset** stands for the ***counter*** once reset to its initial value.

Beanoves manual

Incremental specifications

```
\begin {frame}
\Beanoves {
  ⟨name_k⟩ = ⟨start_k⟩ :: ⟨length_k⟩,
  ...
}
```

Each logical overlay range has a ***counter***, with dedicated alias and operations. Within a specification query:

- ⟨***name***$_k$⟩, with no trailing "**.**", is an alias for the ***counter***
- **++**⟨***name***$_k$⟩ stands for the ***counter*** once incremented by 1
- ⟨***name***$_k$⟩**+=**⟨***i***⟩ stands for the ***counter*** after being incremented by ⟨i⟩.
- ⟨***name***$_k$⟩**.reset** stands for the ***counter*** once reset to its initial value.

Beanoves manual

Incremental specifications

```
\begin {frame}
\Beanoves {
  ⟨name_k⟩ = ⟨start_k⟩ :: ⟨length_k⟩,
  ...
}
```

Each logical overlay range has a ***counter***, with dedicated alias and operations. Within a specification query:

- ⟨***name***$_k$⟩, with no trailing "**.**", is an alias for the ***counter***
- **++**⟨***name***$_k$⟩ stands for the ***counter*** once incremented by 1
- ⟨***name***$_k$⟩**+=**⟨***i***⟩ stands for the ***counter*** after being incremented by ⟨i⟩.
- ⟨***name***$_k$⟩**.reset** stands for the ***counter*** once reset to its initial value

⏳ Revisit the example...

Beanoves manual

Incremental specifications in practice

```
\Beanoves{
  Water = 1 :: 3,
}
\begin{frame}

\begin{itemize} [ <?(Water++)-> ]
  \item Octopus
  \item Starfish
  \item Picasso fish
\end{itemize}
\end{frame}
```

To uncover items one at a time,

Beanoves manual

Incremental specifications in practice

```
\Beanoves{
  Water = 1 :: 3,
}
\begin{frame}

\begin{itemize} [<?(Water++)->]
  \item Octopus
  \item Starfish
  \item Picasso fish
\end{itemize}
\end{frame}
```

To uncover items one at a time, we use beamer's advanced overlay specifications.

Beanoves manual

Incremental specifications in practice

- Octopus

```
\Beanoves{
  Water = 1 :: 3,
}
\begin{frame}

\begin{itemize} [ <?(Water++)-> ]
  \item Octopus
  \item Starfish
  \item Picasso fish
\end{itemize}
\end{frame}
```

Beanoves manual

Incremental specifications in practice

- Octopus
- Starfish

```
\Beanoves{
  Water = 1 :: 3,
}
\begin{frame}

\begin{itemize} [ <?(Water++)-> ]
  \item Octopus
  \item Starfish
  \item Picasso fish
\end{itemize}
\end{frame}
```

Notice how the lines of code are hilighted at the same time the items are uncovered...

Beanoves manual

Incremental specifications in practice

- Octopus
- Starfish
- Picasso fish

```
\Beanoves{
  Water = 1 :: 3,
}
\begin{frame}

\begin{itemize} [ <?(Water++)-> ]
  \item Octopus
  \item Starfish
  \item Picasso fish
\end{itemize}
\end{frame}
```

Notice how the lines of code are hilighted at the same time the items are uncovered...

Beanoves manual

Incremental specifications in practice

- Octopus
- Starfish
- Picasso fish

```
\Beanoves{
  Water = 1 :: 3,
}
\begin{frame}
\setbeamercovered{ transparent = 30 }
\begin{itemize} [ <?(Water++)> ]
  \item Octopus
  \item Starfish
  \item Picasso fish
\end{itemize}
\end{frame}
```

Managing transparency is straightforward

Beanoves manual

Incremental specifications in practice

- Octopus
- Starfish
- Picasso fish

```
\Beanoves{
  Water = 1 :: 3,
}
\begin{frame}
\setbeamercovered{ transparent = 30 }
\begin{itemize} [ <?(Water++)> ]
  \item Octopus
  \item Starfish
  \item Picasso fish
\end{itemize}
\end{frame}
```

Managing transparency is straightforward we use beamer's facilities.

Beanoves manual

Incremental specifications in practice

- Octopus
- Starfish
- Picasso fish

```
\Beanoves{
  Water = 1 :: 3,
}
\begin{frame}
\setbeamercovered{ transparent = 30 }
\begin{itemize} [ <?(Water++)> ]
  \item Octopus
  \item Starfish
  \item Picasso fish
\end{itemize}
\end{frame}
```

Once again the code is properly hilighted to illustrate the behavior of the itemized list...

Beanoves manual

Incremental specifications in practice

- Octopus
- Starfish
- Picasso fish

```
\Beanoves{
  Water = 1 :: 3,
}
\begin{frame}
\setbeamercovered{ transparent = 30 }
\begin{itemize} [ <?(Water++)> ]
  \item Octopus
  \item Starfish
  \item Picasso fish
\end{itemize}
\end{frame}
```

Once again the code is properly hilighted to illustrate the behavior of the itemized list...

Beanoves manual

Incremental specifications in practice

- Octopus
- Starfish
- Picasso fish

```
\Beanoves{
  Water = 1 :: 3,
}
\begin{frame}
\setbeamercovered{ transparent = 30 }
\begin{itemize} [ <?(Water++)> ]
  \item Octopus
  \item Starfish
  \item Picasso fish
\end{itemize}
\end{frame}
```

Once again the code is properly hilighted to illustrate the behavior of the itemized list...

Beanoves manual

Why aliases are helpful

- As soon as one leaves basic frame layouts to make presentations more attractive and efficient, then beanoves aliases should come into play.
- One can organize the slides with logical names for a better understanding: aliases and integer expressions rather than raw integers make specifications more explicit
- Adding or removing a slide from one slide range does not significantly affect the other slide ranges.