



Nelsie

Framework for Creating Slides

Ada Böhm
ada@kreatrix.org

Hello world example

```
from nelsie import SlideDeck

deck = SlideDeck()

@deck.slide()
def hello_world(slide):
    slide.text("Hello world!")

deck.render("slides.pdf")
```

Nelsie supports ...

Nelsie supports ...
... fragment ...

Nelsie supports ...
... fragment ...
... revealing.

Simple mechanism for complex slide changes



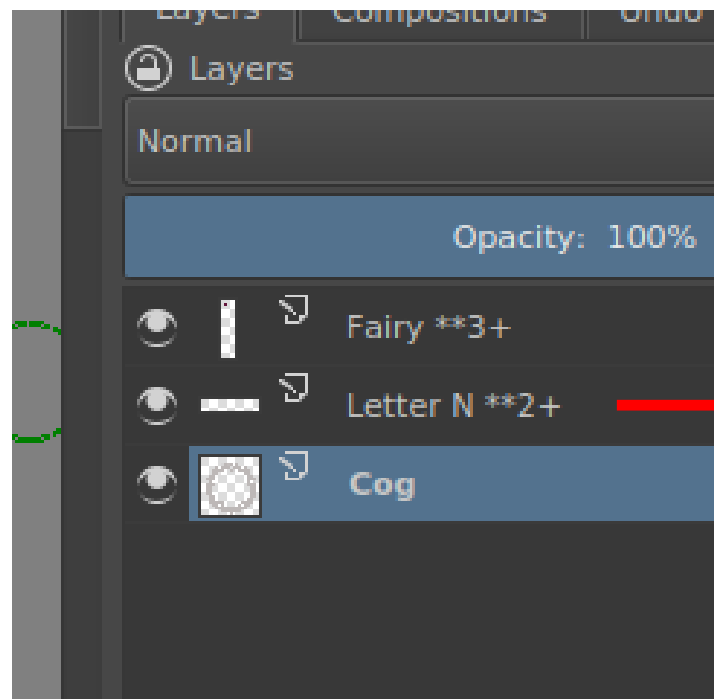
Simple mechanism for complex slide changes



Simple mechanism for complex slide changes

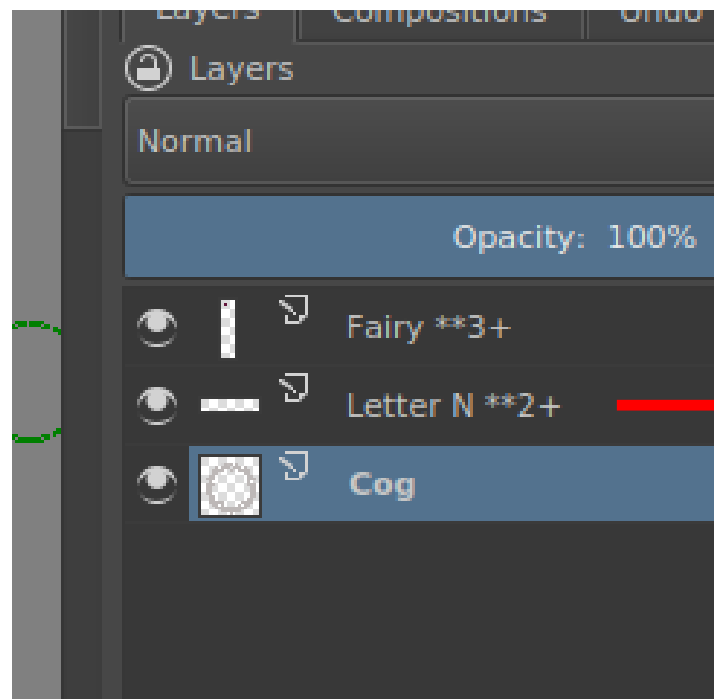


Name-driven revealing layers in SVG and ORA images



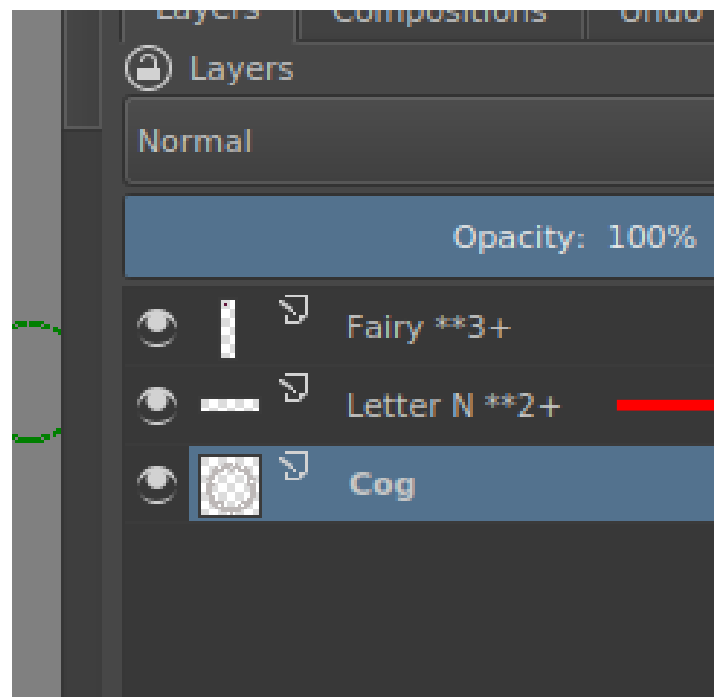
Layer is shown from the second step

Name-driven revealing layers in SVG and ORA images



Layer is shown from the second step

Name-driven revealing layers in SVG and ORA images



Layer is shown from the second step

Headers & Titles

Syntax highlighting

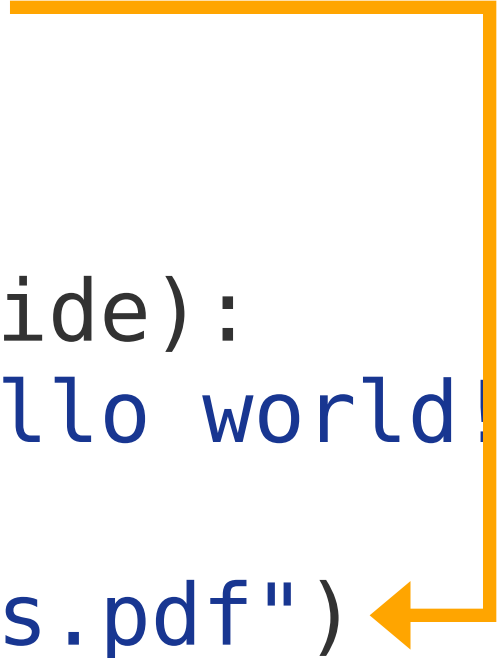
```
# This is comment  
from nelsie import SlideDeck  
  
deck = SlideDeck()  
  
@deck.slide()  
def hello_world(slide):  
    slide.text("Hello world!")  
  
deck.render("slides.pdf")
```

Line highlighting

```
# This is comment  
from nelsie import SlideDeck  
  
deck = SlideDeck()  
  
@deck.slide()  
def hello_world(slide):  
    slide.text("Hello world!")  
  
deck.render("slides.pdf")
```

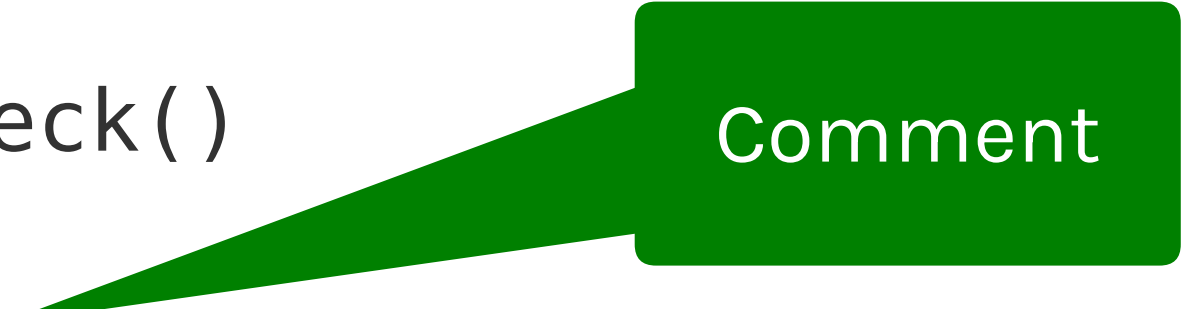
Pointing into text

```
# This is comment  
from nelsie import SlideDeck  
  
deck = SlideDeck()  
  
@deck.slide()  
def hello_world(slide):  
    slide.text("Hello world!")  
  
deck.render("slides.pdf")
```



Pointing into text

```
# This is comment  
from nelsie import SlideDeck  
  
deck = SlideDeck()  
  
@deck.slide()  
def hello_world(slide):  
    slide.text("Hello world!")  
  
deck.render("slides.pdf")
```



Comment

Pointing into text

```
# This is comment  
from nelsie import SlideDeck  
  
deck = SlideDeck()  
  
@deck.slide()  
def hello_world(slide):  
    slide.text("Hello world!")  
deck.render("slides.pdf")
```



Own styles in syntax highlight

```
# This is comment  
from nelsie import SlideDeck  
  
deck = SlideDeck()  
  
@deck.slide()  
def hello_world(slide):  
    slide.text("Hello world!")  
  
deck.render("slides.pdf")
```

Line hiding/showing

Definition (step definition after @)

```
@deck.slide()
def line_demo(slide):
    slide.code("""
def compute_something(x): @1
def compute_something(x, y): @2+
    print("Computing...") @e; 3+
    return x * y @e; 4+
""")
```

Result

```
def compute_something(x):
```

Line hiding/showing

Definition (step definition after @)

```
@deck.slide()
def line_demo(slide):
    slide.code("""
def compute_something(x): @1
def compute_something(x, y): @2+
    print("Computing...") @e; 3+
    return x * y @e; 4+
""")
```

Result

```
def compute_something(x, y):
```

Line hiding/showing

Definition (step definition after @)

```
@deck.slide()
def line_demo(slide):
    slide.code("""
def compute_something(x): @1
def compute_something(x, y): @2+
    print("Computing...") @e; 3+
    return x * y @e; 4+
""")
```

Result

```
def compute_something(x, y):
    print("Computing...")
```

Line hiding/showing

Definition (step definition after @)

```
@deck.slide()
def line_demo(slide):
    slide.code("""
def compute_something(x): @1
def compute_something(x, y): @2+
    print("Computing...") @e; 3+
    return x * y @e; 4+
""")
```

Result

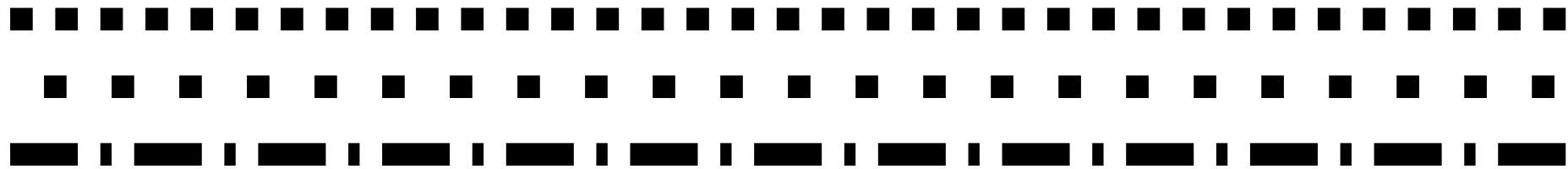
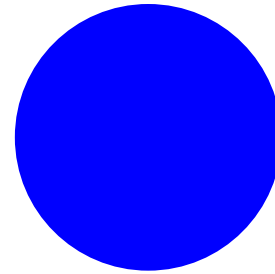
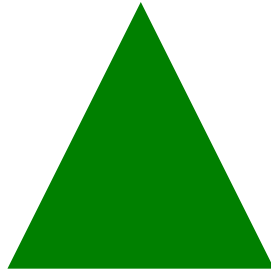
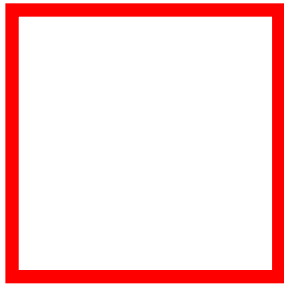
```
def compute_something(x, y):
    print("Computing...")
    return x * y
```

Console demo

```
~/nelsie/example/bigdemo$ ls  
bigdemo.py  imgs  karla_font
```

```
~/nelsie/example/bigdemo$ python3 bigdemo.py
```

Shapes



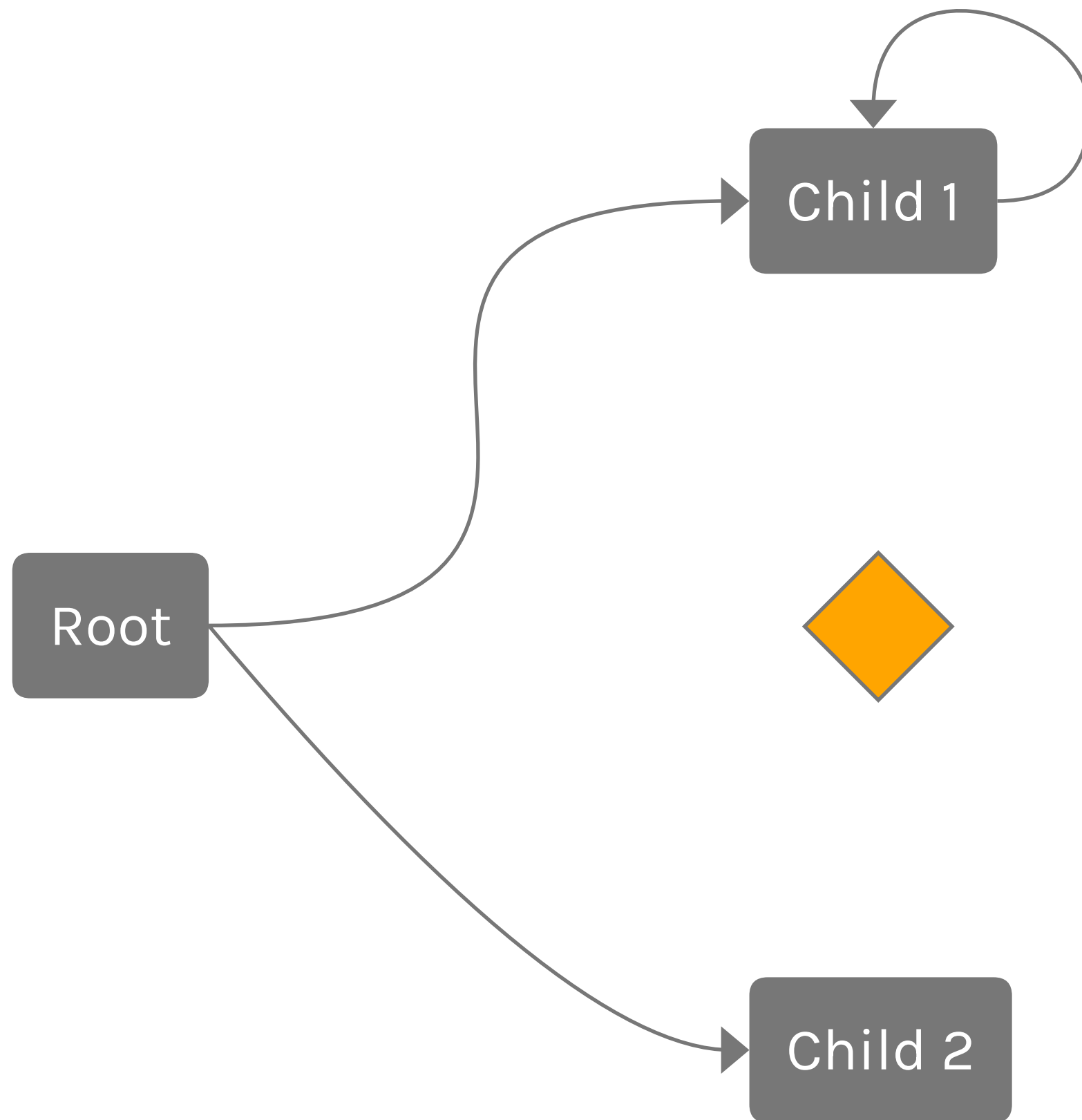
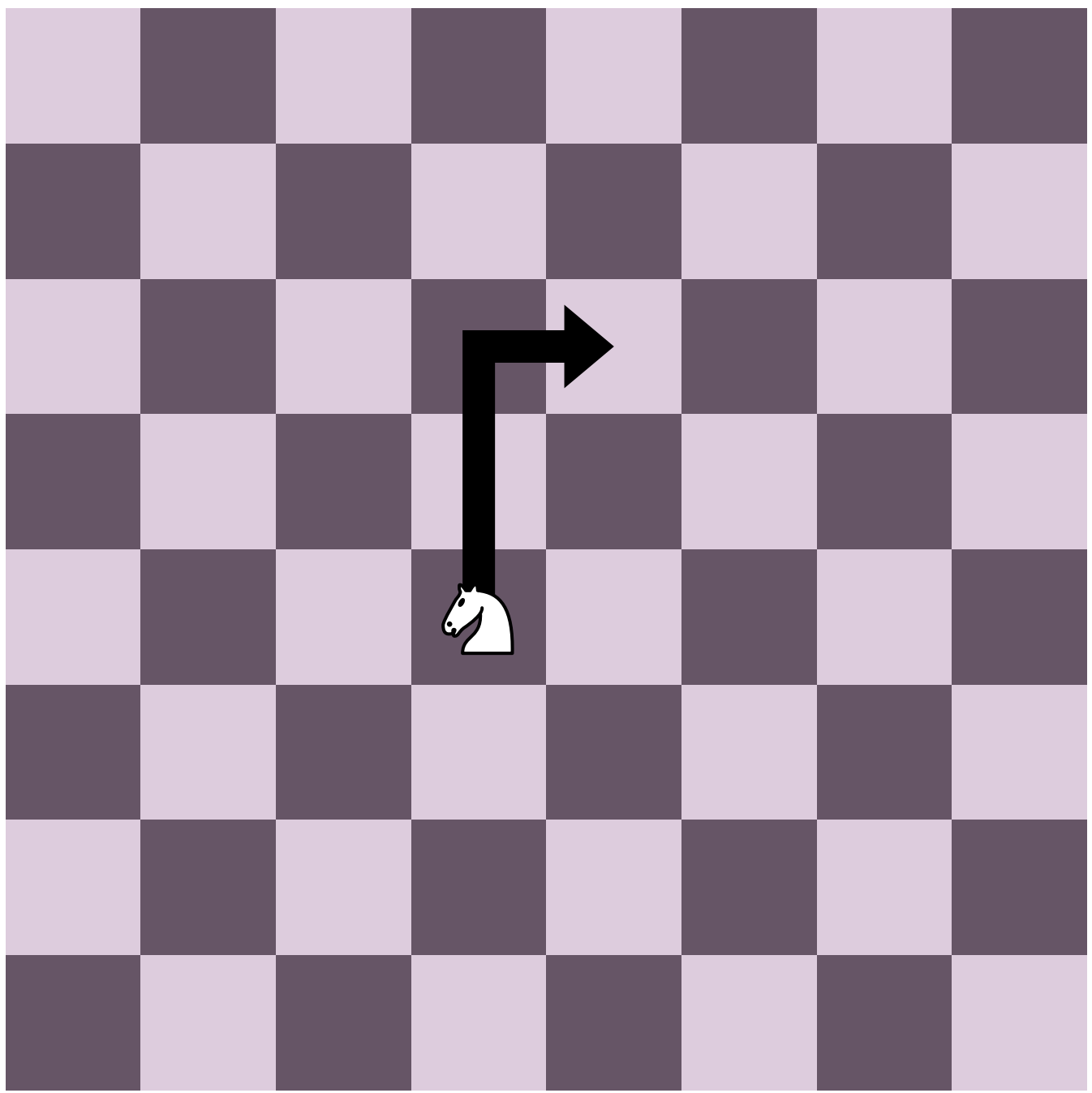
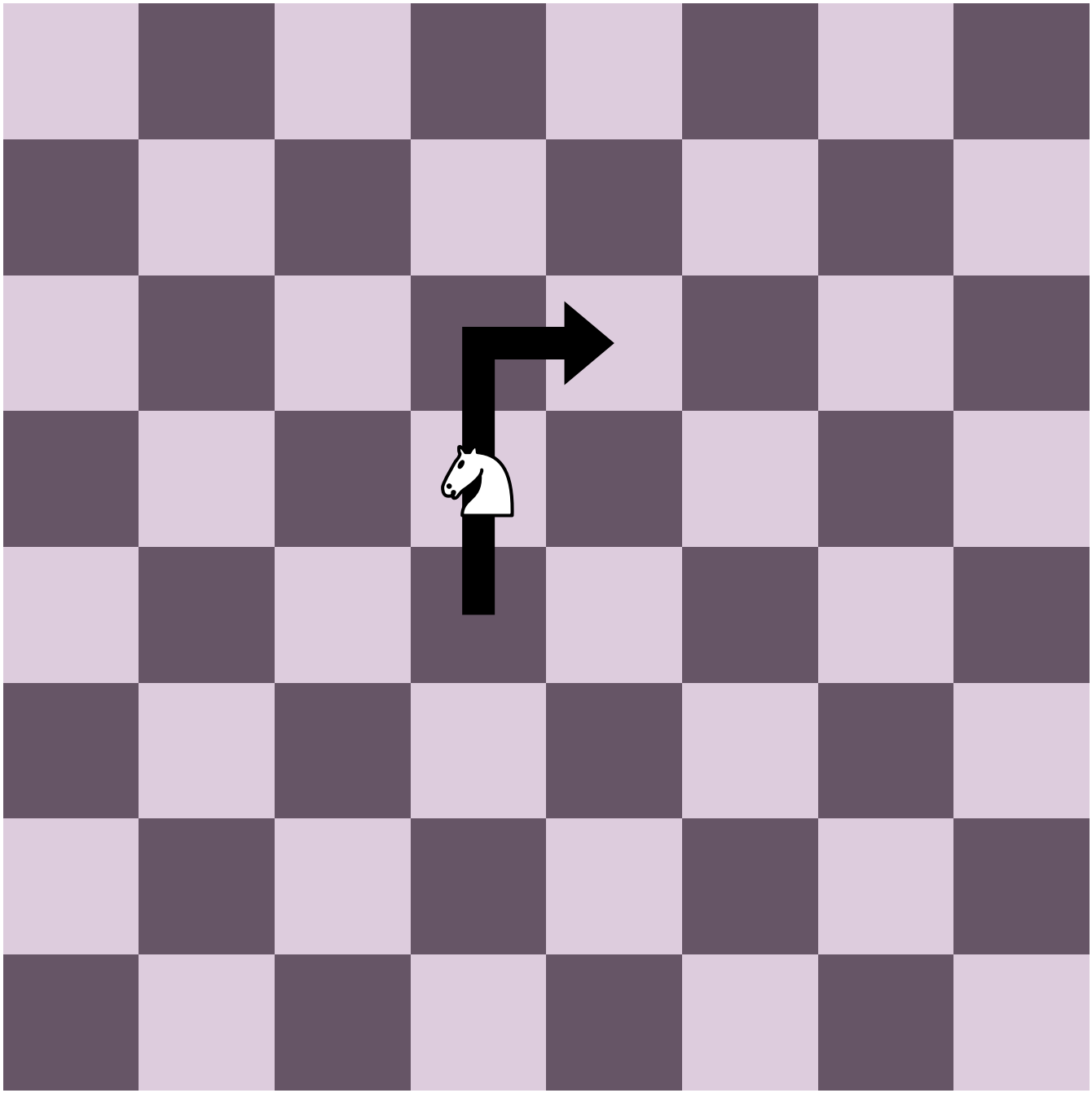
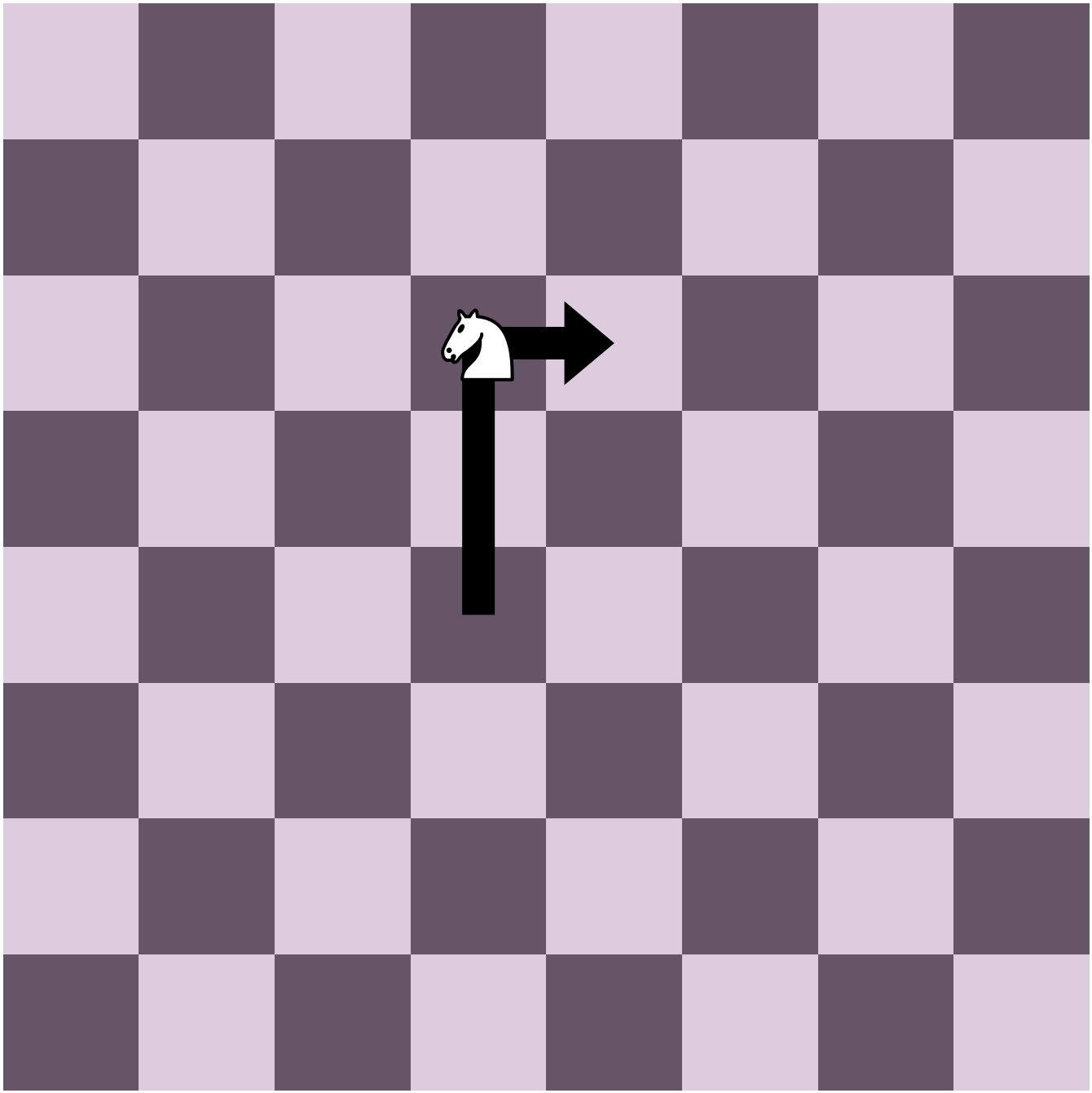


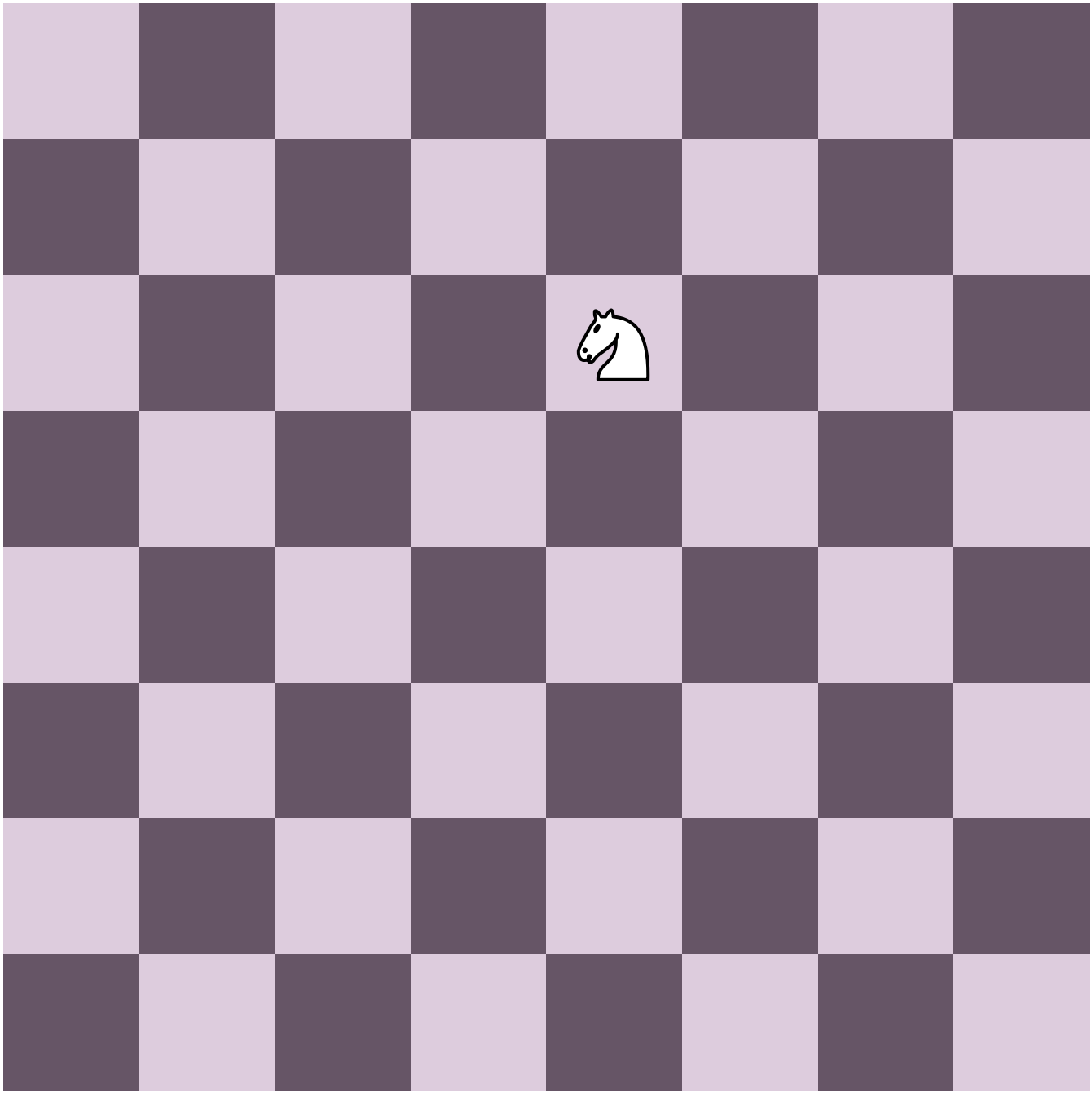
Table demo

Name	Time	Type
Jane	3.5	A1
John	4.1	B7
Johanna	12.0	C1
Elise	12.5	D4
Max	320.2	E1

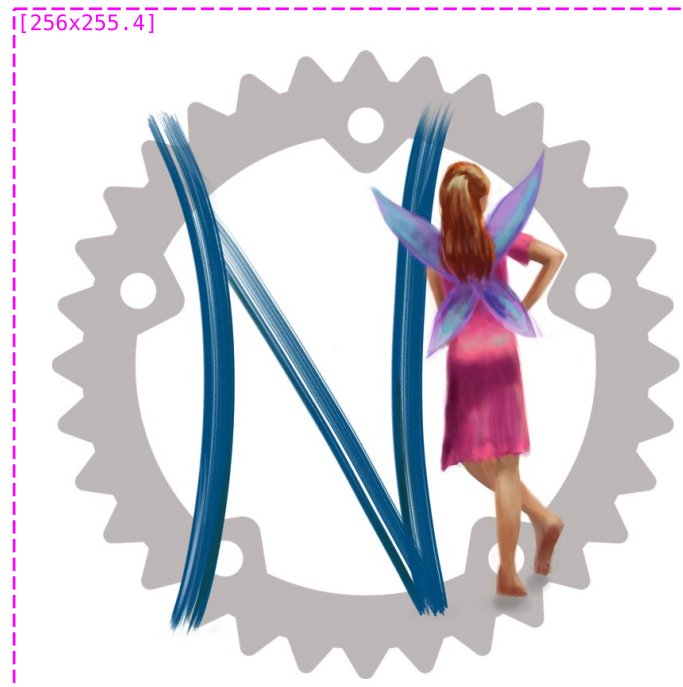








Clickable links in PDF



[317.86x154]

Debugging frames

title box [1024x195]

[196.19x77]

Nelsie

[604.49x53]

Framework for Creating Slides

[173.39x62]

Ada Böhm
ada@kreatrix.org