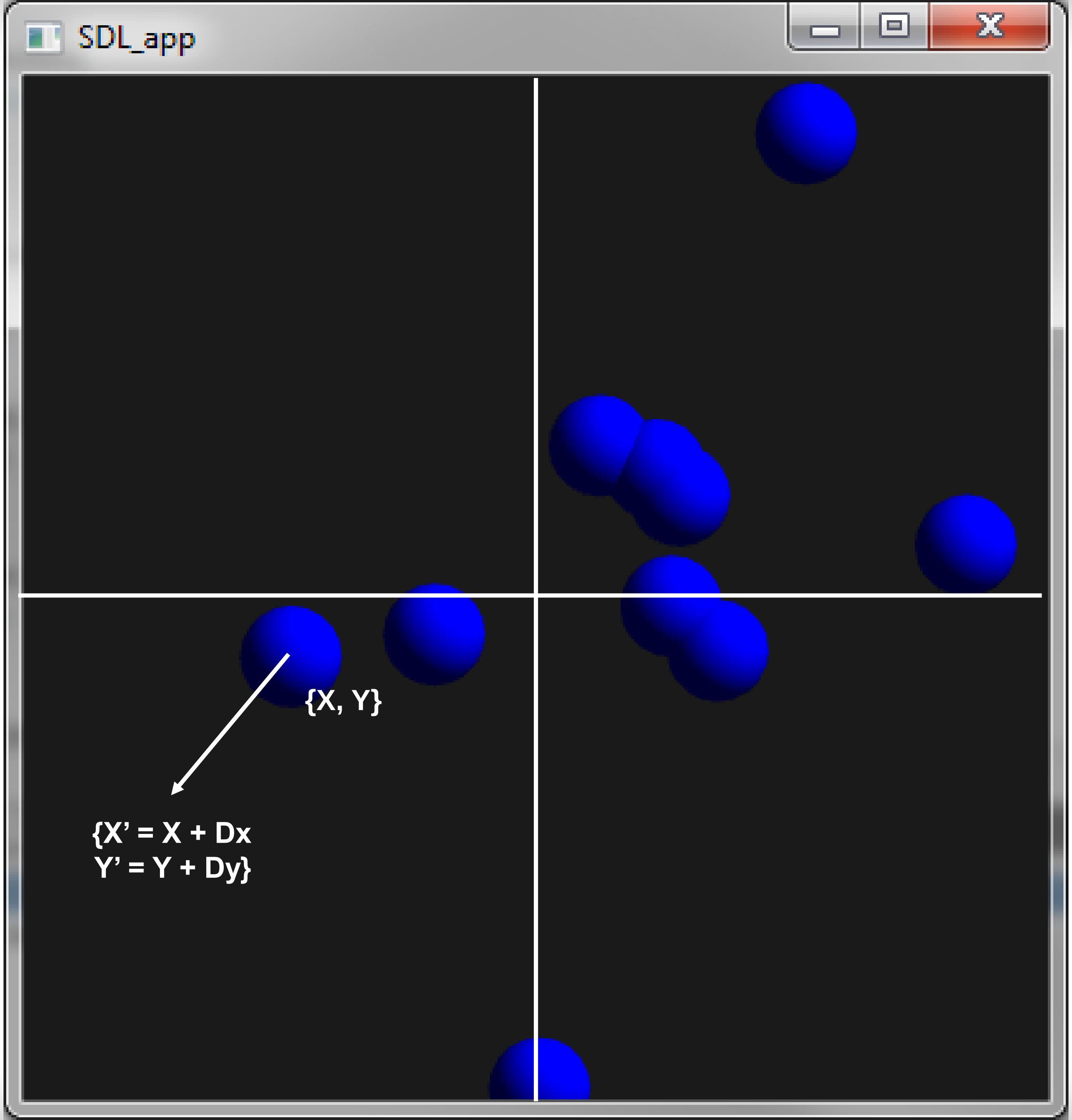
Ten Bouncing Balls

Presented by Quentin Ochem

university.adacore.com


SDL_app
{X, Y}
{X' = X + Dx
Y' = Y + Dy}

```
with Display;                     use Display;
with Display.Basic;               use 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),
         Dy    => (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;
```

```
with Display;                       use Display;
with Display.Basic;                 use 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),
         Dy     => (Random (Seed) * 0.05 + 0.02) * (if Random (Seed) > 0.5 then 1.0 else -1.0)));
```

Checks if a value is within an range

```
   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;
```

```
with Display;                      use Display;
with Display.Basic;                use 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),
         Dy     => (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;
```

Moves using a vector

```
with Display;                       use Display;
with Display.Basic;                 use 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),
         Dy    => (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;
```

Get access to random functions

```
with Display;                     use Display;
with Display.Basic;               use 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),
         Dy     => (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;
```

Initializes a generator to the default

```
with Display;                      use Display;
with Display.Basic;                use 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),
         Dy     => (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;
```

Value in [0.0 .. 1.0]

```
with Display;                    use Display;
with Display.Basic;              use 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),
         Dy     => (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;
```

Value in [0.0 .. 0.05]

```
with Display;                   use Display;
with Display.Basic;             use 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),
         Dy    => (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;
```

Value in [0.02 .. 0.07]

```
with Display;                   use Display;
with Display.Basic;             use 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),
         Dy    => (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;
```

Value in [0.02 .. 0.07]

Value in [0.0 .. 1.0]

```
with Display;                     use Display;
with Display.Basic;               use 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),
         Dy     => (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;
```

Value in [0.02 .. 0.07]

Value in {-1.0, 1.0}

```
with Display;                    use Display;
with Display.Basic;              use 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),
         Dy     => (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;
```

Value in {[-0.07 .. -0.02], [0.02 .. 0.07]}

Declares an array type

```
with Display;                        use Display;
with Display.Basic;                  use 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),
         Dy     => (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;
```

The array in indexed by Integer

```
with Display;                    use Display;
with Display.Basic;              use 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),
         Dy    => (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;
```

The array contains a number of elements to be specified at declaration

```
with Display;                      use Display;
with Display.Basic;                use 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),
         Dy    => (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;
```

```
with Display;                    use Display;
with Display.Basic;              use 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),
         Dy     => (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;
```

The array contains instances of Ball_Type

```
with Display;                      use Display;
with Display.Basic;                use 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),
         Dy    => (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;
```

Declare an array of type Ball_Array

```
with Display;                       use Display;
with Display.Basic;                 use 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),
         Dy     => (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;
```

Gives boundaries between 1 and 10 (10 elements)

Initializes the array through an aggregate

```
with Display;                        use Display;
with Display.Basic;                  use 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),
         Dy    => (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;
```

```
with Display;                      use Display;
with Display.Basic;                use 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 : Generat
```

Every value is initialized through the same expression

```
   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),
        Dy    => (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;
```

```
with Display;                        use Display;
with Display.Basic;                  use 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),
        Dy    => (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;
```

Initializes each component through the same record aggregate

```
with Display;                      use Display;
with Display.Basic;                use 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),
         Dy    => (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;
```

These will be recomputed for each element!

```
with Display;                   use Display;
with Display.Basic;             use 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),
         Dy    => (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;
```

Iterate through each element of Balls

Quiz

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) :=
     ((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) :=
     ((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;
```

“range <>” needs to be specified for an unconstrained array

```
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) :=
     ((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;
```

“others =>” is missing to specify that a value is given 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,
                 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;
```

"of" is the notation 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  => 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