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 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

```
\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

Slide 2

- 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

Slide 3

- 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

Slide 4

- 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

Slide 5

- 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

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}
```

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
  ...
\end{itemize}
```

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
  ...
\end{itemize}
```

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
  ...
\end{itemize}
```

Beanoves manual

Defining logical overlay ranges

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

Beanoves manual

Defining logical overlay ranges

```
\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		Integer value
$\langle \mathit{name}_k \rangle$`.1`	$\longleftrightarrow$	$\langle \mathit{start}_k \rangle$
$\langle \mathit{name}_k \rangle$`.2`	$\longleftrightarrow$	$\langle \mathit{start}_k \rangle+$
$\langle \mathit{name}_k \rangle$`.`$\langle i \rangle$	$\longleftrightarrow$	$\langle \mathit{start}_k \rangle+\langle$
$\langle \mathit{name}_k \rangle$`.length`	$\longleftrightarrow$	$\langle \mathit{length}_k \rangle$
$\langle \mathit{name}_k \rangle$`.next`	$\longleftrightarrow$	$\langle \mathit{start}_k \rangle+\langle$
$\langle \mathit{name}_k \rangle$`.last`	$\longleftrightarrow$	$\langle \mathit{start}_k \rangle+\langle$
$\langle \mathit{name}_k \rangle$`.previous`	$\longleftrightarrow$	$\langle \mathit{start}_k \rangle-1$

Revisit the example...

Beanoves manual

Defining logical overlay ranges

```
\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 logical overlay ranges

```
\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 logical overlay ranges

```
\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 logical overlay ranges

```
\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 logical overlay ranges

```
\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 apdate

Beanoves manual

Overlay specification query

- Simple specifications

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

- Incremental specifications

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

- Specification queries

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***:

$\langle \mathit{name}_k \rangle$`.range` $\longleftrightarrow$ $\langle \mathit{name}_k \rangle$`.1` – $\langle \mathit{name}_k \rangle$`.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***:

  $\langle \mathit{name}_k \rangle$`.range` $\longleftrightarrow$ $\langle \mathit{name}_k \rangle$`.1 – `$\langle \mathit{name}_k \rangle$`.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 ra

⟨n

wh

```
?
```

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

💡 The middle slide of the Air topic is

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

💡 What corresponds to next query?

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

Beanoves manual

Incremental specifications

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

Each logical overlay range has a current slide which number is ***cursor***, with dedicated alias and operations. Within a specification query:

- ⟨***name***$_k$⟩, with not following "`.`", is an alias for the ***cursor***
- `++`⟨***name***$_k$⟩ stands for the ***cursor*** once incremented by 1
- ⟨***name***$_k$⟩`+=`⟨***i***⟩ stands for the ***cursor*** once incremented by ⟨*i*⟩.
- ⟨***name***$_k$⟩`.reset` stands for the ***cursor*** once reset.

Beanoves manual
Incremental specifications

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

Each logical overlay range has a current slide which number is ***cursor***, with dedicated alias and operations. Within a specification query:

- ⟨***name***$_k$⟩, with not following "`.`", is an alias for the ***cursor***
- `++`⟨***name***$_k$⟩ stands for the ***cursor*** once incremented by 1
- ⟨***name***$_k$⟩`+=`⟨***i***⟩ stands for the ***cursor*** once incremented by ⟨i⟩.
- ⟨***name***$_k$⟩`.reset` stands for the ***cursor*** once reset.

Beanoves manual

Incremental specifications

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

Each logical overlay range has a current slide which number is ***cursor***, with dedicated alias and operations. Within a specification query:

- ⟨***name_k***⟩, with not following "`.`", is an alias for the ***cursor***
- `++`⟨***name_k***⟩ stands for the ***cursor*** once incremented by 1
- ⟨***name_k***⟩`+=`⟨***i***⟩ stands for the ***cursor*** once incremented by ⟨i⟩.
- ⟨***name_k***⟩`.reset` stands for the ***cursor*** once reset.

Beanoves manual

Incremental specifications

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

Each logical overlay range has a current slide which number is ***cursor***, with dedicated alias and operations. Within a specification query:

- **⟨*name*$_k$⟩**, with not following "`.`", is an alias for the ***cursor***
- **++⟨*name*$_k$⟩** stands for the ***cursor*** once incremented by 1
- **⟨*name*$_k$⟩+=⟨*i*⟩** stands for the ***cursor*** once incremented by ⟨*i*⟩.
- **⟨*name*$_k$⟩`.reset`** stands for the ***cursor*** once reset.

Beanoves manual

Incremental specifications

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

Each logical overlay range has a current slide which number is ***cursor***, with dedicated alias and operations. Within a specification query:

- **⟨*name*$_k$⟩**, with not following "`.`", is an alias for the ***cursor***
- **++⟨*name*$_k$⟩** stands for the ***cursor*** once incremented by 1
- **⟨*name*$_k$⟩+=⟨*i*⟩** stands for the ***cursor*** once incremented by ⟨*i*⟩.
- **⟨*name*$_k$⟩.reset** stands for the ***cursor*** once reset.

Beanoves manual

Incremental specifications

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

Each logical overlay range has a current slide which number is ***cursor***, with dedicated alias and operations. Within a specification query:

- $\langle \mathit{name}_k \rangle$, with not following "`.`", is an alias for the ***cursor***
- `++`$\langle \mathit{name}_k \rangle$ stands for the ***cursor*** once incremented by 1
- $\langle \mathit{name}_k \rangle$`+=`$\langle i \rangle$ stands for the ***cursor*** once incremented by $\langle i \rangle$.
- $\langle \mathit{name}_k \rangle$`.reset` stands for the [illegible] set.

⌛ Revisit the example...

Beanoves manual

Why aliases are helpful

- As soon as one leaves basic frame layouts to make presentations more attractive and efficient, then bealover 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.