

# Beanoves demonstration manual

## Introduction

### Beamer named **overlay** specifications

- ▶ 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.

# Beanoves demonstration manual

## Beamer facts

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
\only<--+>\transparent{0.3}

```

# 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
\only<-->{\transparent{0.3}}

```

# 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
\only<--+>\transparent{0.3}

```

# 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



We had 7 slides

## Beanoves example: Uncovered items

### Chronology

Slide: 1 2 3 4 5 6 7 →



We had 7 slides

## Beanoves example: Uncovered items

### Chronology

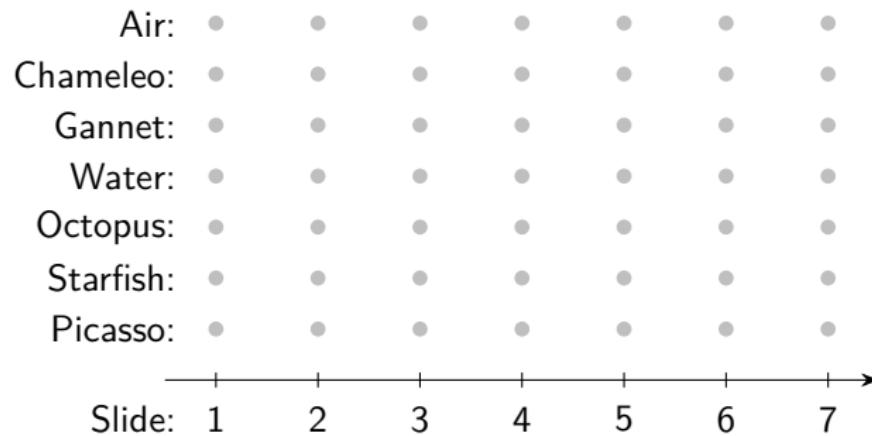
Slide: 1 2 3 4 5 6 7 →



We had 7 slides for 7 texts

# Beanoves example: Uncovered items

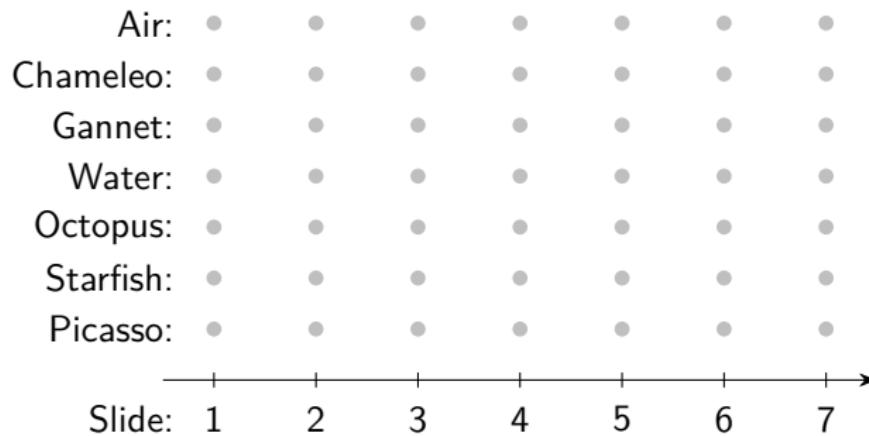
## Chronology



We had 7 slides for 7 texts

# Beanoves example: Uncovered items

## Chronology



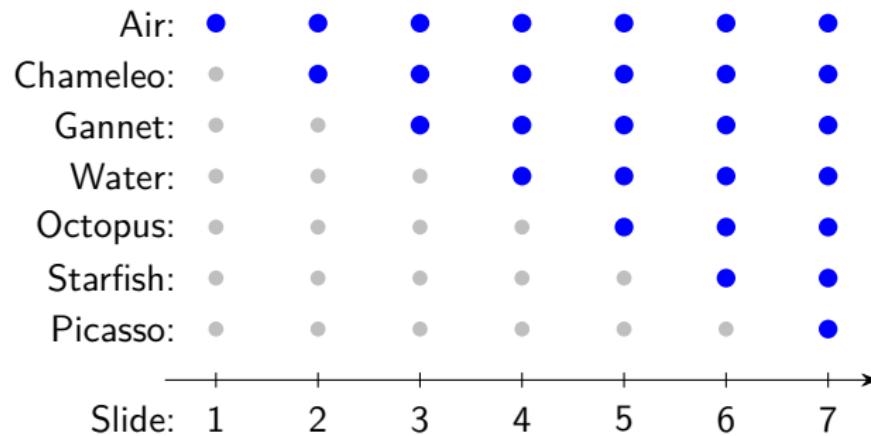
We had 7 slides for 7 texts



Blue dots for uncovered text

# Beanoves example: Uncovered items

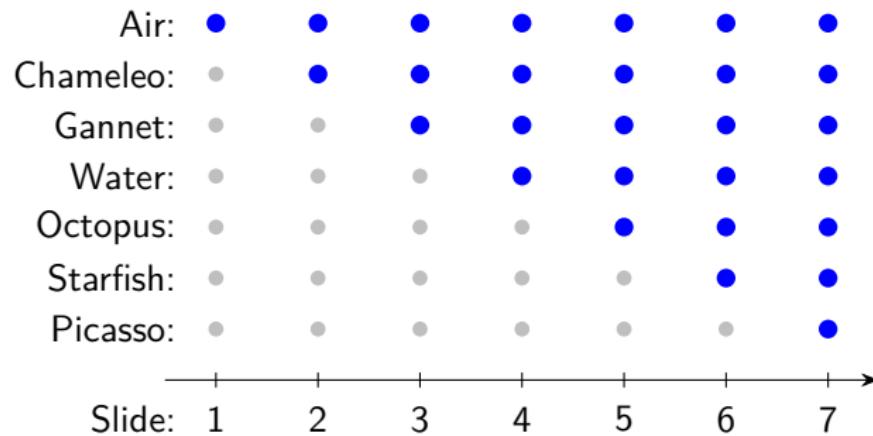
## Chronology



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

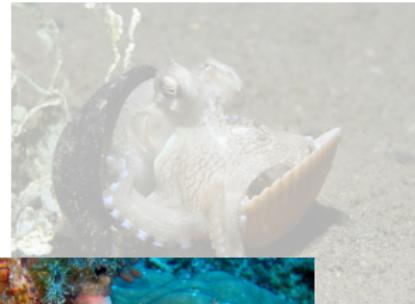
- ▶ Air
  - ▶ Chameleo
  - ▶ Gannet
- ▶ Water
  - ▶ Octopus
  - ▶ Starfish
  - ▶ Picasso fish



# Beanoves example: Uncovered items + images

Slide 7

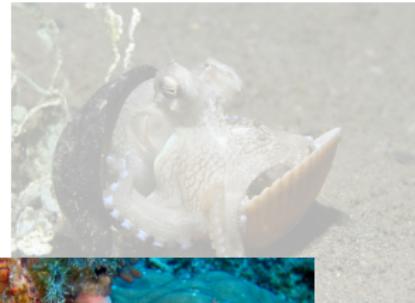
- ▶ Air
  - ▶ Chameleo
  - ▶ Gannet
- ▶ Water
  - ▶ Octopus
  - ▶ Starfish
  - ▶ Picasso fish



# Beanoves example: Uncovered items + images

Slide 8

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

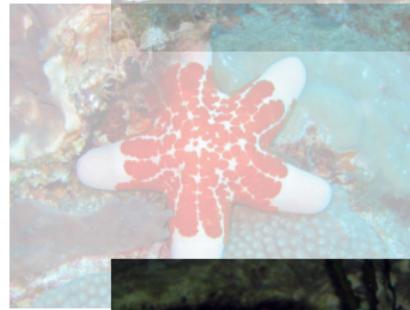
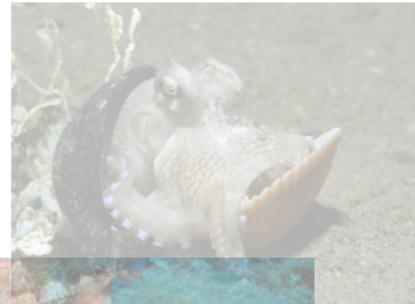


Overlap allows for larger images

# 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



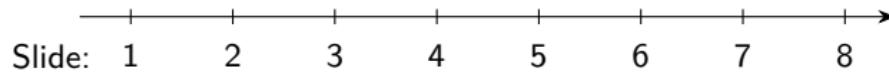
# Beanoves example: Uncovered items + images

## Chronology

- We had 8 slides

# Beanoves example: Uncovered items + images

## Chronology



- We had 8 slides

# Beanoves example: Uncovered items + images

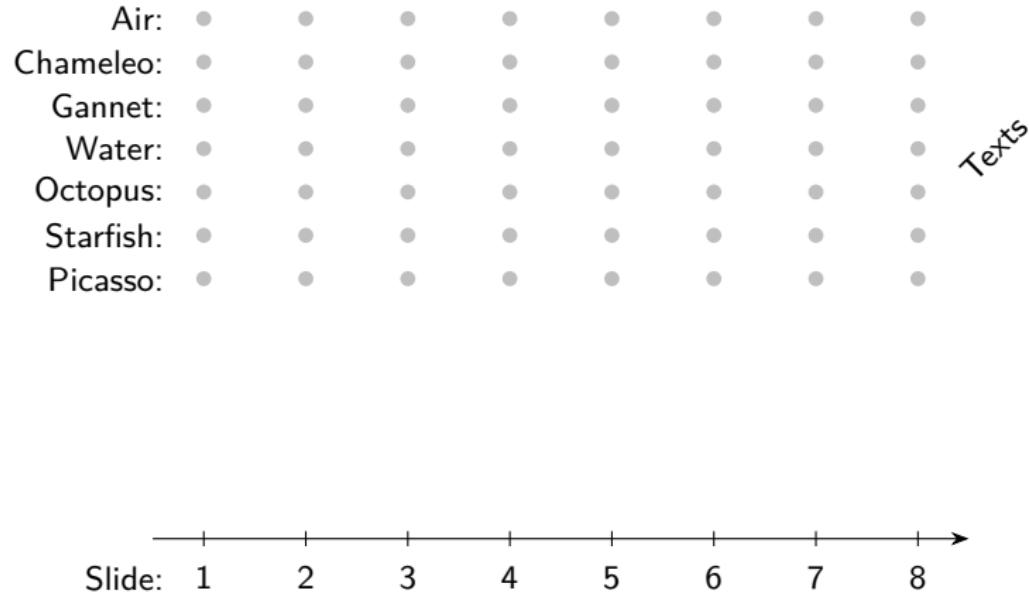
## Chronology

Slide: 1 2 3 4 5 6 7 8

- We had 8 slides for 7 texts

# Beanoves example: Uncovered items + images

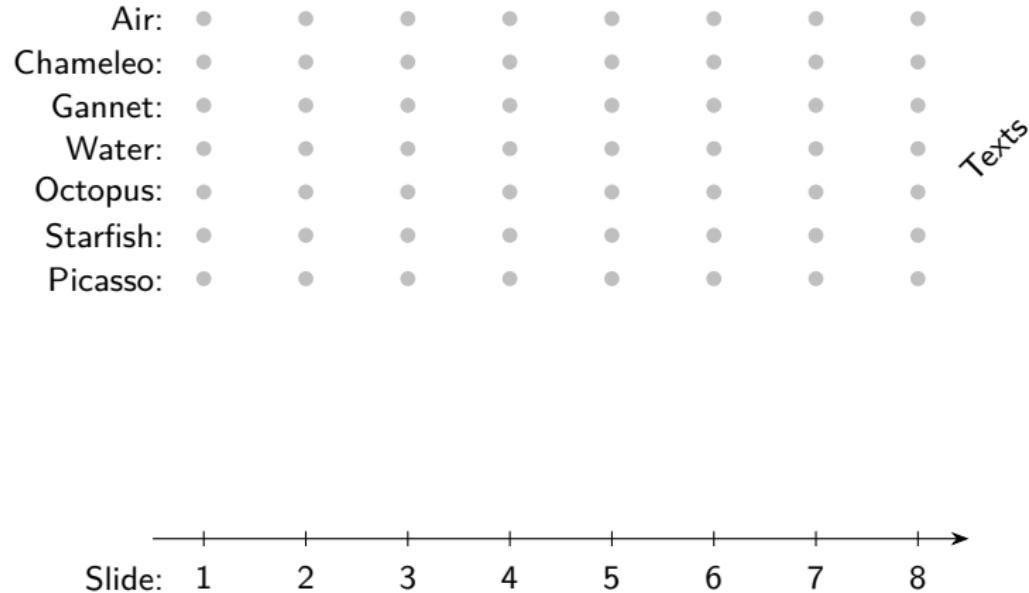
## Chronology



- We had 8 slides for 7 texts

# Beanoves example: Uncovered items + images

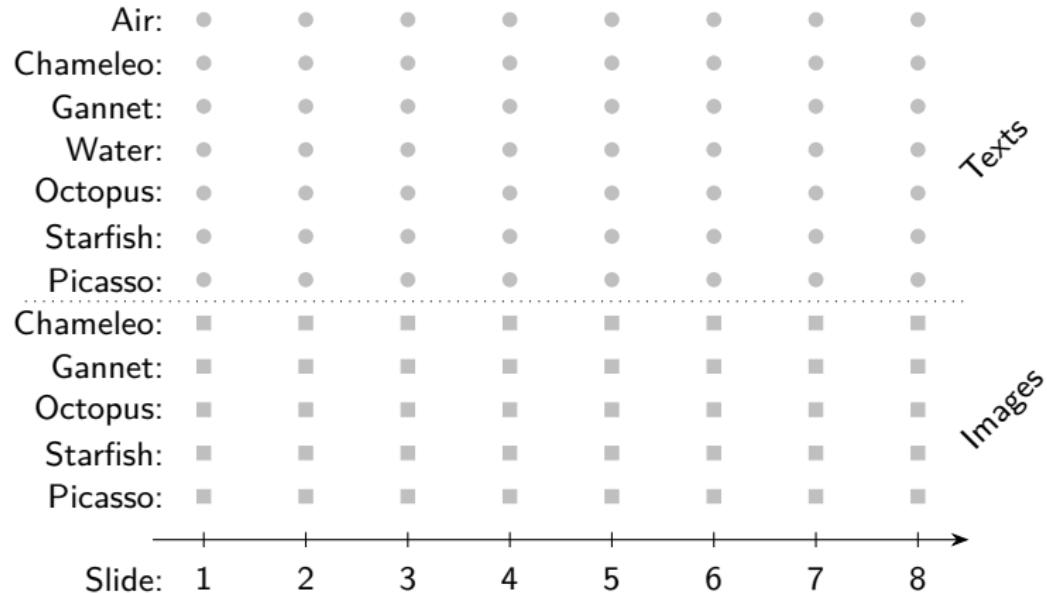
## Chronology



- We had 8 slides for 7 texts and 5 images

# Beanoves example: Uncovered items + images

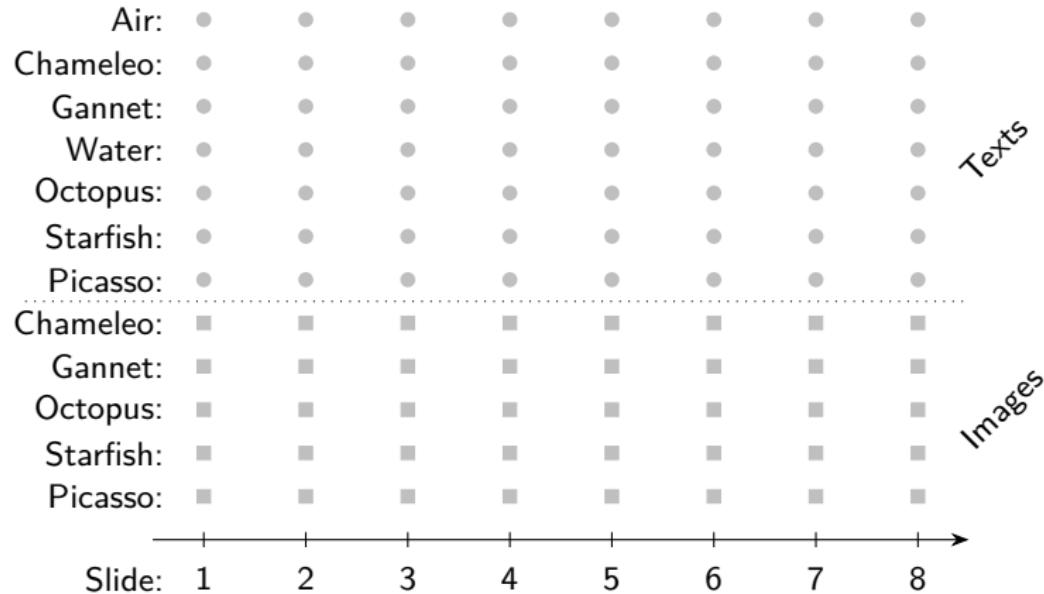
## Chronology



- We had 8 slides for 7 texts and 5 images

# Beanoves example: Uncovered items + images

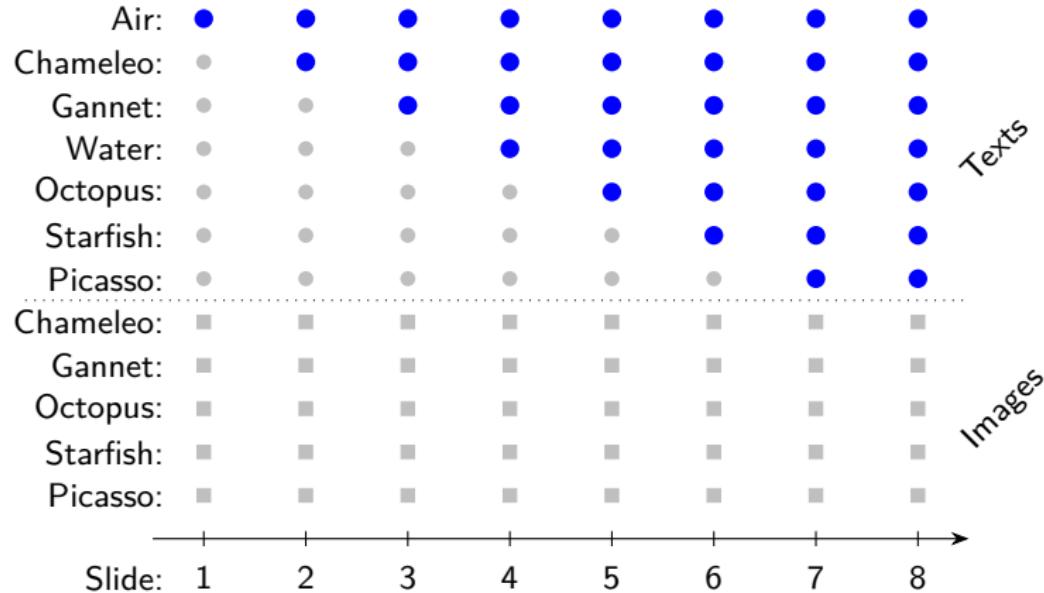
## Chronology



- Blue dots for uncovered text

# Beanoves example: Uncovered items + images

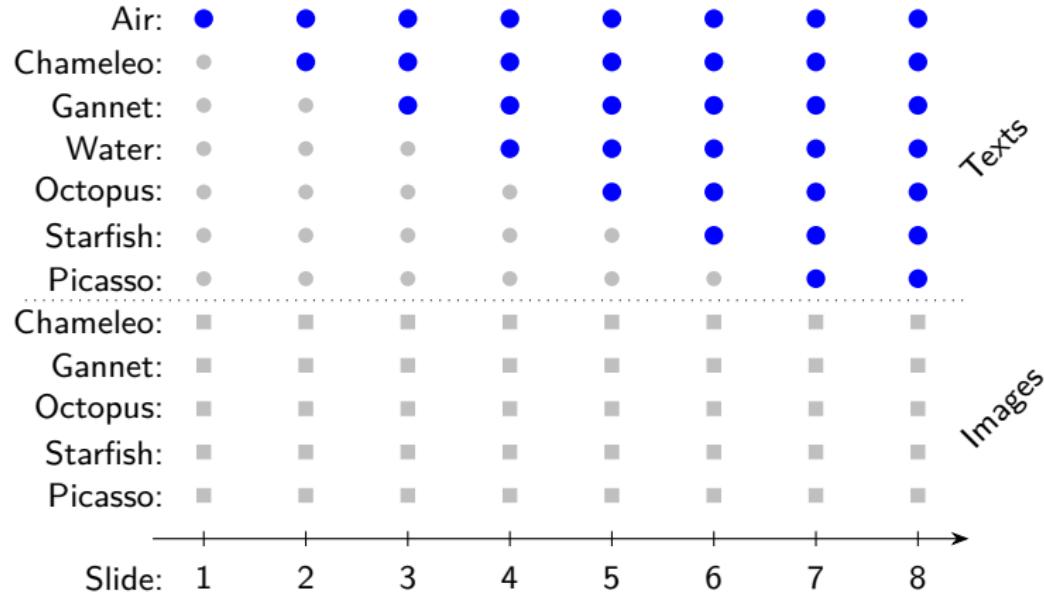
## Chronology



- Blue dots for uncovered text

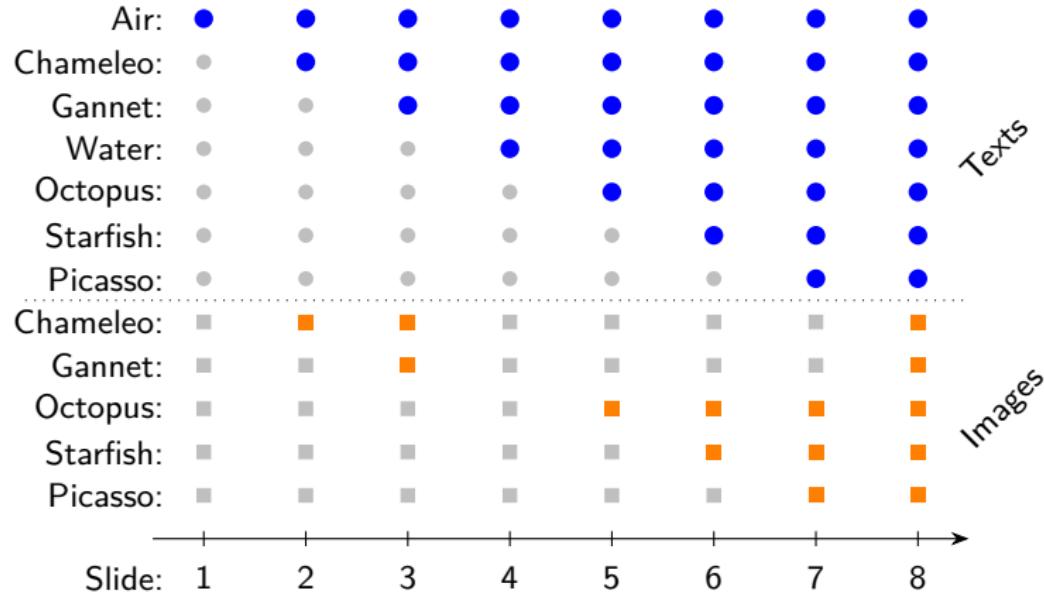
# Beanoves example: Uncovered items + images

## Chronology



# Beanoves example: Uncovered items + images

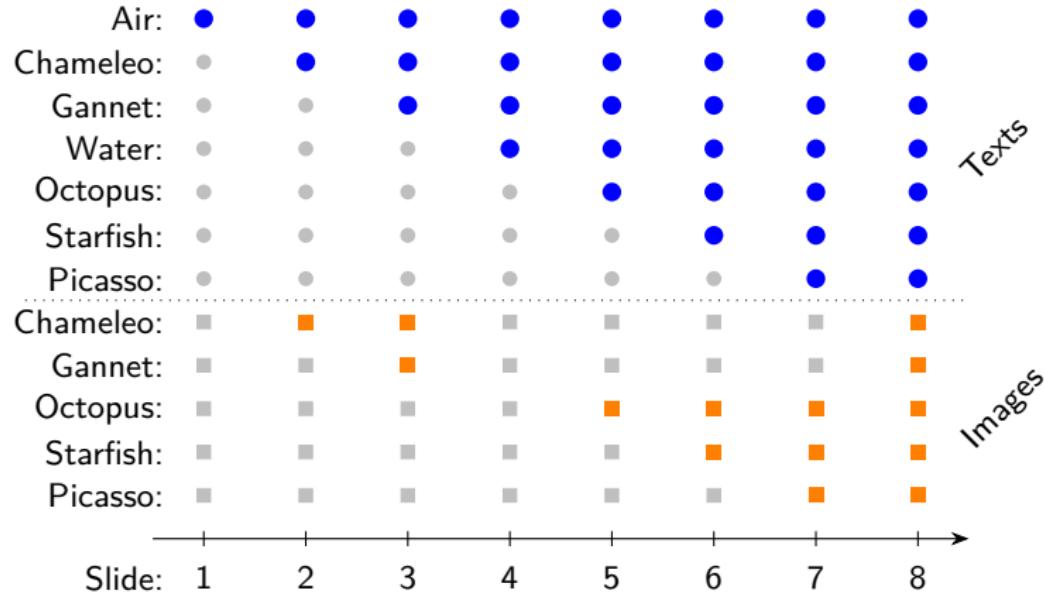
## Chronology



- orange squares for uncovered images

# Beanoves example: Uncovered items + images

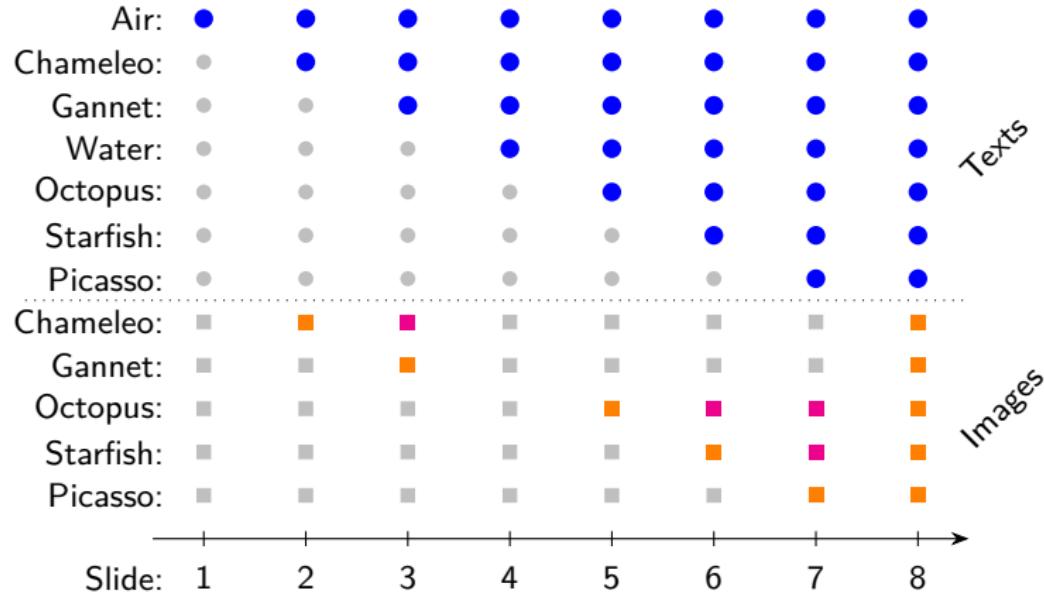
## Chronology



- magenta squares for uncovered transparent images

# Beanoves example: Uncovered items + images

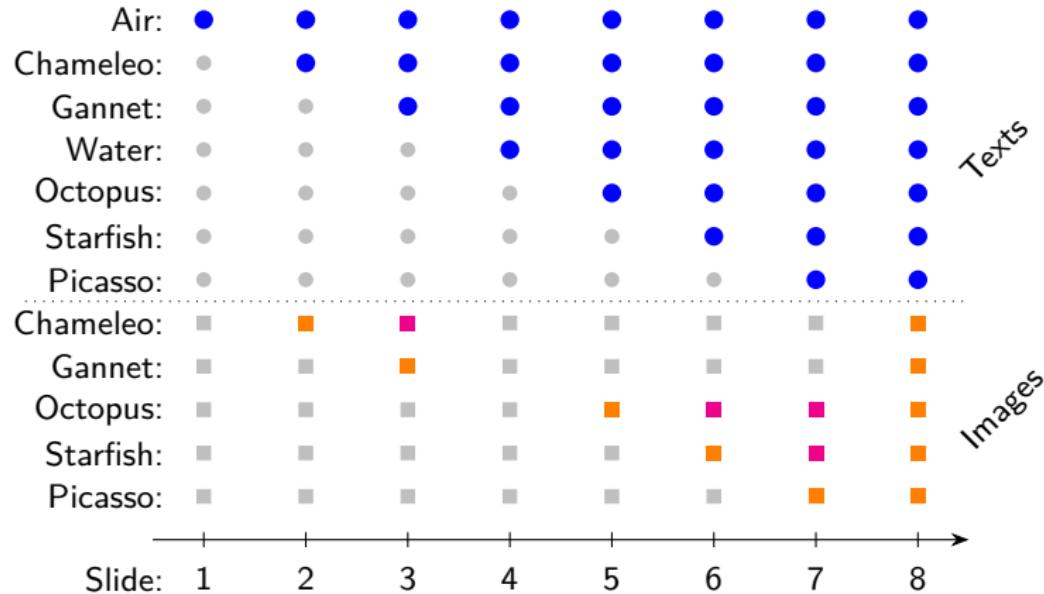
## Chronology



- magenta squares for uncovered transparent images

# Beanoves example: Uncovered items + images

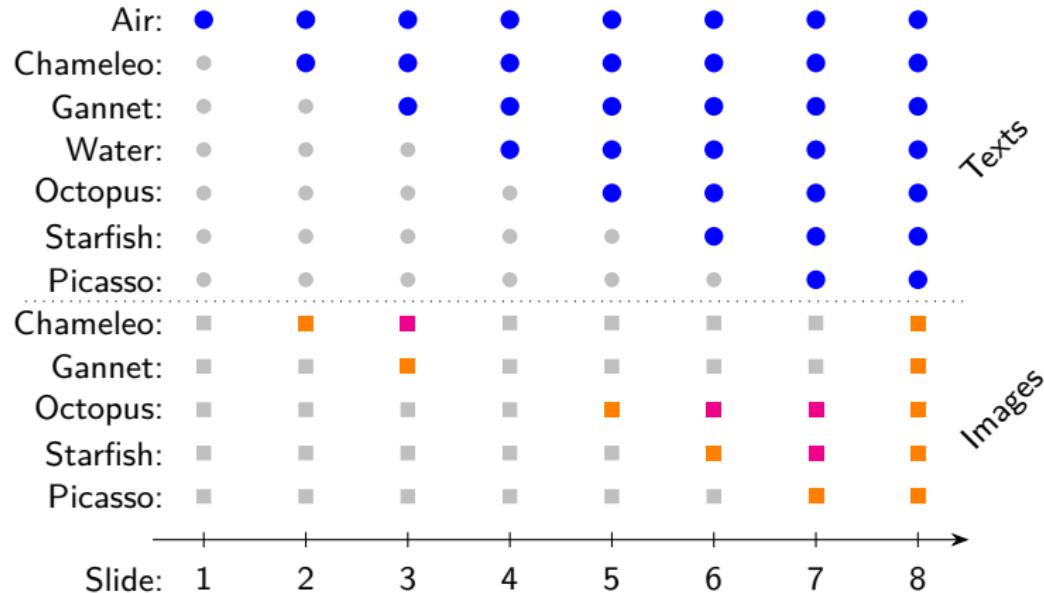
## Chronology



Let us organize all this

# Beanoves example: Uncovered items + images

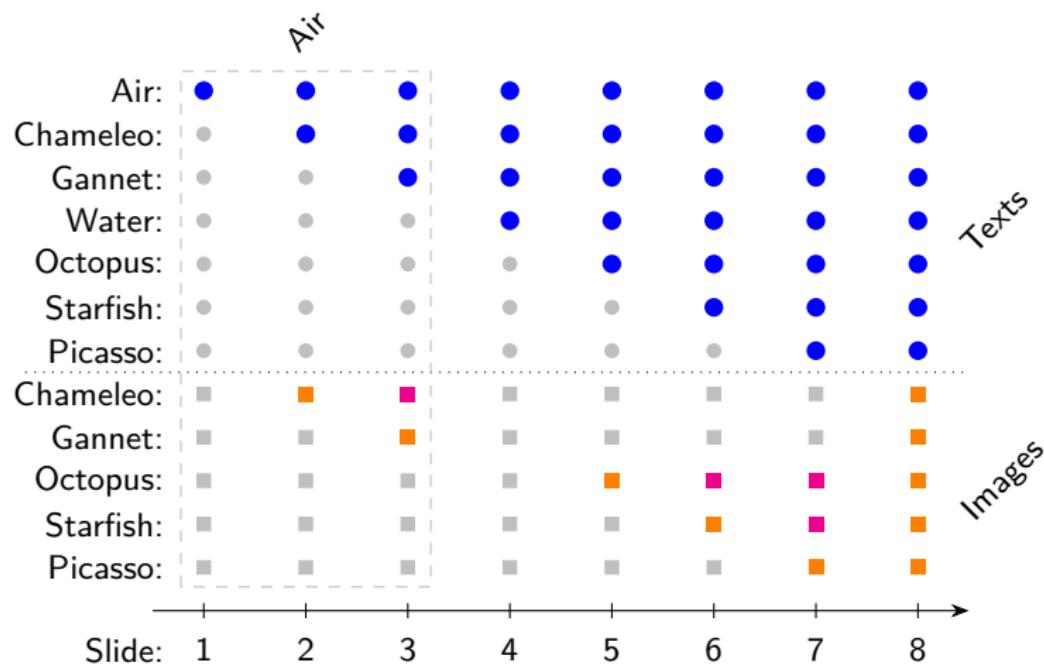
## Chronology



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

# Beanoves example: Uncovered items + images

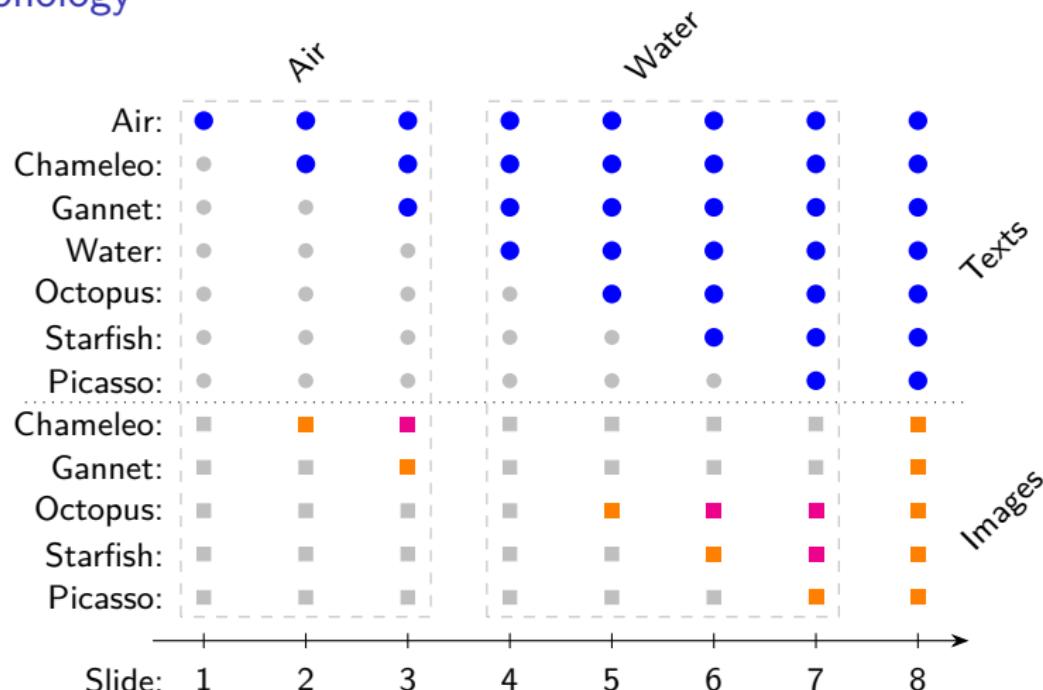
## Chronology



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

# Beanoves example: Uncovered items + images

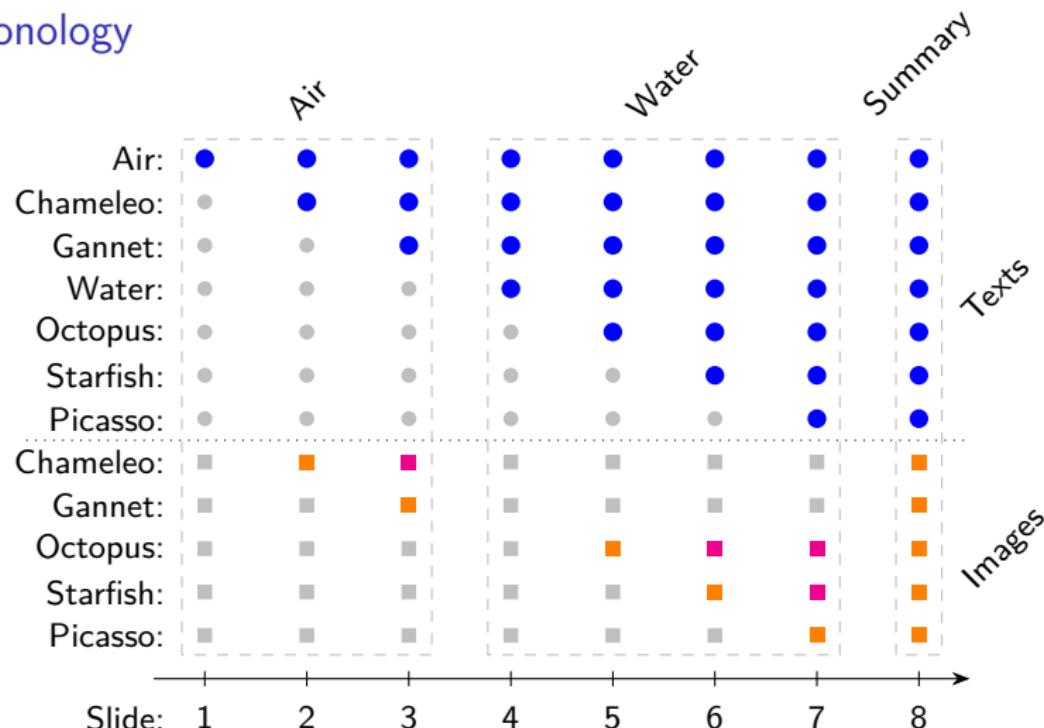
## Chronology



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

# Beanoves example: Uncovered items + images

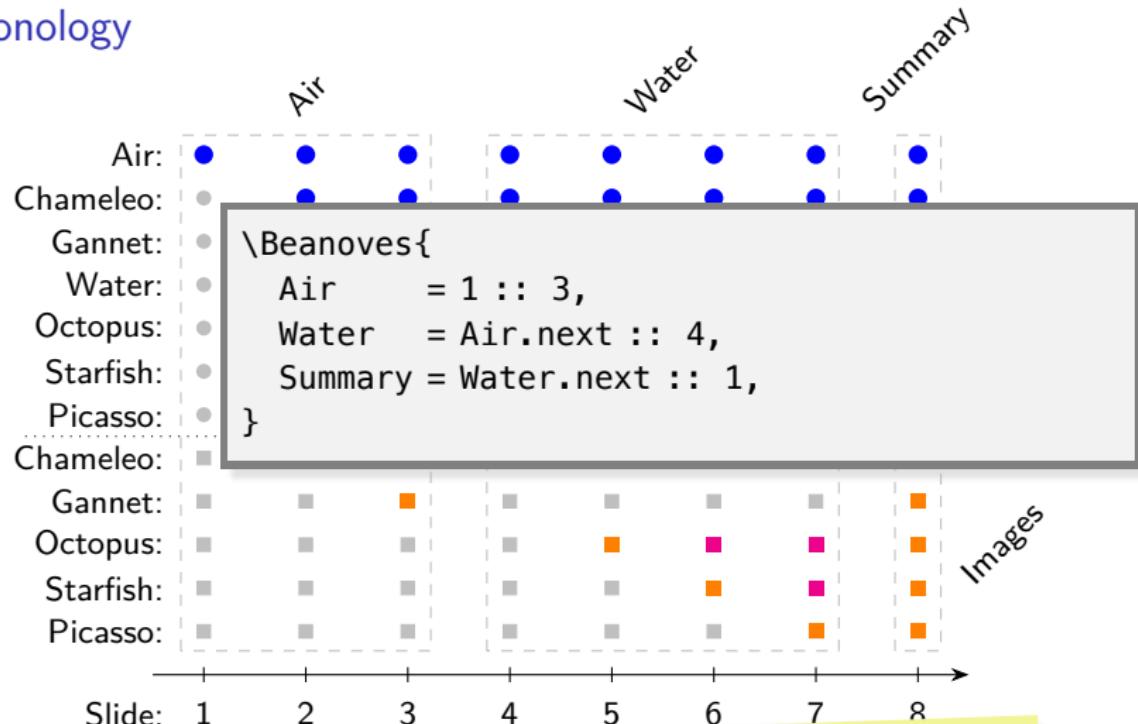
## Chronology



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

# Beanoves example: Uncovered items + images

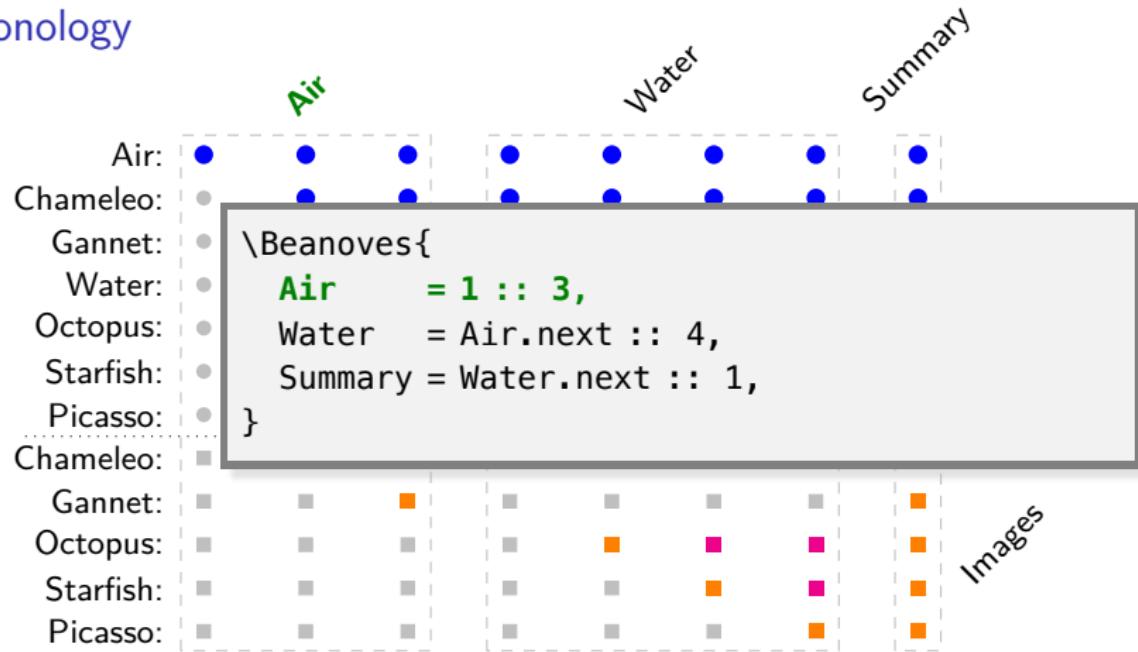
## Chronology



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

## Beanves example: Uncovered items + images

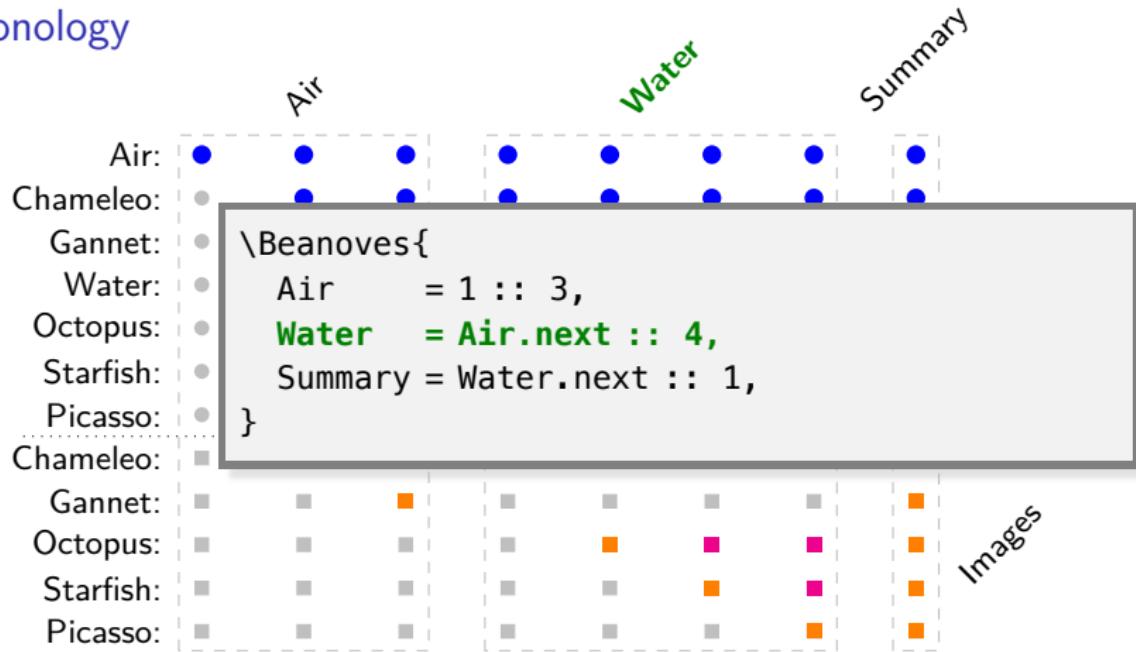
## Chronology



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

## Beanves example: Uncovered items + images

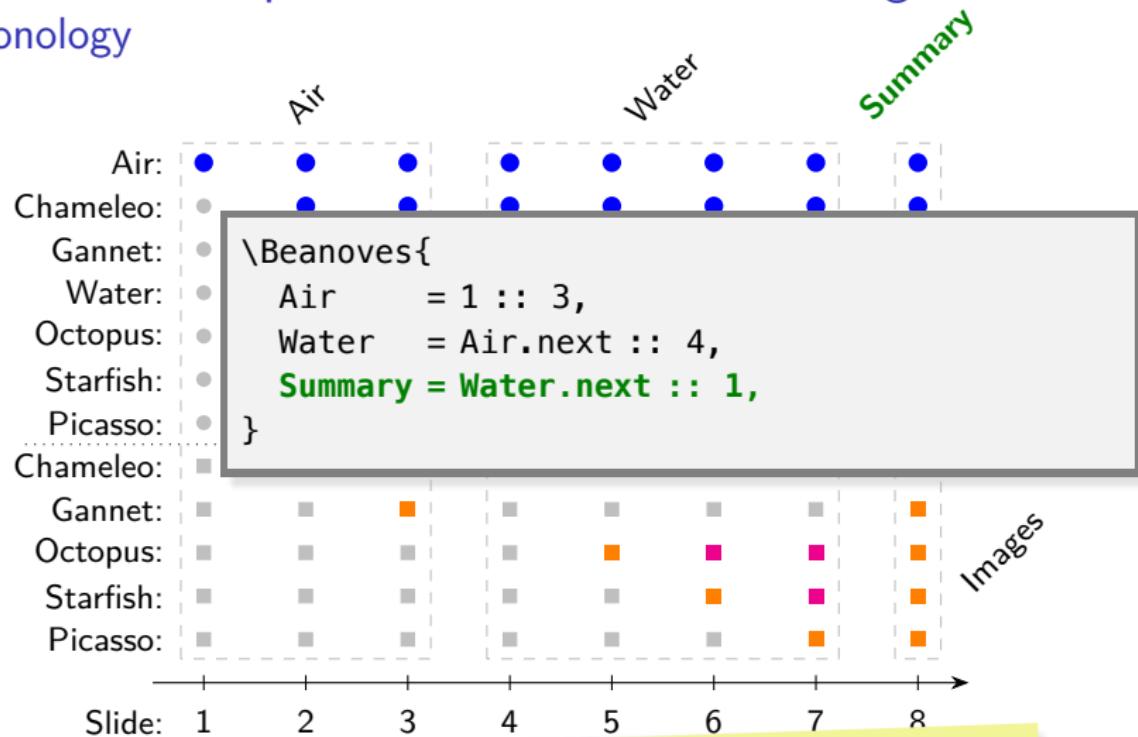
## Chronology



- 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

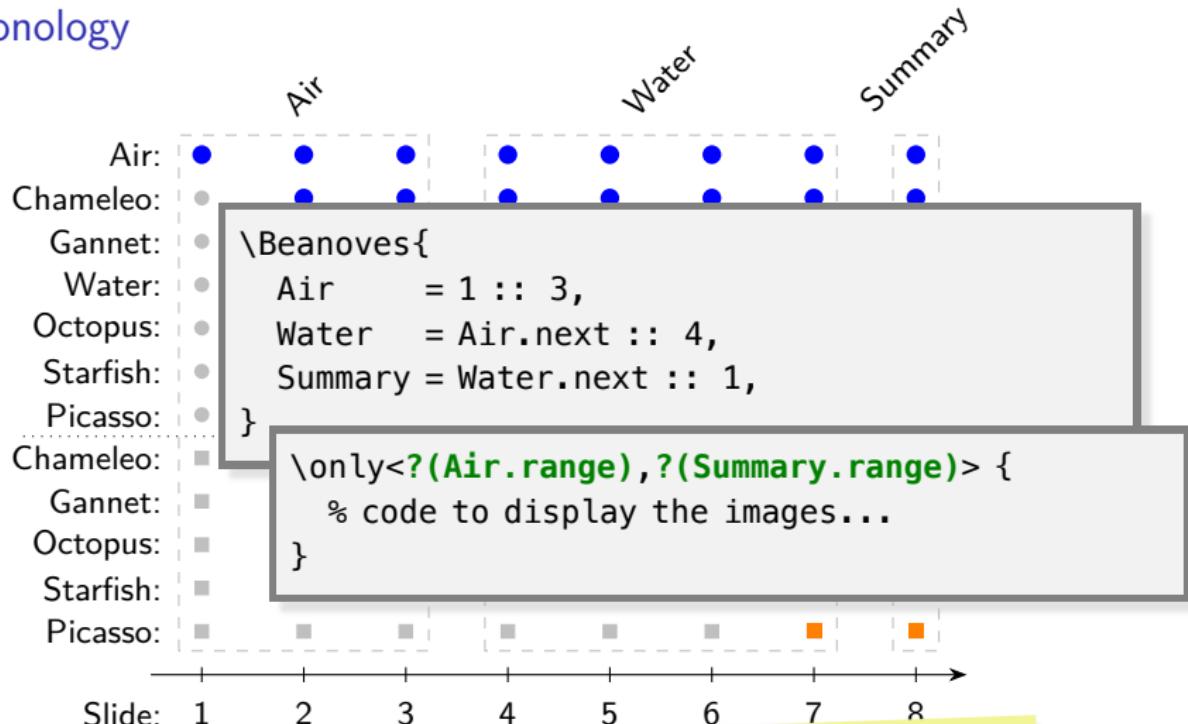
## Chronology



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

## Beanves example: Uncovered items + images

## Chronology



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

# Beanoves example: Uncovered items + images

## Chronology

	Air		Water		Summary	
Air:	●		●		●	
Chameleo:	●	●	●	●	●	●
Gannet:	●	●	●	●	●	●
Water:	●	●	●	●	●	●
Octopus:	●	●	●	●	●	●
Starfish:	●	●	●	●	●	●
Picasso:	●	●	●	●	●	●
Chameleo:	■					
Gannet:	■					
Octopus:	■					
Starfish:	■					
Picasso:	■	■	■	■	■	■

Texts

Slide: 1 2 3 4 5 6 7 8

```
\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



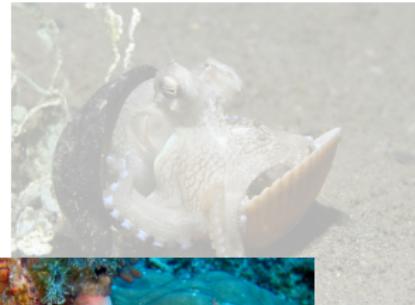
On next slide, the "Octopus" image is expected to  
be transparent



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{  
    <math_k> = <start_k> :: <length_k>,  
    ...  
}
```

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

Reference	$\leftrightarrow$	Integer value
$\langle name_k \rangle . 1$	$\leftrightarrow$	$\langle start_k \rangle$
$\langle name_k \rangle . 2$	$\leftrightarrow$	$\langle start_k \rangle + 1$
$\langle name_k \rangle . \langle i \rangle$	$\leftrightarrow$	$\langle start_k \rangle + \langle i \rangle - 1$
$\langle name_k \rangle . length$	$\leftrightarrow$	$\langle length_k \rangle$
$\langle name_k \rangle . next$	$\leftrightarrow$	$\langle start_k \rangle + \langle length_k \rangle$
$\langle name_k \rangle . last$	$\leftrightarrow$	$\langle start_k \rangle + \langle length_k \rangle - 1$
$\langle name_k \rangle . previous$	$\leftrightarrow$	$\langle start_k \rangle - 1$

# Beanoves manual

## Defining named overlay sets

```
\Beanoves{  
    <math_k> = <start_k> :: <length_k>,  
    ...  
}
```

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

Reference	$\leftrightarrow$	Integer value
$\langle name_k \rangle . 1$	$\leftrightarrow$	$\langle start_k \rangle$
$\langle name_k \rangle . 2$	$\leftrightarrow$	$\langle start_k \rangle + 1$
$\langle name_k \rangle . \langle i \rangle$	$\leftrightarrow$	$\langle start_k \rangle + \langle i \rangle - 1$
$\langle name_k \rangle . length$	$\leftrightarrow$	$\langle length_k \rangle$
$\langle name_k \rangle . next$	$\leftrightarrow$	$\langle start_k \rangle + \langle length_k \rangle$
$\langle name_k \rangle . last$	$\leftrightarrow$	$\langle start_k \rangle + \langle length_k \rangle - 1$
$\langle name_k \rangle . previous$	$\leftrightarrow$	$\langle start_k \rangle - 1$

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

Revisit the example...

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

# Beanoves manual

## Defining named overlay sets

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

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

Reference	↔	Integer value
Water.1	↔	3
Water.2	↔	4
Water. $\langle i \rangle$	↔	$\langle i \rangle + 2$
Water.length	↔	3
Water.next	↔	6
Water.last	↔	5
Water.previous	↔	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	↔	Integer value
Water.1	↔	4
Water.2	↔	5
Water.< i >	↔	$\langle i \rangle + 3$
Water.length	↔	4
Water.next	↔	7
Water.last	↔	6
Water.previous	↔	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
- ▶ Incremental specifications
- ▶ **Specification queries**

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

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

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

# Beanoves manual

## Overlay specification query

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

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

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

```
\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**:

$$\langle name_k \rangle . range \leftrightarrow \langle name_k \rangle . 1 - \langle name_k \rangle . last$$

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**:

$\langle name_k \rangle . range \leftrightarrow \langle name_k \rangle . 1 - \langle name_k \rangle . last$

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.

- ▶ Logical range specification

 The middle slide of the Air topic is  
 $\langle r \rangle$  where  $r = ?((Air.1+Air.last)/2).$

 What corresponds to next query?

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

Range expressions can be combined like in

?(Air.range)-10, leading to the incorrect syntax **1-2-10**.

# Beanoves manual

## Incremental specifications

```
\begin {frame}
\Beanoves {
  <namek> = <startk> :: <lengthk>,
  ...
}
```

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

- ▶  $\langle name_k \rangle$ , with no trailing “.”, is an alias for the **counter**
- ▶  $\text{++}\langle name_k \rangle$  stands for the **counter** once incremented by 1
- ▶  $\langle name_k \rangle \text{+=}\langle i \rangle$  stands for the **counter** after being incremented by  $\langle i \rangle$ .
- ▶  $\langle name_k \rangle.\text{reset}$  stands for the **counter** once reset to its initial value.

# Beanoves manual

## Incremental specifications

```
\begin {frame}
\Beanoves {
  <namek> = <startk> :: <lengthk>,
  ...
}
```

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

- ▶  $\langle \text{name}_k \rangle$ , with no trailing “.”, is an alias for the **counter**
- ▶  $\text{++}\langle \text{name}_k \rangle$  stands for the **counter** once incremented by 1
- ▶  $\langle \text{name}_k \rangle \text{+=}\langle i \rangle$  stands for the **counter** after being incremented by  $\langle i \rangle$ .
- ▶  $\langle \text{name}_k \rangle.\text{reset}$  stands for the **counter** once reset to its initial value.

# Beanoves manual

## Incremental specifications

```
\begin {frame}
\Beanoves {
  <namek> = <startk> :: <lengthk>,
  ...
}
```

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

- ▶  $\langle name_k \rangle$ , with no trailing “.”, is an alias for the **counter**
- ▶  $\text{++}\langle name_k \rangle$  stands for the **counter** once incremented by 1
- ▶  $\langle name_k \rangle \text{+=}\langle i \rangle$  stands for the **counter** after being incremented by  $\langle i \rangle$ .
- ▶  $\langle name_k \rangle.\text{reset}$  stands for the **counter** once reset to its initial value.

# Beanoves manual

## Incremental specifications

```
\begin {frame}
\Beanoves {
   $\langle name_k \rangle = \langle start_k \rangle :: \langle length_k \rangle,$ 
  ...
}
```

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

- ▶  $\langle name_k \rangle$ , with no trailing “.”, is an alias for the **counter**
- ▶  $\textcolor{blue}{++} \langle name_k \rangle$  stands for the **counter** once incremented by 1
- ▶  $\langle name_k \rangle \textcolor{blue}{+=} \langle i \rangle$  stands for the **counter** after being incremented by  $\langle i \rangle$ .
- ▶  $\langle name_k \rangle.\textcolor{violet}{reset}$  stands for the **counter** once reset to its initial value.

# Beanoves manual

## Incremental specifications

```
\begin {frame}
\Beanoves {
   $\langle name_k \rangle = \langle start_k \rangle :: \langle length_k \rangle,$ 
  ...
}
```

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

- ▶  $\langle name_k \rangle$ , with no trailing “.”, is an alias for the **counter**
- ▶  $\textcolor{blue}{++} \langle name_k \rangle$  stands for the **counter** once incremented by 1
- ▶  $\langle name_k \rangle \textcolor{blue}{+=} \langle i \rangle$  stands for the **counter** after being incremented by  $\langle i \rangle$ .
- ▶  $\langle name_k \rangle . \textcolor{red}{reset}$  stands for the **counter** once reset to its initial value.

# Beanoves manual

## Incremental specifications

```
\begin {frame}
\Beanoves {
   $\langle name_k \rangle = \langle start_k \rangle :: \langle length_k \rangle,$ 
  ...
}
```

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

- ▶  $\langle name_k \rangle$ , with no trailing “.”, is an alias for the **counter**
- ▶  $\text{++}\langle name_k \rangle$  stands for the **counter** once incremented by 1
- ▶  $\langle name_k \rangle \text{+=}\langle i \rangle$  stands for the **counter** after being incremented by  $\langle i \rangle$ .
- ▶  $\langle name_k \rangle.\text{reset}$  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

```
\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 highlighted 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 highlighted 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}  
\setbeamercolor{covered}{ 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}  
\setbeamercolor{covered}{ transparent = 30 }  
\begin{itemize} [ <?(Water++)> ]  
    \item Octopus  
    \item Starfish  
    \item Picasso fish  
\end{itemize}  
\end{frame}
```



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

# Beanoves manual

## Why named overlay sets are helpful

- ▶ As soon as one leaves basic frame layouts to make presentations more attractive and efficient, then beanoves named overlay sets can 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.