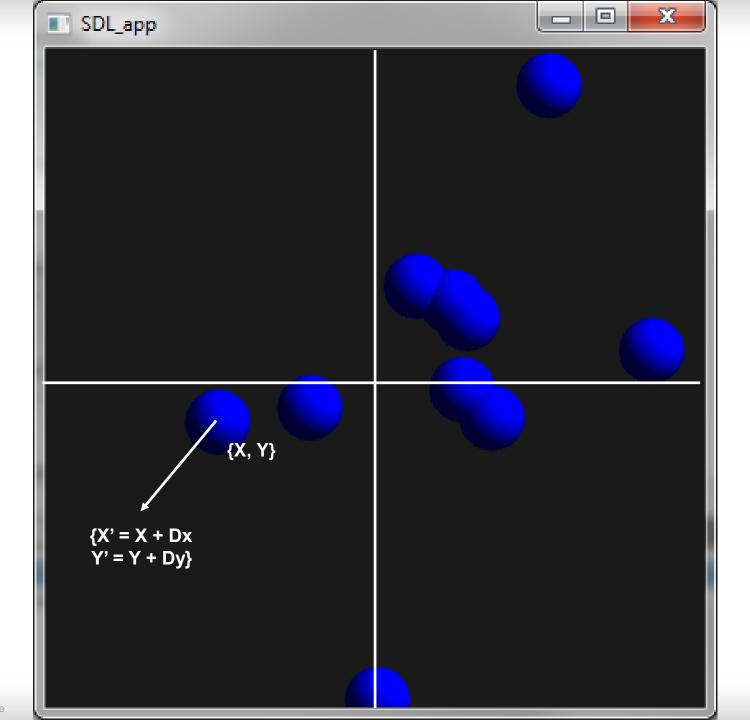


Presented by Quentin Ochem

university.adacore.com



```
with Display;
                              use Display;
                              use Display.Basic;
with Display.Basic;
with Ada. Numerics. Float Random; use Ada. Numerics. Float Random;
procedure Main is
   type Ball Type is record
     Shape : Shape Id;
     Dx, Dy : Float;
   end record;
   type Ball Array is array (Integer range <>) of Ball Type;
   Seed : Generator;
   Balls : Ball Array (1 .. 10) :=
     (others =>
        (Shape => New Circle (0.0, 0.0, 10.0, Blue),
         Dx = (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0),
              => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0)));
   procedure Iterate (V : in out Ball Type) is
   begin
      if Get X (V.Shape) not in -100.0 .. 100.0 then
        V.Dx := -V.Dx;
      end if;
      if Get Y (V.Shape) not in -100.0 .. 100.0 then
        V.Dy := -V.Dy;
      end if;
      Set X (V.Shape, Get X (V.Shape) + V.Dx);
      Set Y (V.Shape, Get Y (V.Shape) + V.Dy);
   end Iterate;
begin
   loop
      for B of Balls loop
        Iterate (B);
      end loop;
      delay 0.001;
   end loop;
end Main;
```

Copyright © Audus 3

```
(Shape => New_Circle Checks if a value is within an range Dx => (Random (Seed) + 0.5 then 1.0 else -1.0),
     Dy => (Random (Seed) * 0.05 \pm 0.02) * (if Random (Seed) > 0.5 \pm 0.02);
procedure Iterate (V : in out Ball Type) is
begin
   if Get X (V.Shape) not in -100.0 .. 100.0 then
      V.Dx := -V.Dx;
   end if;
  if Get Y (V.Shape) not in -100.0 .. 100.0 then
      V.Dy := -V.Dy;
   end if;
   Set X (V.Shape, Get X (V.Shape) + V.Dx);
  Set Y (V.Shape, Get Y (V.Shape) + V.Dy);
end Iterate;
```

Copyright © Audustia

```
procedure Iterate (V : in out Ball Type) is
begin
   if Get X (V.Shape) not in -100.0 .. 100.0 then
     V.Dx := -V.Dx;
                                     Moves using a vector
   end if;
  if Get_Y (V.Shape) not in -100.0 .. 100.0 then
     V.Dy := -V.Dy;
   end if;
Set X (V.Shape, Get X (V.Shape) + V.Dx);
  Set Y (V.Snape, Get Y (V.Snape) + V.Dy);
end Iterate;
```

Copyright © Audustia

```
with Ada.Numerics.Float Random; use Ada.Numerics.Float Random;
                                     Get access to random functions
   Seed : Generator;
        => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0)
```

```
with Ada. Numerics. Float Random; use Ada. Numerics. Float Random;
   Seed : Generator; )
                            Initializes a generator to the default
        => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0),
```

Copyright © A

```
with Ada. Numerics. Float Random; use Ada. Numerics. Float Random;
   Seed : Generator;
        => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0),
                         (\text{Seed}) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0)));
  procedure Iterate (V : in out Ball Type) is
     if Get_X (V.Shape) not in -100.... 100.0 then
                               Value in [0.0 .. 1.0]
```

Copyright © A.

```
with Ada. Numerics. Float Random; use Ada. Numerics. Float Random;
   Seed : Generator;
          => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0),
                            (\text{Seed}) \times 0.05 + 0.02) \times (\text{if } \text{Random } (\text{Seed}) > 0.5 \text{ then } 1.0 \text{ else } -1.0)));
   procedure Iterate (V : in ou Ball Type) is
      if Get_X (V.Shape) not in -100.0 . 100.0 then
                                   Value in [0.0 .. 0.05]
```

```
with Ada. Numerics. Float Random; use Ada. Numerics. Float Random;
   Seed : Generator;
        => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0),
                                 * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0)));
  procedure Iterate (V : in out Ball Type) is
     if Get X (V.Shape) not in -100.0 . 100.0 then
                              Value in [0.02 .. 0.07]
```

```
with Ada. Numerics. Float Random; use Ada. Numerics. Float Random;
   Seed : Generator;
        => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0)
                                  0.05 + 0.02) * (if Rando
                                                            (Seed) > 0.5 then 1.0 else -1.0));
  procedure Iterate (V : in out Ball Type) is
     if Get X (V.Shape) not in -100.0 . 100.0 then
                              Value in [0.02 .. 0.07]
                                                                   Value in [0.0 .. 1.0]
```

```
with Ada. Numerics. Float Random; use Ada. Numerics. Float Random;
   Seed : Generator;
        => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0)
   procedure Iterate (V : in out Ball Type) is
     if Get X (V.Shape) not in -100.0 . 100.0 then
                                                                   Value in {-1.0, 1.0}
                              Value in [0.02 .. 0.07]
```

Copyright © A...... Slide: 12

```
with Ada. Numerics. Float Random; use Ada. Numerics. Float Random;
   Seed : Generator;
        => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0)
              => (Random (Seed) * 0.05 + 0.02) *
                                                                eed) > 0.5 then 1.0 else -1.0));
     if Get X (V.Shape) not in -100.0 .. 100.0 then
                              Value in {[-0.07 .. -0.02], [0.02 .. 0.07]}
```

```
Declares an array type
type Ball Array is array (Integer range <>) of Ball Type;
Balls : Ball Array (1 .. 10) :=
   (others =>
      (Shape => New Circle (0.0, 0.0, 10.0, Blue),
            => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0),
            => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0)));
   for B of Balls loop
      Iterate (B);
   end loop;
```

Copyright © A..... Slide: 14

```
type Ball_Type is recoThe array in indexed by Integer
type Ball Array is array (Integer range <>) of Ball Type;
Balls : Ball Array (1 .. 10) :=
  (others =>
     (Shape => New Circle (0.0, 0.0, 10.0, Blue),
     Dx = (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0),
           => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0)));
  for B of Balls loop
     Iterate (B);
  end loop;
```

```
type Ball_Type is The array contains a number of elements to be specified at declaration
type Ball Array is array (Integer range <>) of Ball Type;
Balls : Ball Array (1 .. 10) :=
  (others =>
     (Shape => New Circle (0.0, 0.0, 10.0, Blue),
     Dx = (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0),
           => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0)));
  for B of Balls loop
     Iterate (B);
  end loop;
```

```
The array contains instances of Ball_Type
type Ball Array is array (Integer range <> of Ball Type;
Balls : Ball Array (1 .. 10) :=
  (others =>
     (Shape => New Circle (0.0, 0.0, 10.0, Blue),
     Dx = (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0),
           => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0)));
  for B of Balls loop
     Iterate (B);
  end loop;
```

Copyright © A...... Slide: 17

```
type Ball Array is array (Integer range <>) of Ball Type;
                                       Declare an array of type Ball_Array
Balls : Ball Array (1 .. 10)
(ethers =>
     (Shape => New Circle (0.0, 0.0, 10.0, Blue),
           => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0),
           => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0)));
   for B of Balls loop
      Iterate (B);
   end loop;
```

```
type Ball Array is array (Integer range <>) of Ball Type;
                                 Gives boundaries between 1 and 10 (10 elements)
Balls : Ball Array (1 .. 10)
  (others =>
     (Shape => New Circle (0.0, 0.0, 10.0, Blue),
           => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0),
           => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0)));
  for B of Balls loop
     Iterate (B);
  end loop;
```

```
Initializes the array through an aggreate
type Ball Array is array (Integer range <>) of Ball Type;
Balls : Ball Array (1 .. 10) :=
  (others =>
     (Shape => New Circle (0.0, 0.0, 10.0, Blue),
           => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0),
           => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0))
procedure Iterate (V : in out Ball Type) is
  for B of Balls loop
     Iterate (B);
  end loop;
```

```
type Ball Array is array (Integer range <>) of Ball Type;
             Every value is initialized through the same expression
Balls . Pall Array (1 .. 10) :=
  others
    (Shape => New Circle (0.0, 0.0, 10.0, Blue),
           => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0),
           => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0)));
      Dv
  for B of Balls loop
     Iterate (B);
  end loop;
```

Copyright © A...... Slide: 21

```
type Ball Array is array (Integer range <>) of Ball Type;
                          Initializes each component through the same record aggregate
Balls : Ball Array (1 .. 10) :=
  (others ->
     (Shape => New Circle (0.0, 0.0, 10.0, Blue),
           => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0),
           => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0))
procedure Iterate (V : in out
  for B of Balls loop
     Iterate (B);
  end loop;
```

```
type Ball Array is array (Integer range <>) of Ball_Type;
                              These will be recomputed for each element!
Balls : Ball Array (1 .. 10) :=
  (others =>
     (Shape => New Circle (0.0, 0.0, 10.0, Blue)
  Dx => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0).
           => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0)));
      Dv
  for B of Balls loop
     Iterate (B);
  end loop;
```

Copyright © A...... Slide: 23

```
type Ball Array is array (Integer range <>) of Ball Type;
Balls : Ball Array (1 .. 10) :=
  (others =>
     (Shape => New Circle (0.0, 0.0, 10.0, Blue),
     Dx = (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0),
           => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0)));
                              Iterate through each element of Balls
  for B of Balls loop
     Iterate (B);
  end loop;
```

Copyright © A.





? Identify the Errors

```
with Display;
                                        use Display;
with Display.Basic;
                                        use Display.Basic;
procedure Main is
  type Ball Type is record
      Shape : Shape Id;
     X, Y : Float;
     Step : Float;
  end record;
  type Ball List is array (Integer) of Ball Type;
  List : Ball List (1 .. 10) :=
     ((Shape => New Circle (0.0, 0.0, 5.0, Blue),
      X => 0.0
      Y => 0.0,
      Step => 1.0));
begin
  loop
     for B in List loop
         B.X := B.X + B.Step;
     end loop;
     delay 0.001;
  end loop;
end Main;
```

```
with Display;
                                         use Display;
with Display.Basic;
                                         use Display.Basic;
procedure Main is
  type Ball Type is record
      Shape : Shape Id;
     X, Y : Float;
      Step : Float;
  end record;
  type Ball List is array (Integer) of Ball Type;
  List : Ball List (1 .. 10) :=
    ((Sbape => New Circle (0.0, 0.0, 5.0, Blue),
          => 0.0,
       Y => 0.0,
       Step \Rightarrow 1.0));
begin
  loop
      for B in List loop
         B.X := B.X + B.Step;
      end loop;
      delay 0.001;
  end loop;
end Main;
```

```
with Display;
                                         use Display;
with Display.Basic;
                                         use Display.Basic;
procedure Main is
   type Ball Type is record
                                                           "range <>" needs to be
      Shape : Shape Id;
                                                           specified for an
     X, Y : Float;
      Step : Float;
                                                           unconstrained array
  end record;
  type Ball List is array (Integer) of Ball Type;
  List : Ball List (1 .. 10) :=
     ((Shape => New Circle (0.0, 0.0, 5.0, Blue),
       X => 0.0
       Y => 0.0,
       Step \Rightarrow 1.0));
begin
  loop
      for B in List loop
         B.X := B.X + B.Step;
      end loop;
      delay 0.001;
  end loop;
end Main;
```

```
with Display;
                                                             use Display;
                   with Display.Basic;
                                                             use Display.Basic;
                   procedure Main is
                      type Ball Type is record
                         Shape : Shape Id;
                         X, Y : Float;
                         Step : Float;
                      end record;
missing to specify
                      type Ball List is array (Integer range <>) of Ball Type;
that a value is given
                      List : Ball List (1 .. 10) :=
                        ((Shape => New Circle (0.0, 0.0, 5.0, Blue),
                                => 0.0,
                                => 0.0,
                          Step \Rightarrow 1.0));
                   begin
                      loop
                         for B in List loop
                            B.X := B.X + B.Step;
                         end loop;
                         delay 0.001;
                      end loop;
                   end Main;
```

"others =>" is

to all objects

```
with Display;
                                                           use Display;
                  with Display.Basic;
                                                           use Display.Basic;
                  procedure Main is
                     type Ball Type is record
                        Shape : Shape Id;
                        X, Y : Float;
                        Step : Float;
                     end record;
                     type Ball List is array (Integer range <>) of Ball Type;
                     List : Ball List (1 .. 10) :=
                        (others => (Shape => New Circle (0.0, 0.0, 5.0, Blue),
                                    X => 0.0
"of" is the notation
                                    Y => 0.0,
                                    Step \Rightarrow 1.0));
                  begin
                        for B in List loop
                            B.X := B.X + B.Step;
                        end loop;
                        delay 0.001;
                     end loop;
                  end Main;
```

to iterate over the

elements

```
with Display;
                                        use Display;
with Display.Basic;
                                        use Display.Basic;
procedure Main is
  type Ball Type is record
      Shape : Shape Id;
     X, Y : Float;
      Step : Float;
  end record;
  type Ball List is array (Integer range <>) of Ball Type;
  List : Ball List (1 .. 10) :=
     (others => (Shape => New Circle (0.0, 0.0, 5.0, Blue),
                 X => 0.0
                 Y => 0.0,
                 Step \Rightarrow 1.0);
begin
  loop
      for B of List loop
         B.X := B.X + B.Step;
      end loop;
      delay 0.001;
  end loop;
end Main;
```





university.adacore.com

Copyright © AdaCore