2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

A SystemVerilog Framework for Efficient Randomization of Images with Complex Inter-Pixel Dependencies

Axel Voss - Gabriel Jönsson - Lars Viklund

AXIS® COMMUNICATIONS

accellera SYSTEMS INITIATIVE

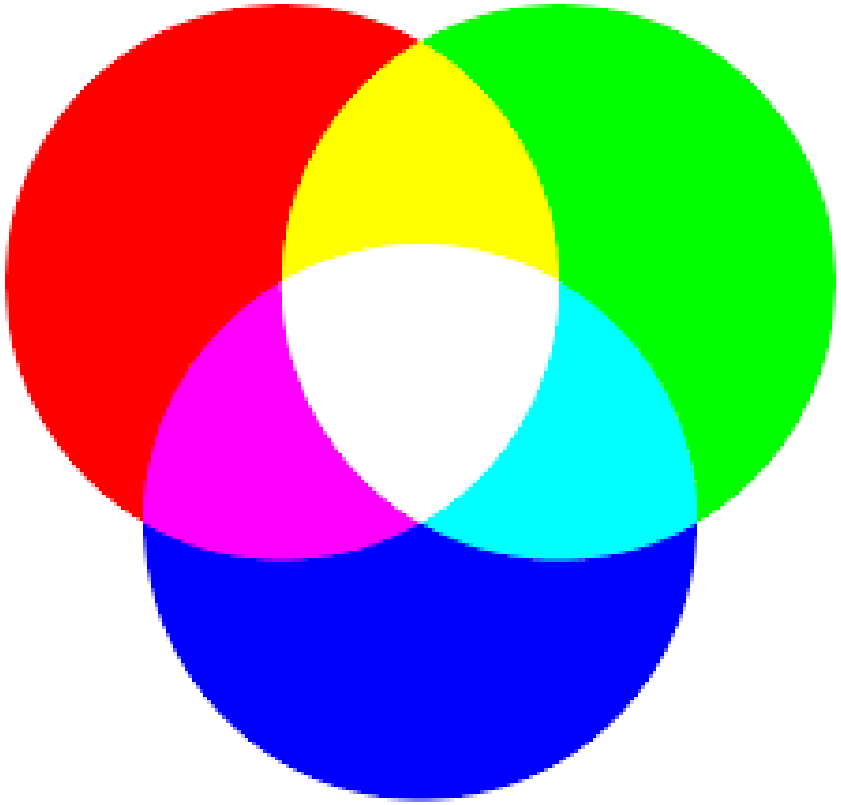
2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Agenda

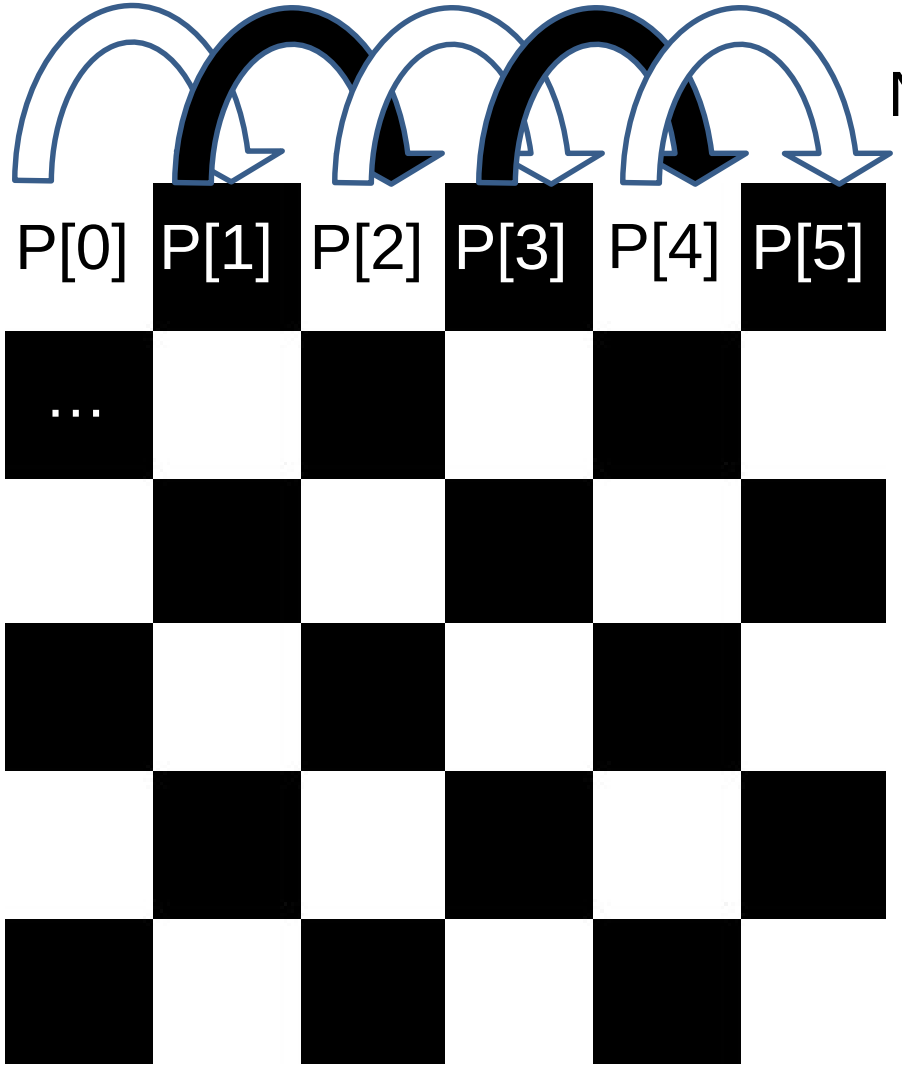
- Introduction
- Implementation with Examples
- Results
- Conclusions

accellera SYSTEMS INITIATIVE

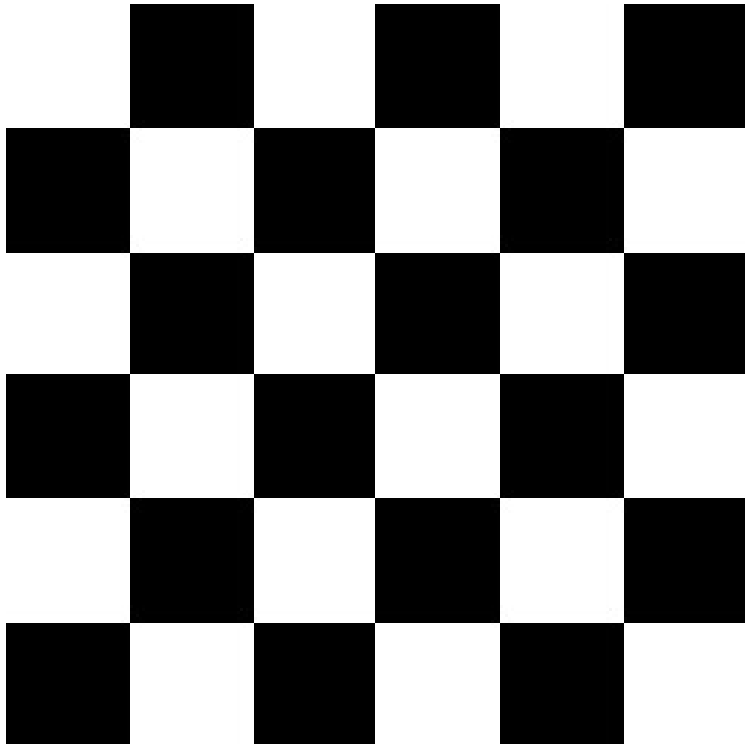
Verification Focus

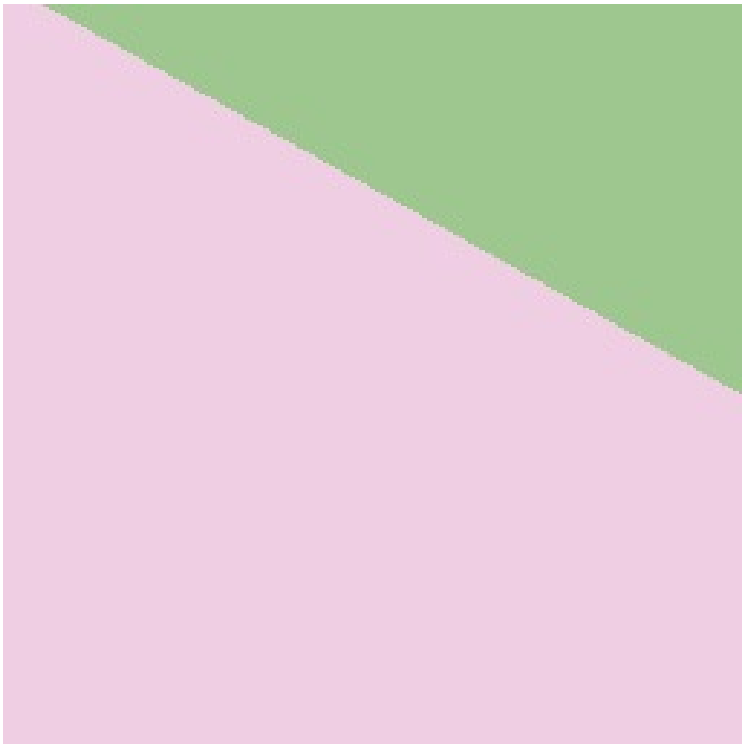


Pixel Dependencies

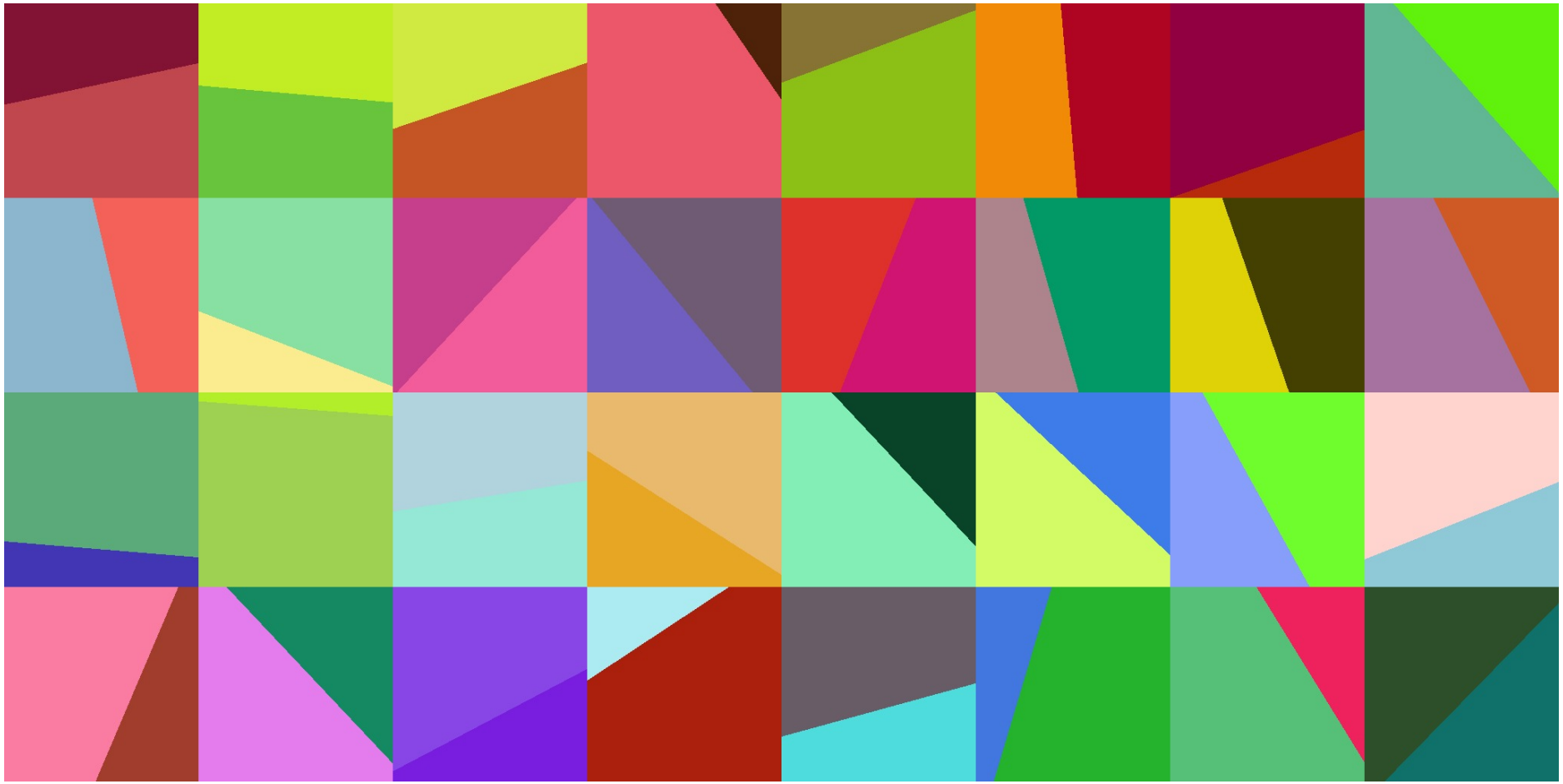


Neighbor based calculation

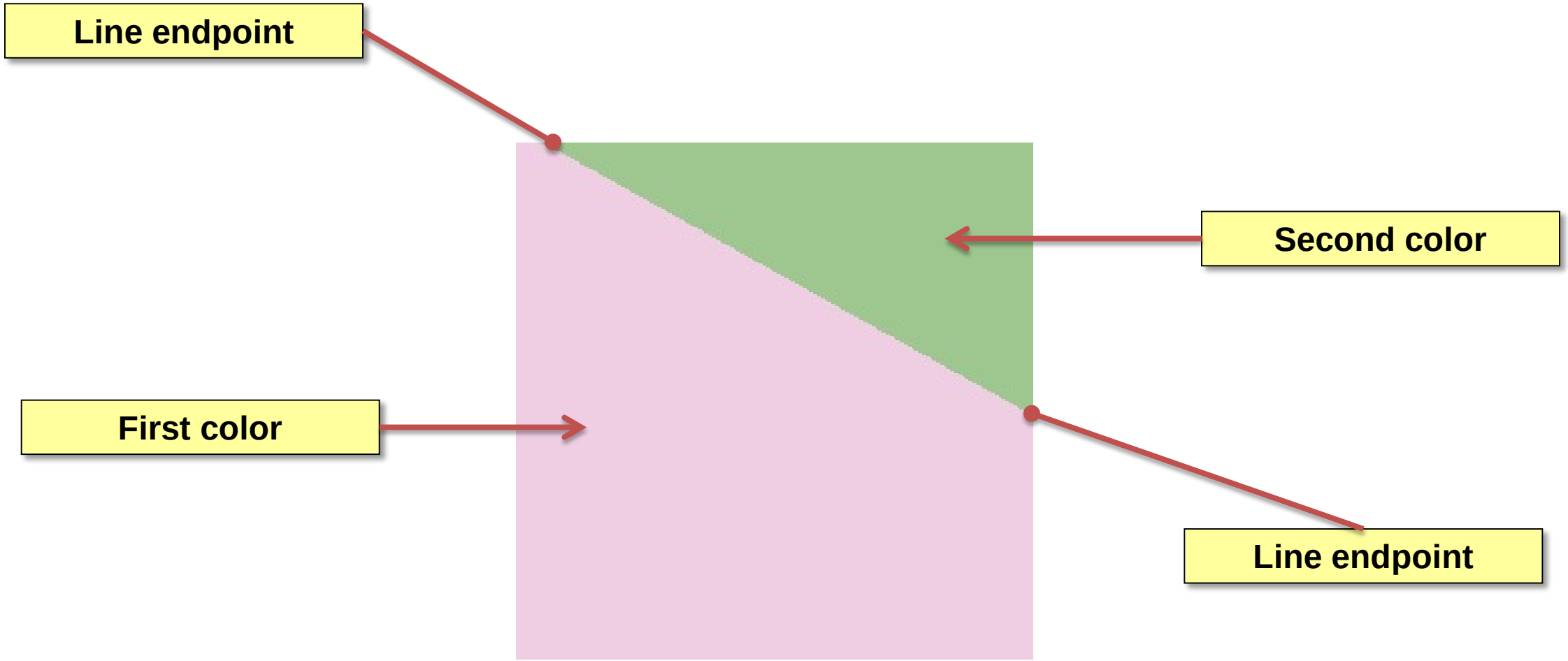




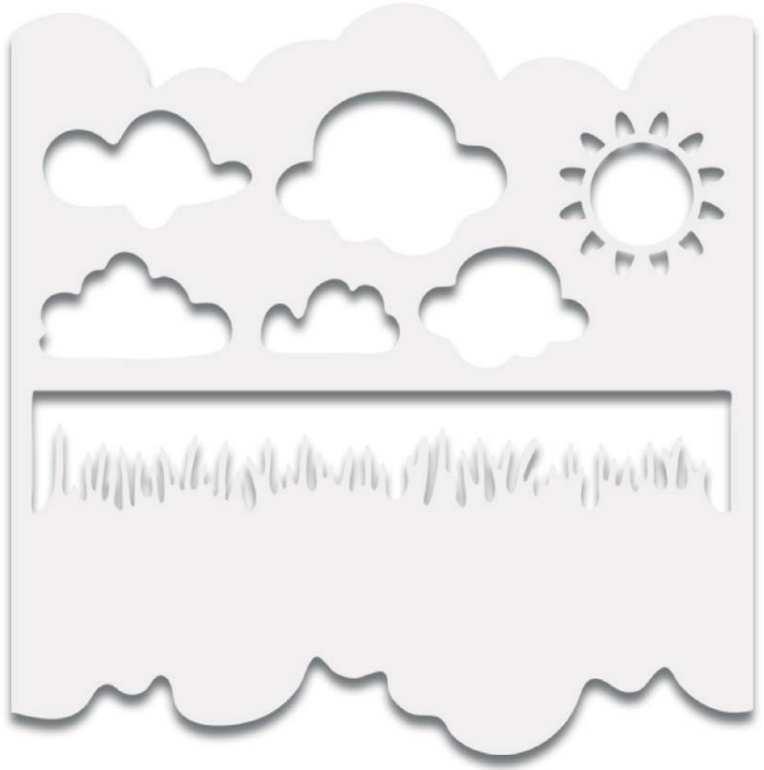
Constraint Hell

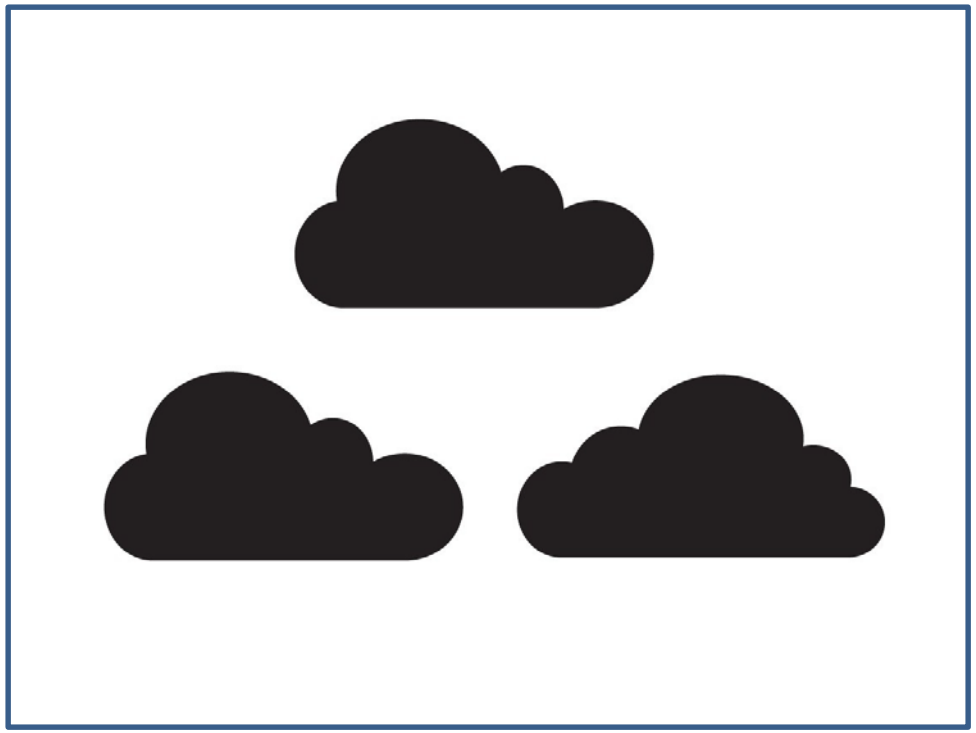


Randomized Features

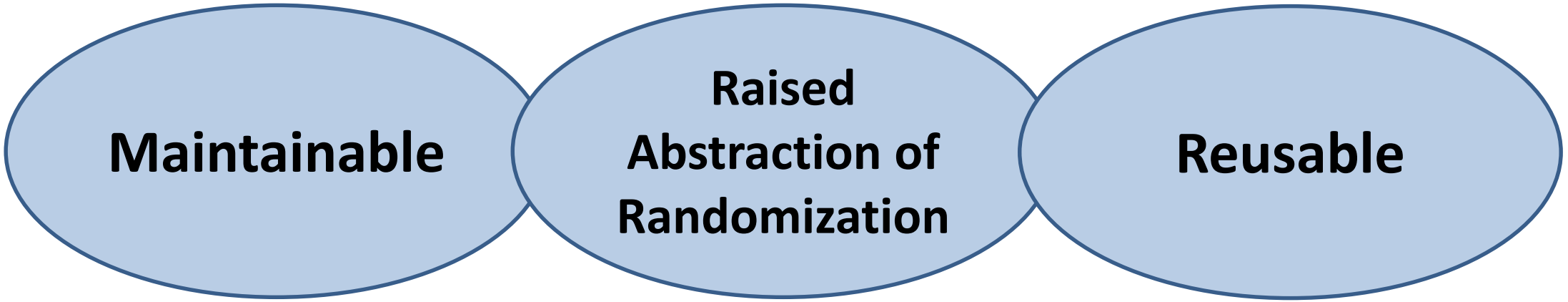


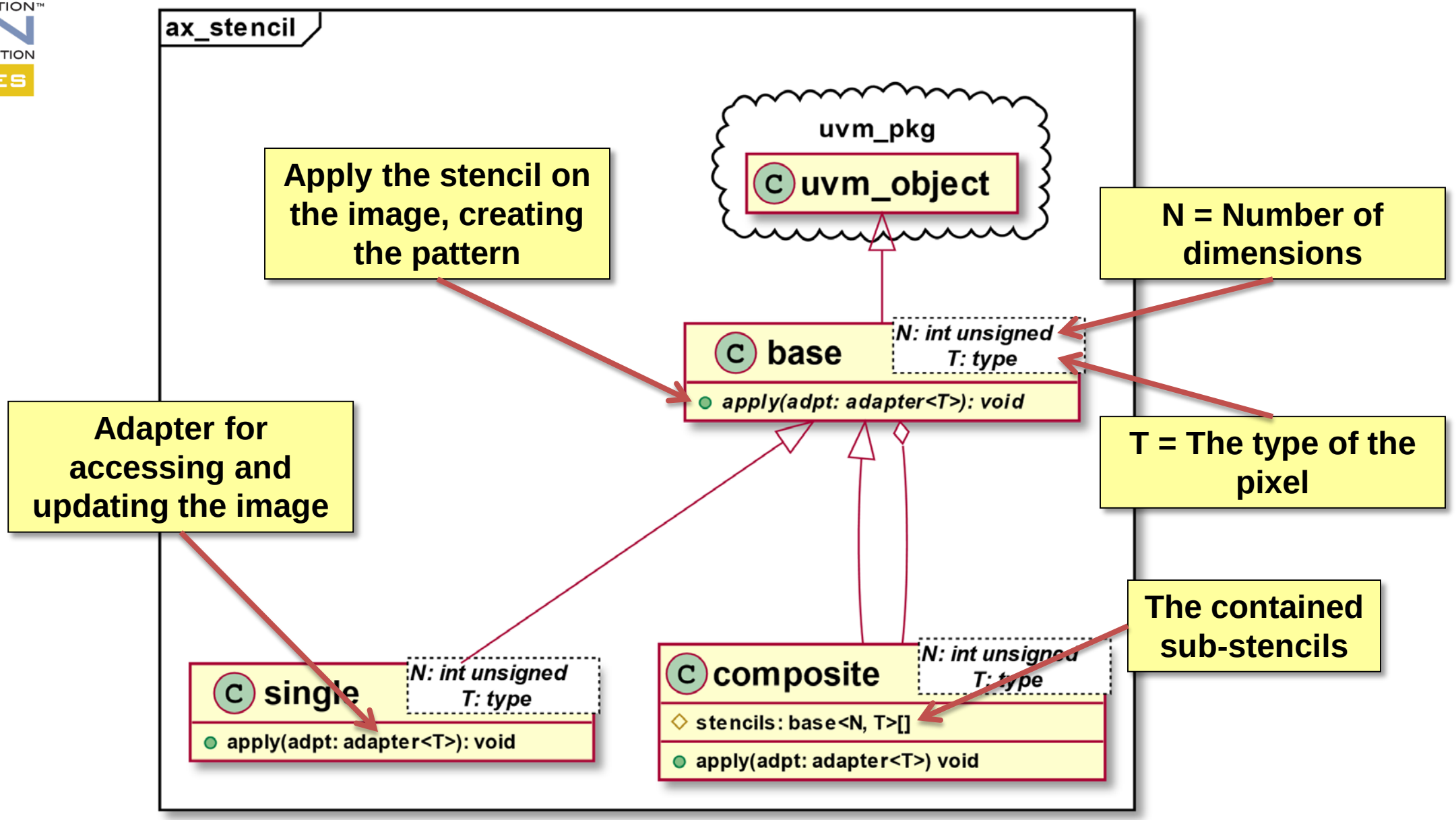
Stencil and Pattern



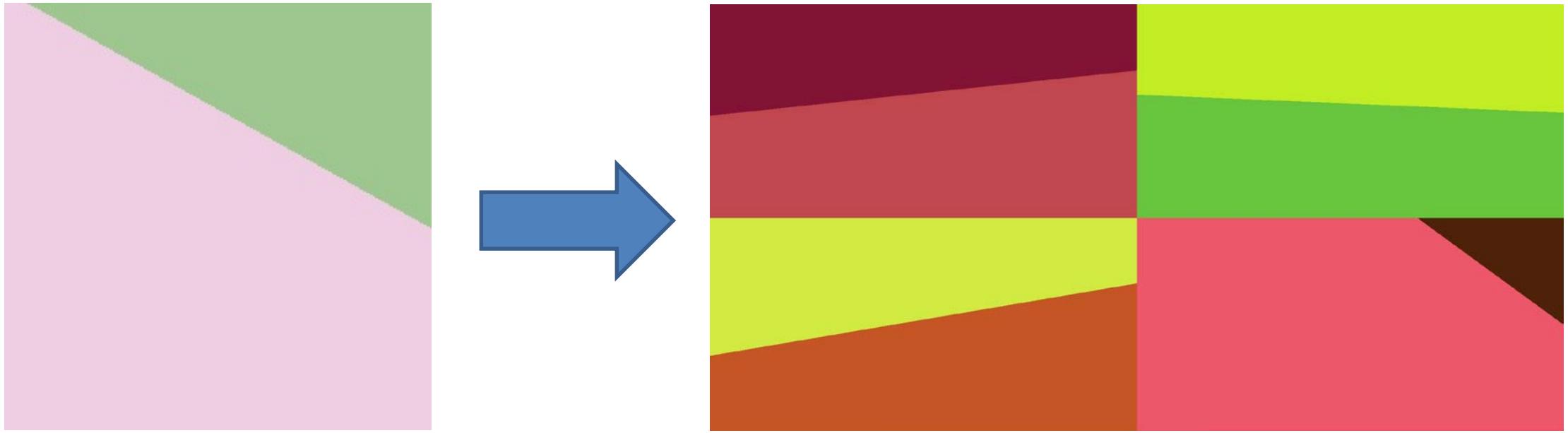


Goal


Maintainable
Raised Abstraction of Randomization
Reusable


ax_stencil
uvm_pkg
uvm_object
Apply the stencil on the image, creating the pattern
N = Number of dimensions
base
N: int unsigned
T: type
apply(adpt: adapter<T>): void
Adapter for accessing and updating the image
T = The type of the pixel
single
N: int unsigned
T: type
apply(adpt: adapter<T>): void
composite
N: int unsigned
T: type
stencils: base<N, T>[]
apply(adpt: adapter<T>) void
The contained sub-stencils

Single and Composite



2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

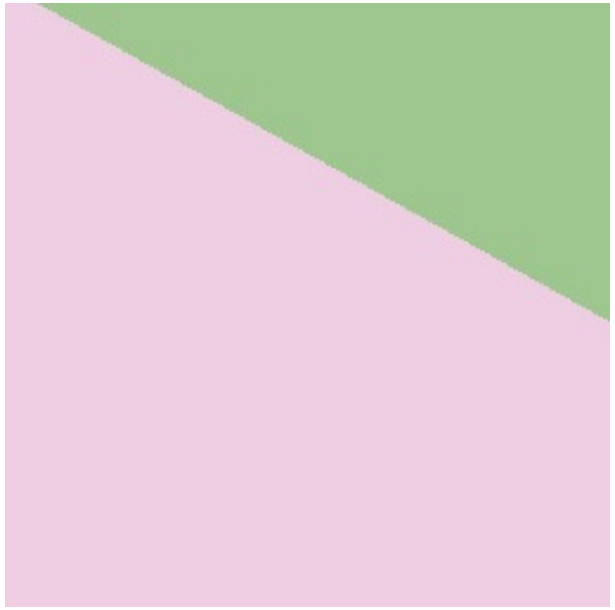
Features

- Decoupled from data representation
- Flexible pixel type support
- Easy to constrain values in the pattern
- Randomize multiple patterns
- Any number of dimensions

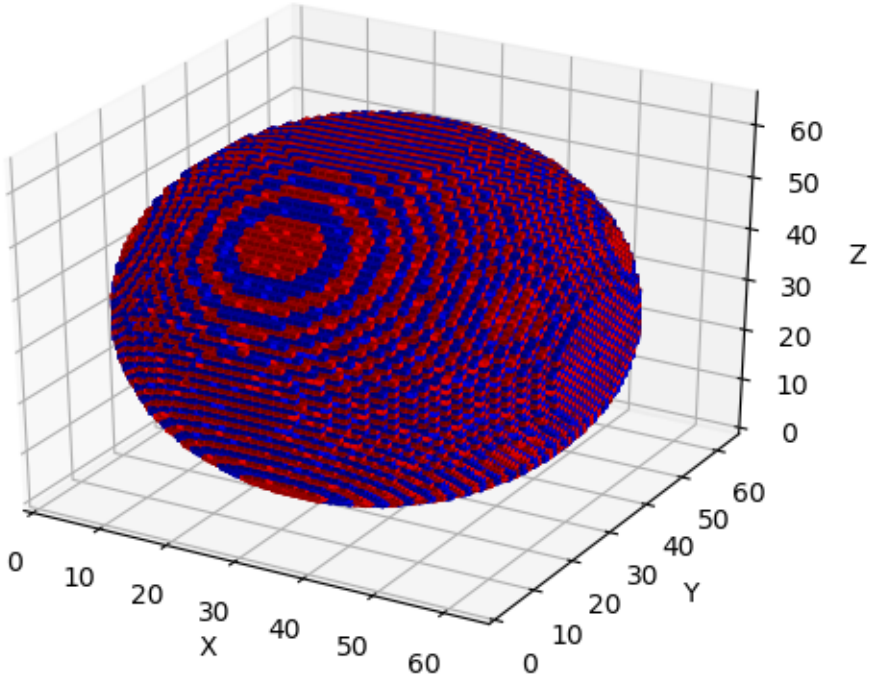
accellera SYSTEMS INITIATIVE

Patterns in Multiple Dimensions

N = 2

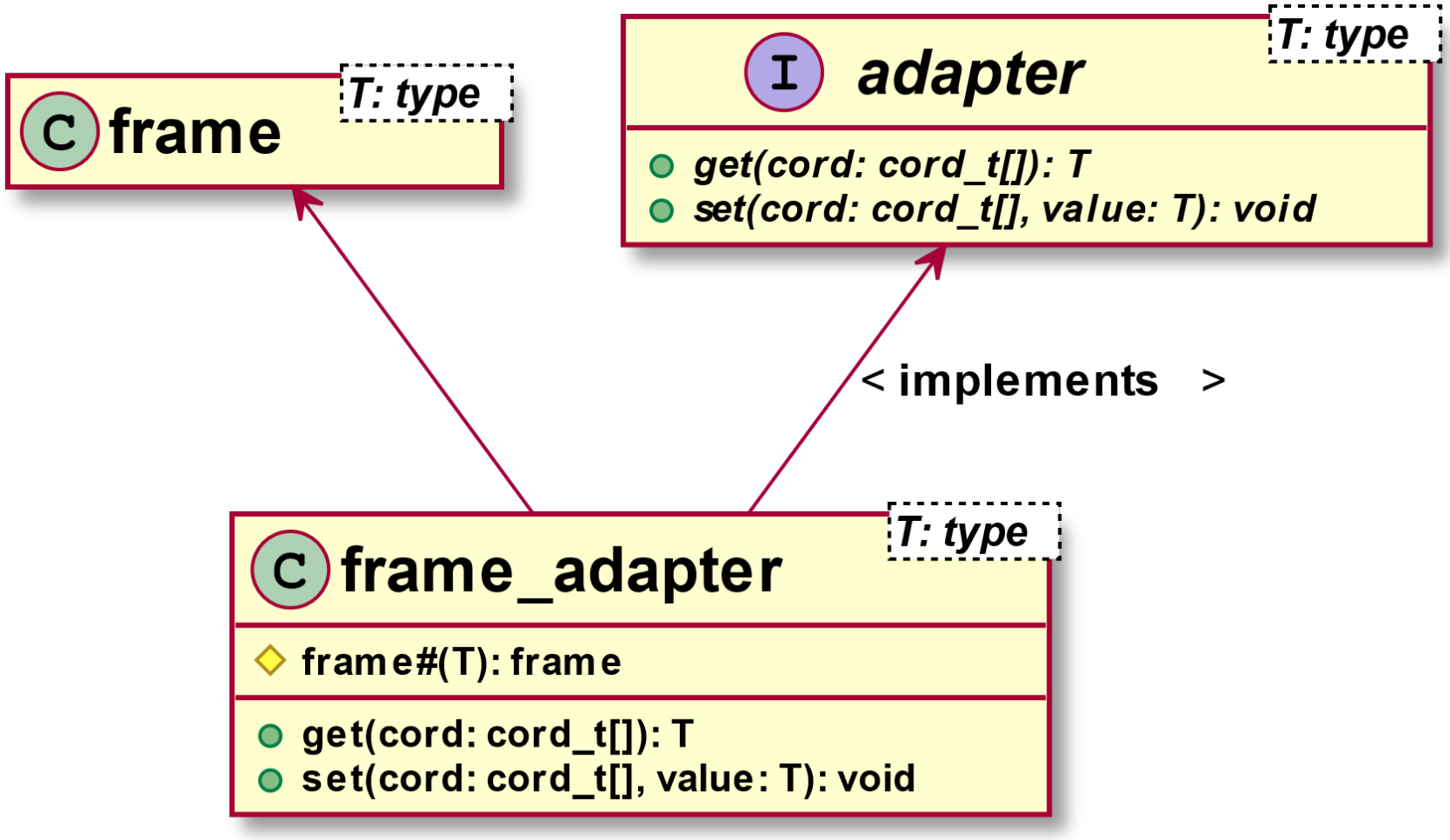


N = 3



N = 1

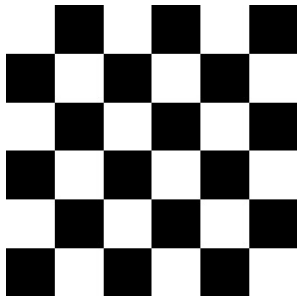
Frame Adapter Implementation



Frame Adapter Implementation

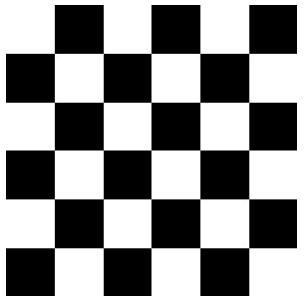
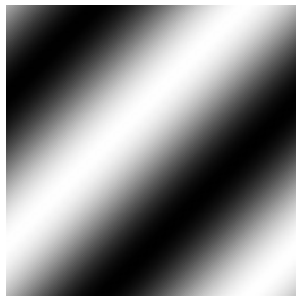
```
class frame_adapter implements adapter#(int unsigned);
  frame fr; // Frame handle

  function new(frame fr);
    ……
  endfunction : new

  function int unsigned get(/* Coordinates */);
    ……
  endfunction : get

  function void set(/* Coordinates */, int unsigned value);
    ……
  endfunction : set

endclass : frame_adapter
```

id = 0

Available Stencils = '{  }

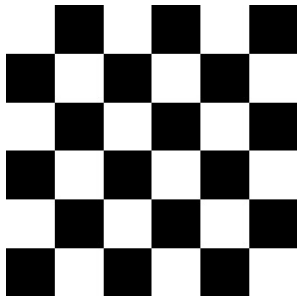
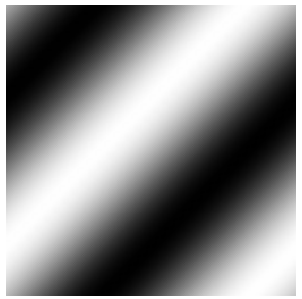
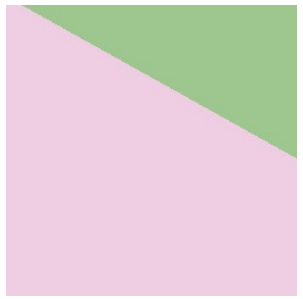
Composite

id = 0 1

Available Stencils = '{  ,  }

Composite

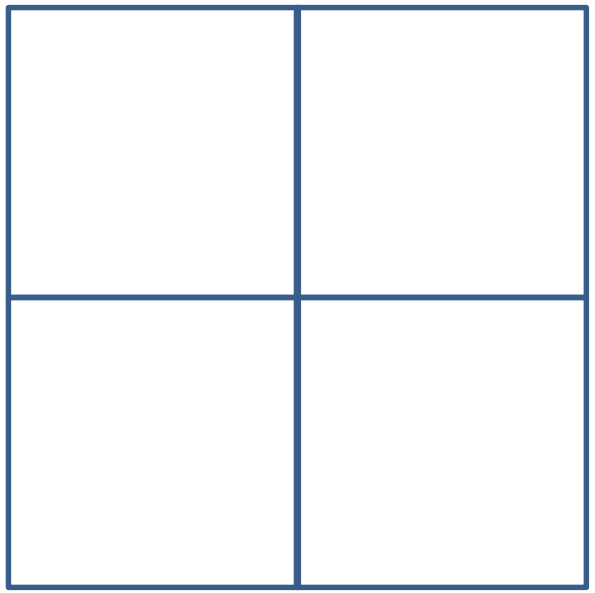
id = 0 1 2

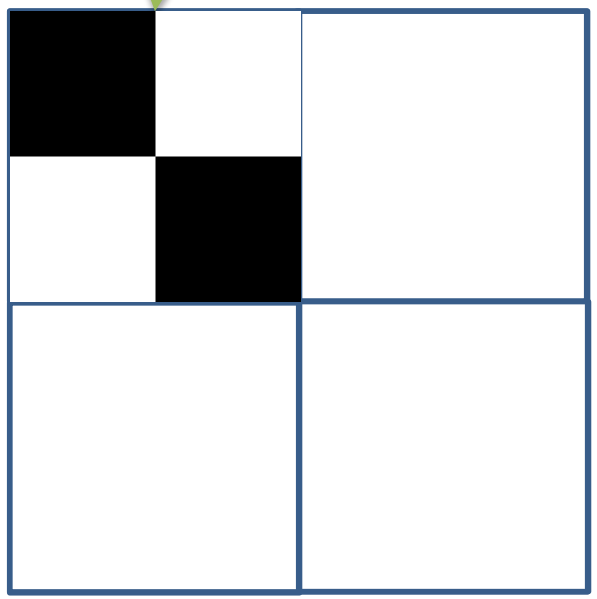
Available Stencils = '{  ,  ,  }

Composite

ids = '{0, 2, 1, 0}

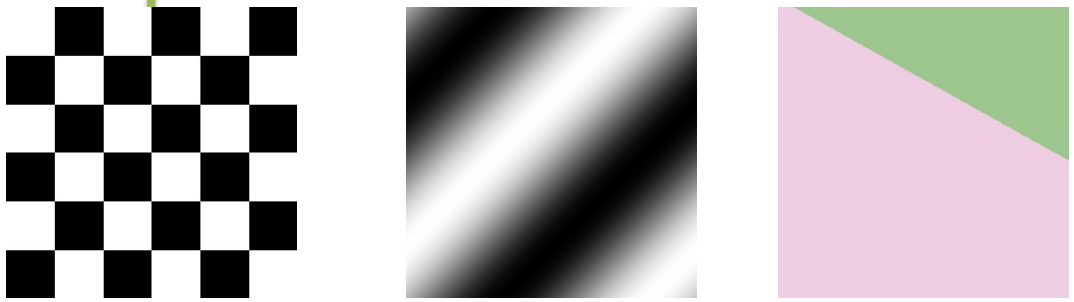
Randomize



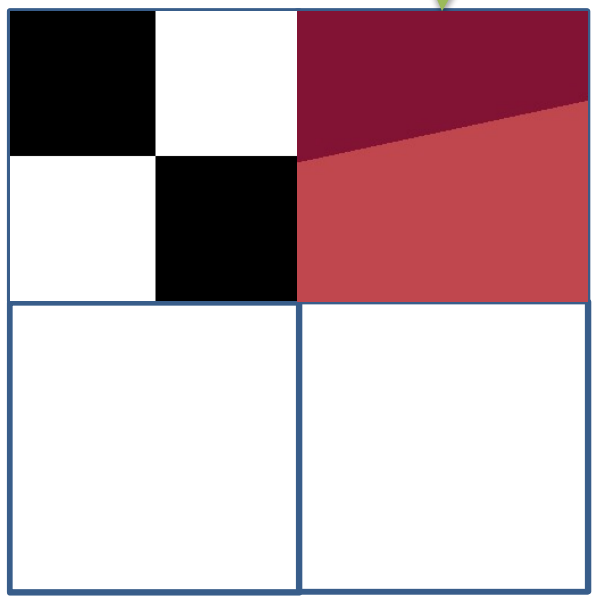


randomize ← clone

id = 0 1 2

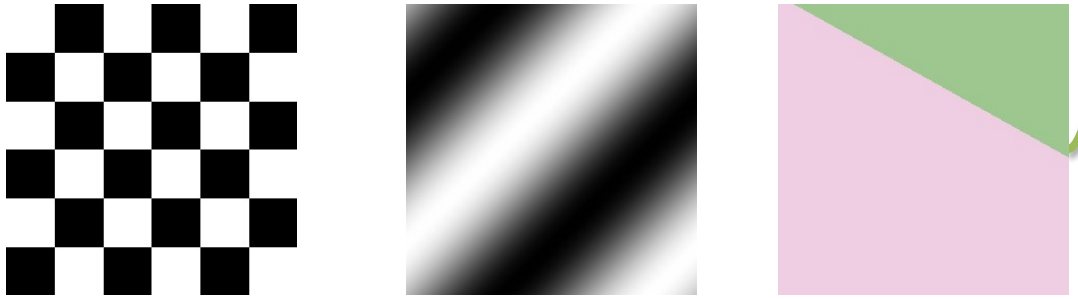


ids = '{0, 2, 1, 0}

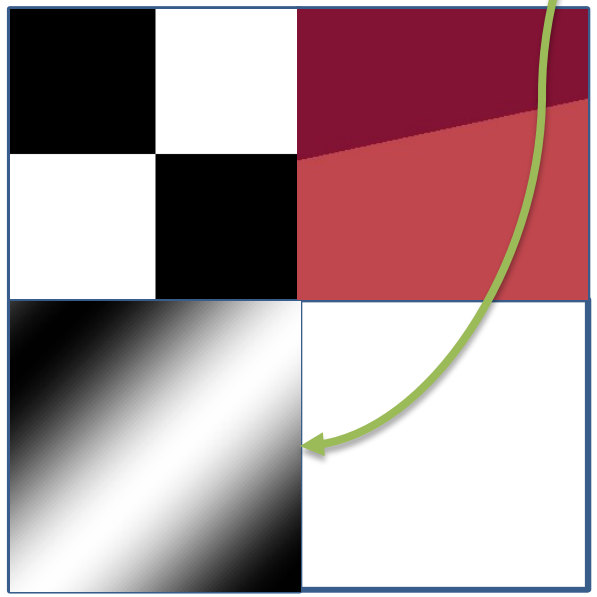


randomize ← clone

id = 0 1 2

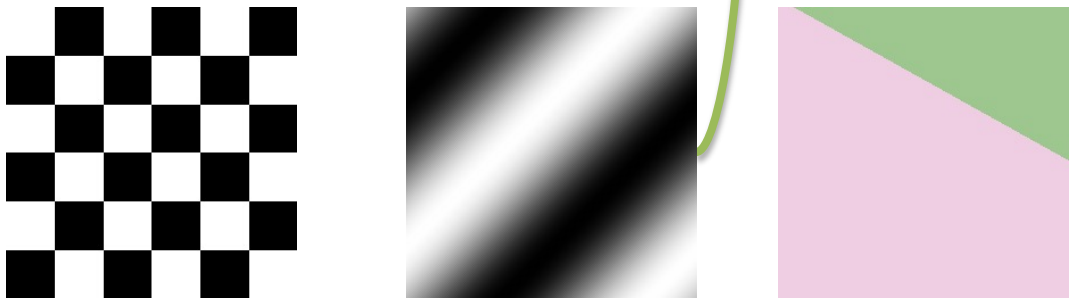


ids = '{0, 2, 1, 0}



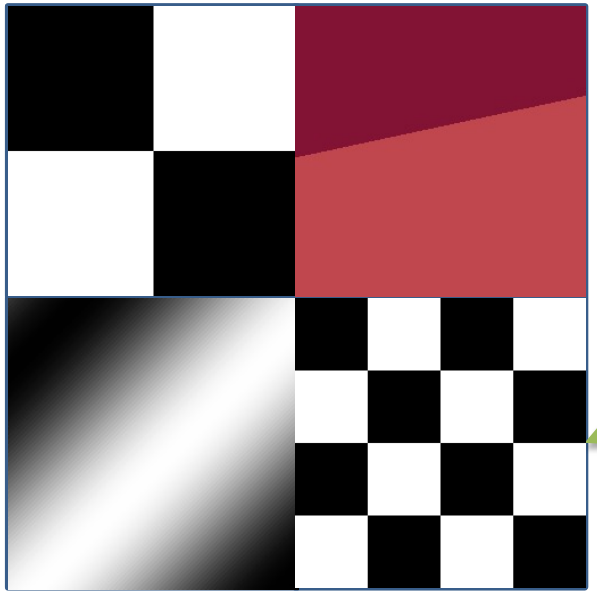
randomize ← clone

id = 0 1 2

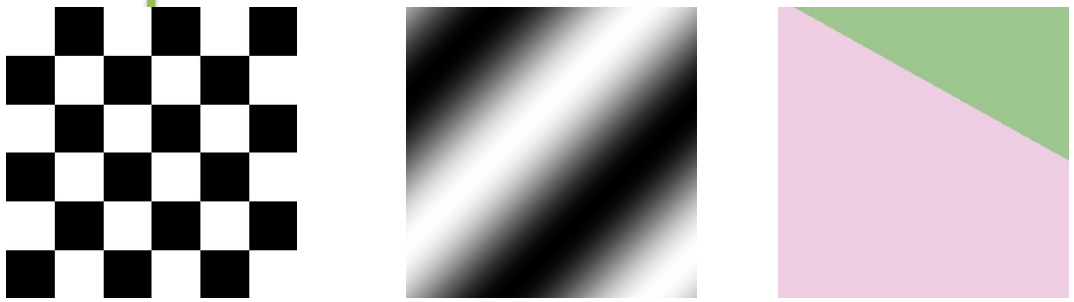


ids = '{0, 2, 1, 0}

randomize ← clone



id = 0 1 2



ids = '{0, 2, 1, 0}

Stencil Example – Vertical Line

```
class vertical_line#(type T) extends single#(2, T);
    rand int unsigned x_line_pos;
    rand T left_color, right_color;
    ……

    function T calc_value(cord_t cord[]);
      bit val = (cord[0] - this.origin[0]) > this.x_line_pos;
      calc_value = val ? this.right_color : this.left_color;
    endfunction : calc_value

    constraint c_x_pos {
      this.x_line_pos < this.length[0];
    }
  endclass : vertical_line
```

Locking down the dimension number

Randomized features of the stencil

Calculate value dependent on position

Vertical line inside the stencil

Pattern Examples – Vertical Line

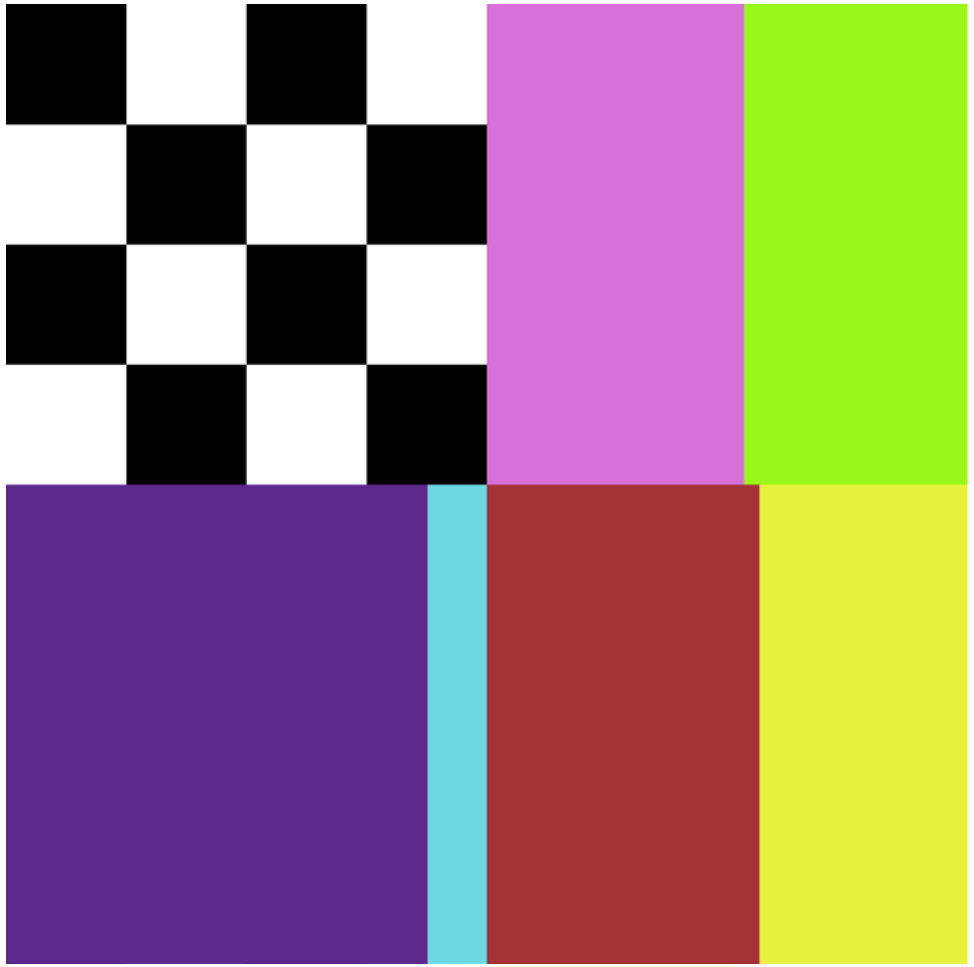
```
stencil = vline_t::type_id::create();

adapter = new(256, 256);

stencil.randomize() with {length[0] == 256; length[1] == 256};

stencil.apply(adpt);
```

Pattern Examples - Grid

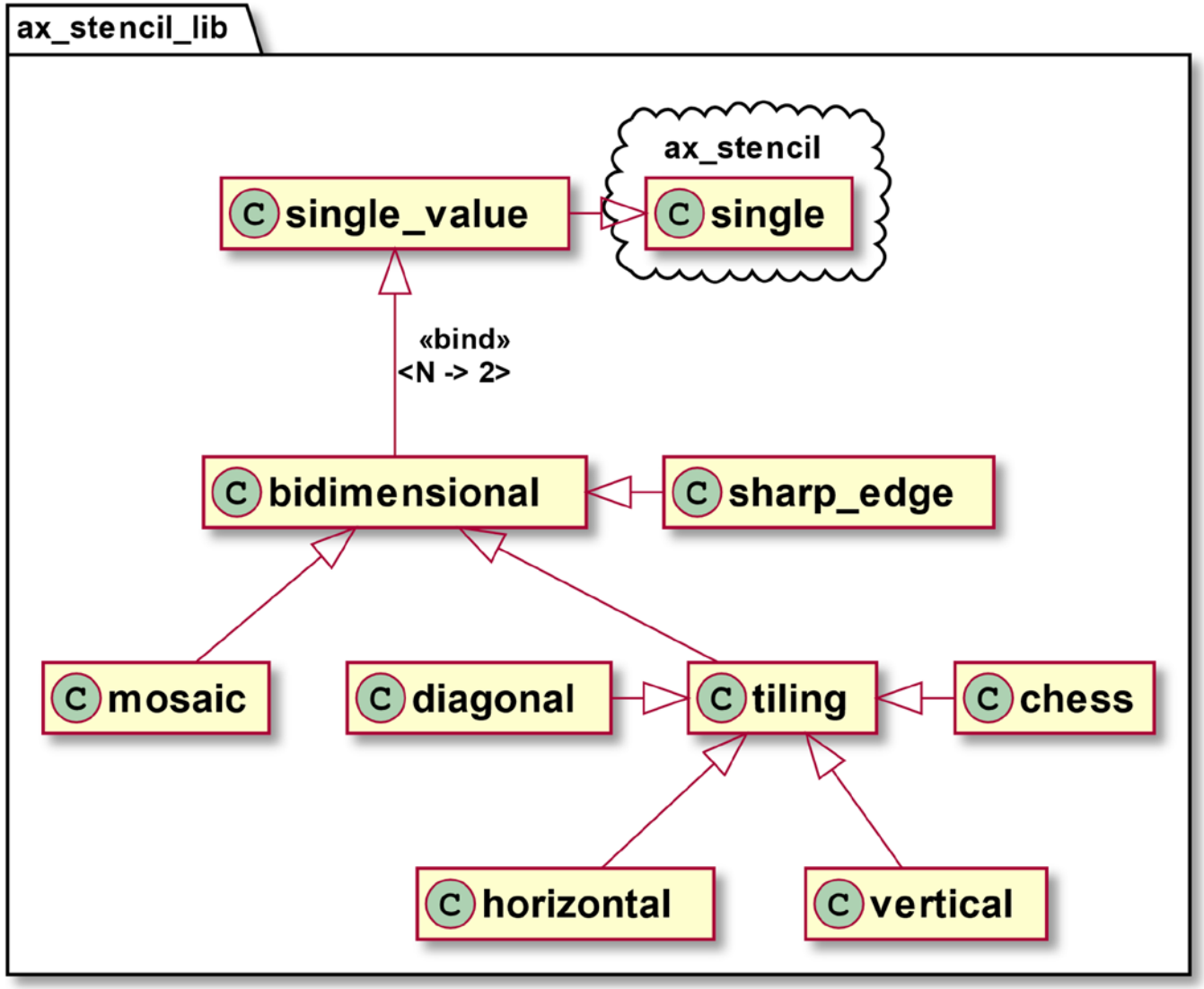
```
class grid_example extends grid#(int unsigned);
  vertical_line#(int unsigned, integral_values) vline;
  chess#(int unsigned, integral_values)         chess;
  ……

  function new(string name = "grid_example");
    ……
    // creating patterns & adding value decorators
    ……
    this.add_stencil(this.vline);
    this.add_stencil(this.chess);
  endfunction : new

endclass : grid_example
```

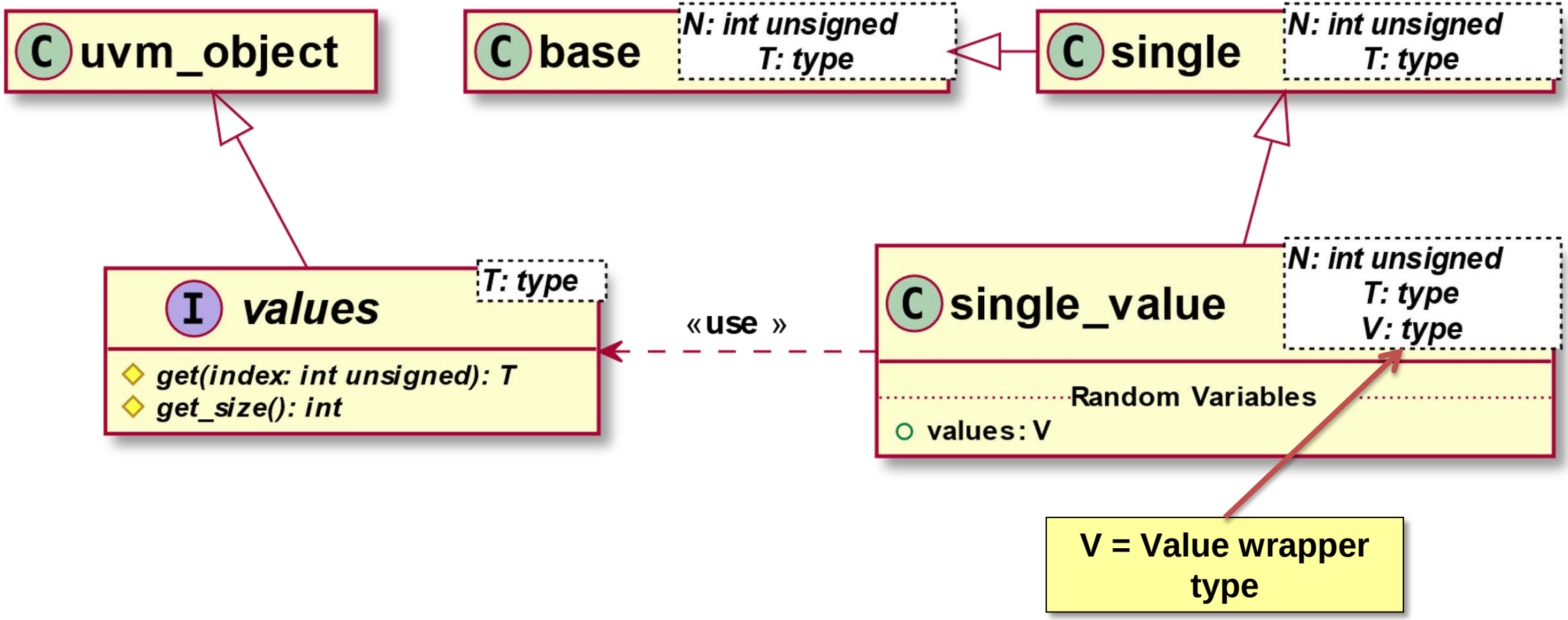
Pattern Examples - Grid

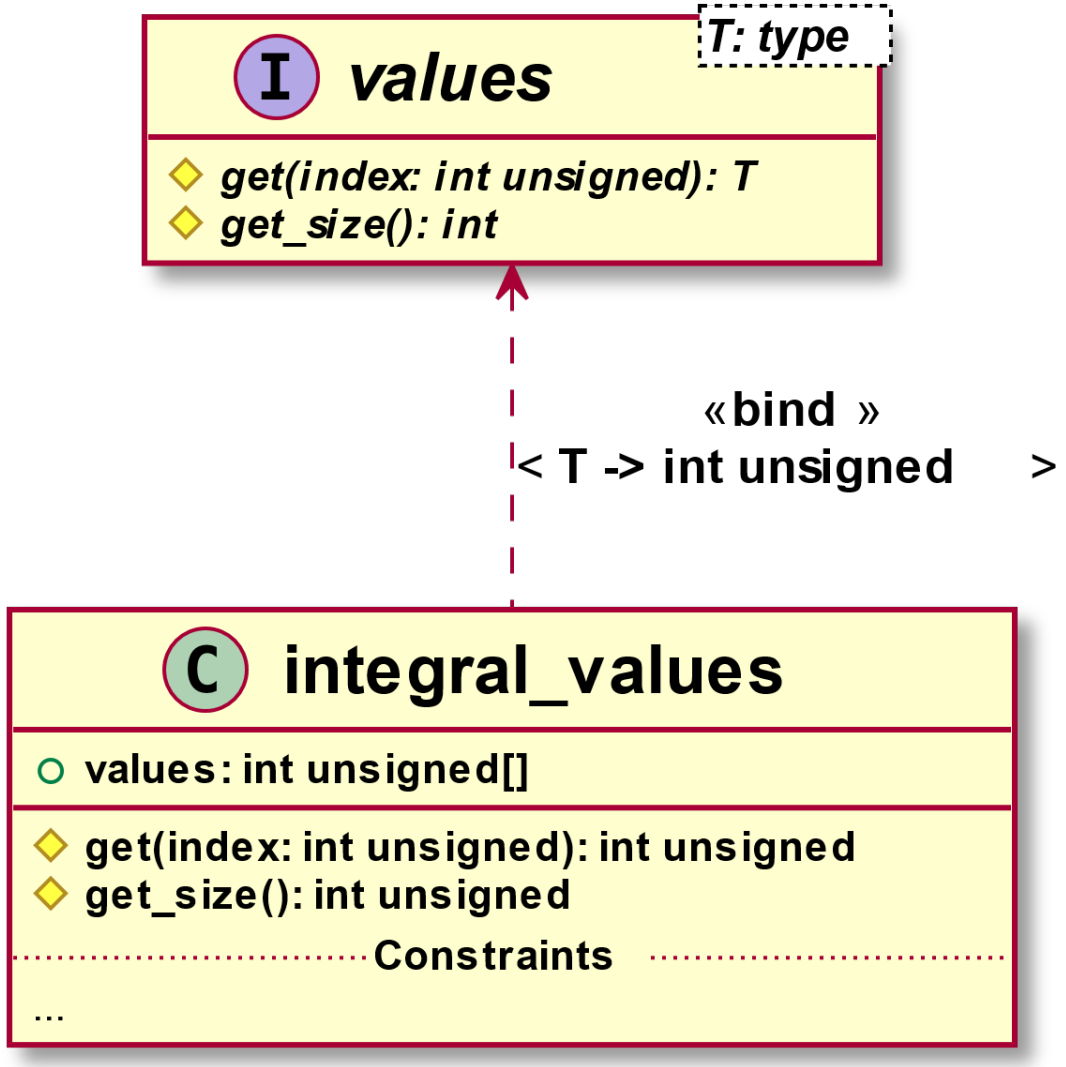
```
grid_stencil.randomize() with {
  length[0] == 256;
  length[1] == 256;
  dx        == 128;
  dy        == 128;
};
```

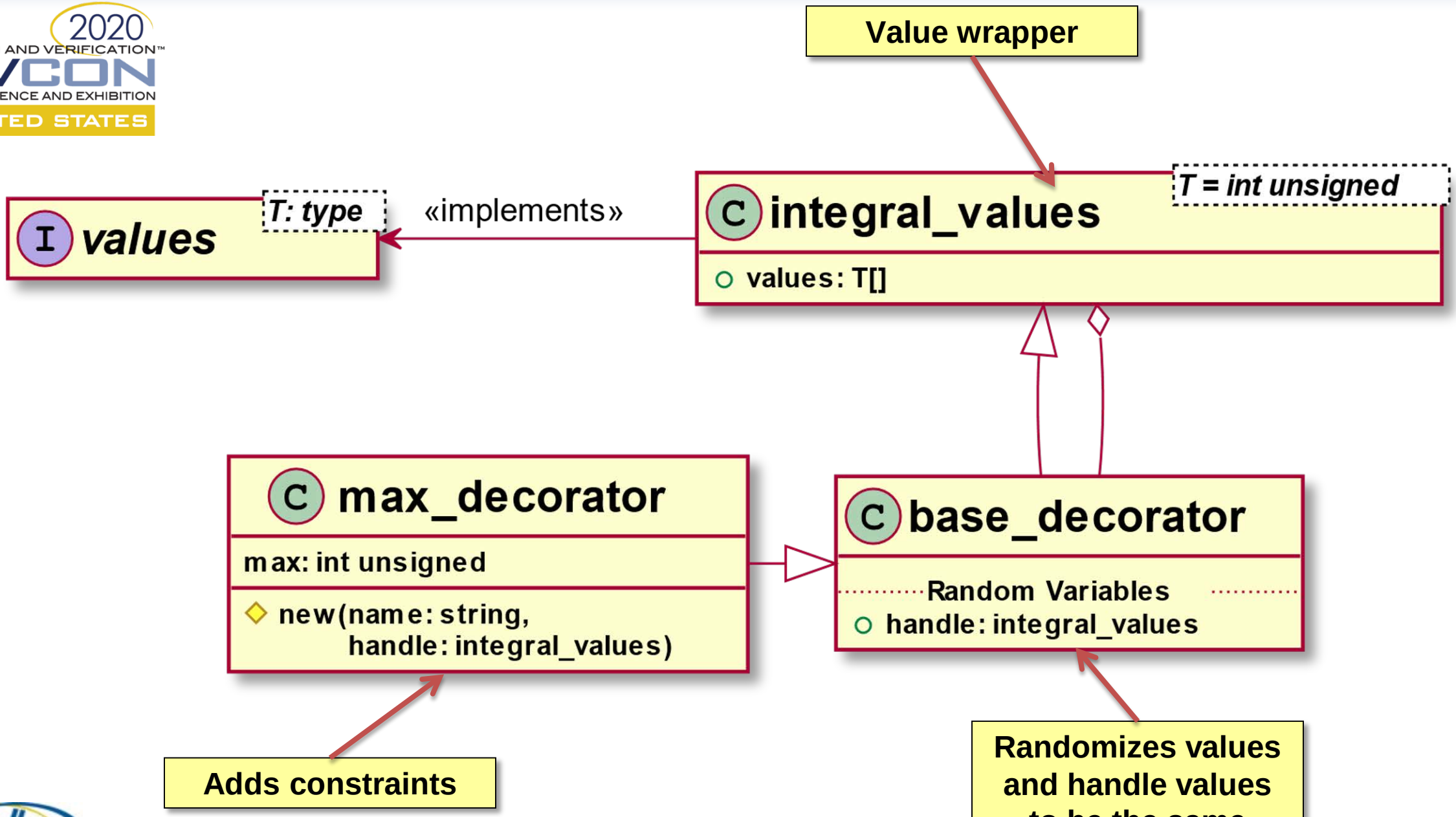


Stencil Library




uvm_object
base
N: int unsigned
T: type
single
N: int unsigned
T: type
values
T: type
get(index: int unsigned): T
get_size(): int
«use»
single_value
N: int unsigned
T: type
V: type
Random Variables
values: V
V = Value wrapper type


I values
T: type
get(index: int unsigned): T
get_size(): int
«bind »
< T -> int unsigned >
C integral_values
values: int unsigned[]
get(index: int unsigned): int unsigned
get_size(): int unsigned
Constraints
...


Value wrapper
I values
T: type
«implements»
C integral_values
T = int unsigned
values: T[]
C max_decorator
max: int unsigned
new(name: string,
handle: integral_values)
C base_decorator
Random Variables
handle: integral_values
Adds constraints
Randomizes values and handle values to be the same

Decorator Example

```
stencil = pattern_t::type_id::create();
values  = integral_values::new("integral_values");
values  = dec_rgb::new("rgb", values);
values  = dec_few::new("few", values);
values  = dec_even::new("even", values);
values  = dec_max::new("max", values);

pattern.values = values;
```

Base value wrapper

Passing handle

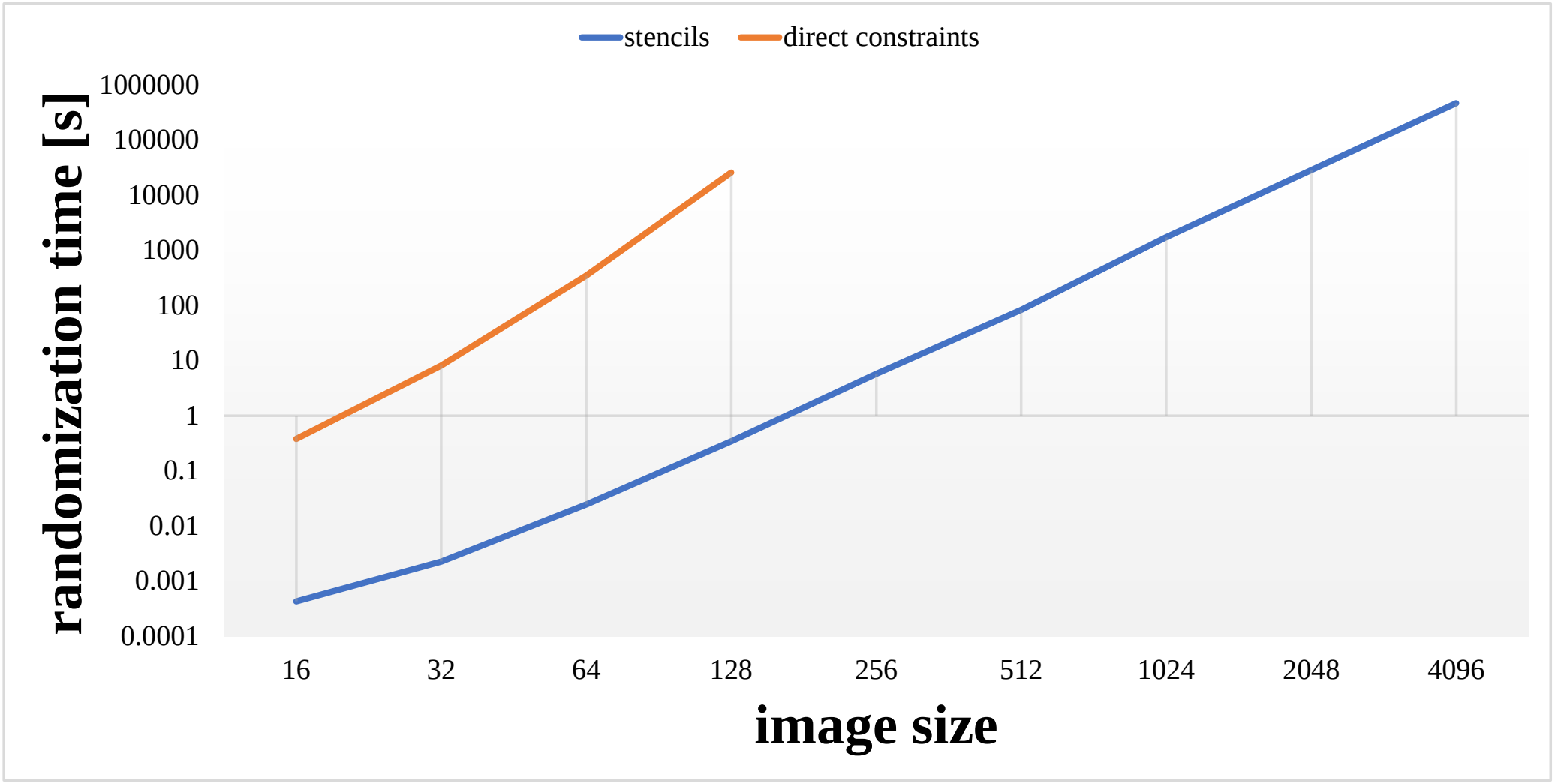
Passing handle

Passing handle

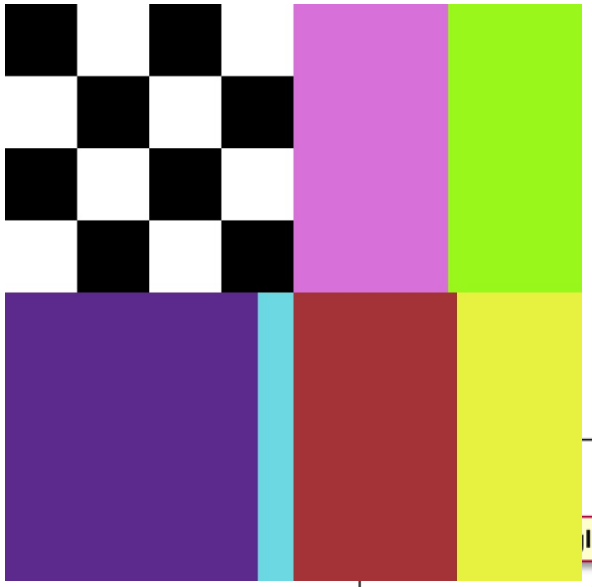
Passing handle

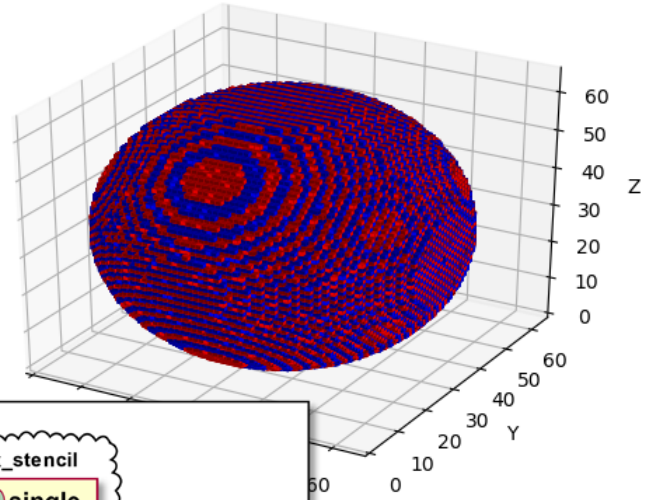
Assigning decorated values

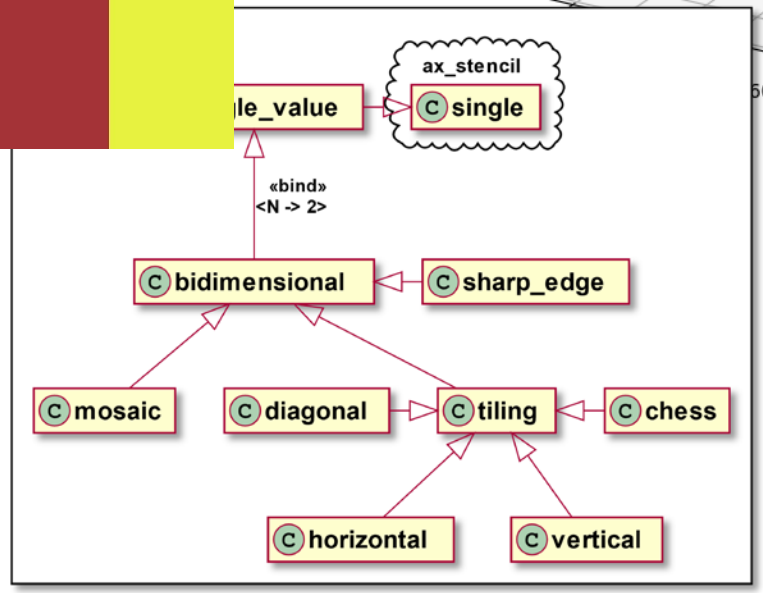
Performance

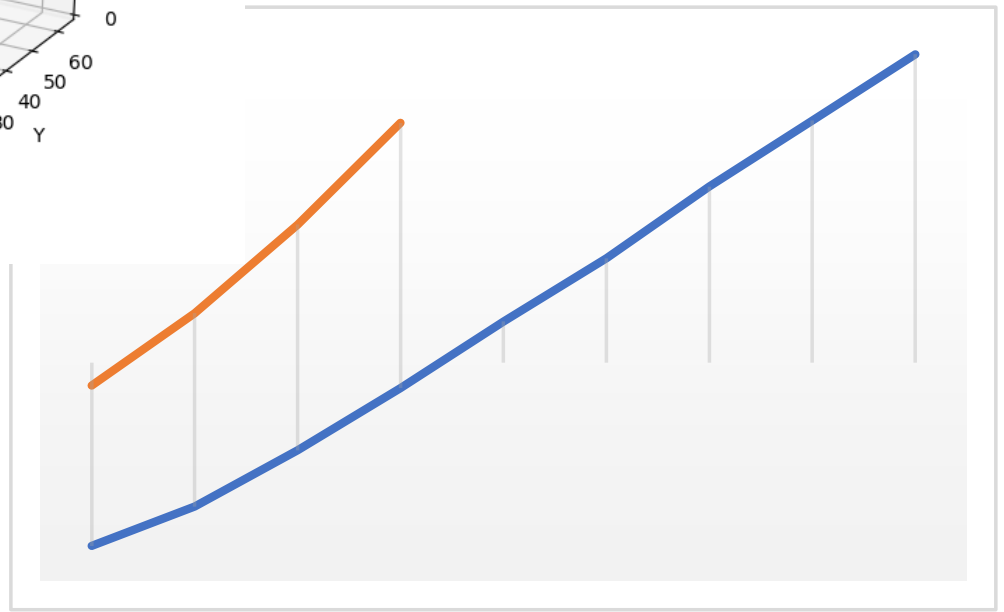


Conclusions










Megapixel
Quality
Ease of use
Network video
Reliability
open
Innovation
Easy installation
Protect
Service
all the way
support
integration
Thank you!
Image usability
Thermal
Convergence
HDTV
Access control
World Leader
Partners
Intelligent
Competence
Quality
Focus
Video encoder
Safe
Cameras
Outdoor
Secure
Global