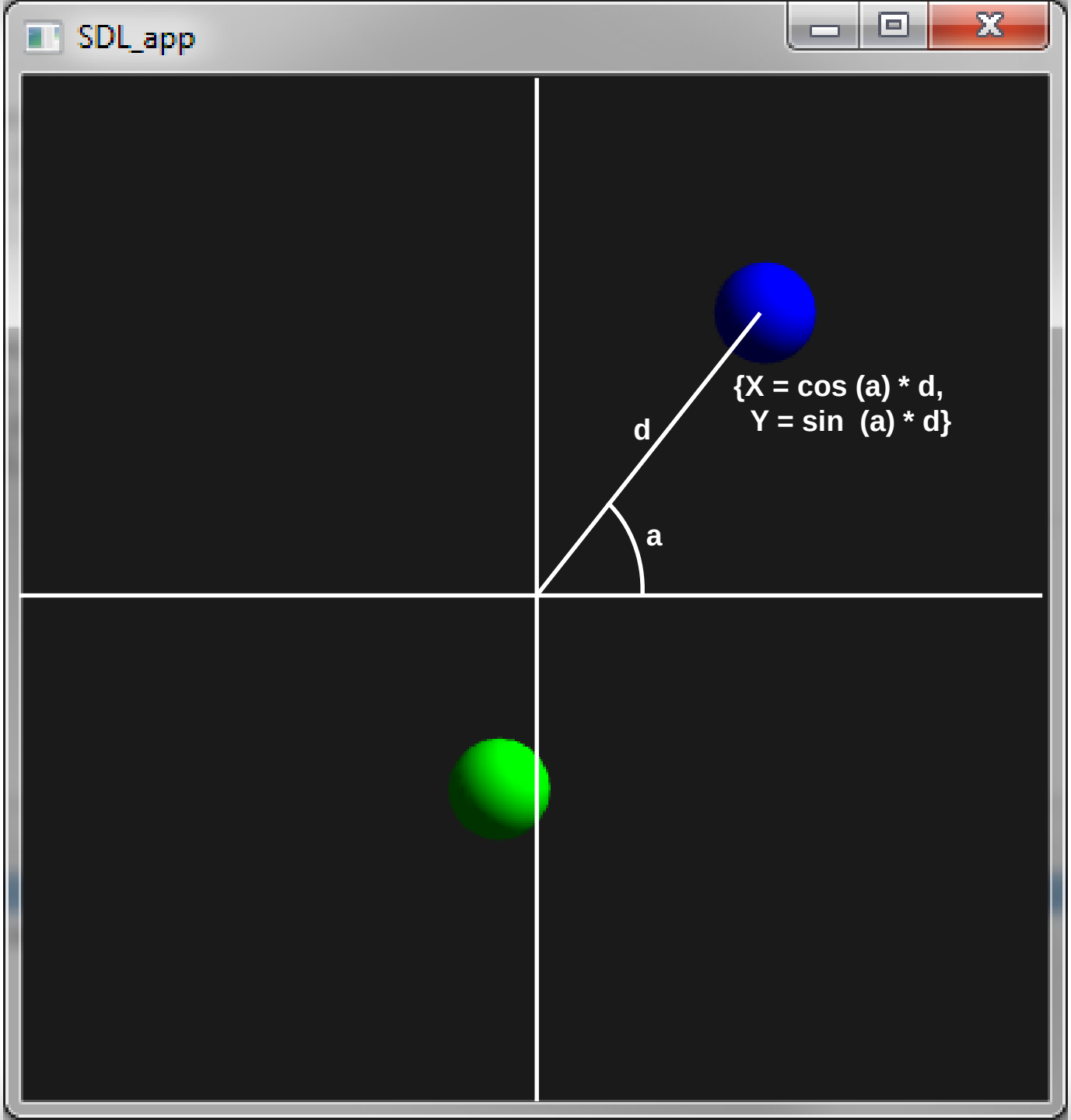
Rotating Balls

Presented by Quentin Ochem

university.adacore.com



```
with Display;                           use Display;
with Display.Basic;                     use Display.Basic;
with Ada.Numerics.Elementary_Functions; use Ada.Numerics.Elementary_Functions;

procedure Main is
   type Ball_Type is record
      Shape       : Shape_Id;
      Angle_Speed : Float;
      Angle       : Float;
      Distance    : Float;
   end record;

   procedure Iterate (V : in out Ball_Type) is
   begin
      V.Angle := V.Angle + V.Angle_Speed;
      Set_X (V.Shape, Cos (V.Angle) * V.Distance);
      Set_Y (V.Shape, Sin (V.Angle) * V.Distance);
   end Iterate;

   B1 : Ball_Type :=
     (Shape       => New_Circle (0.0, 0.0, 10.0, Blue),
      Angle_Speed => 0.001,
      Angle       => 0.0,
      Distance    => 70.0);

   B2 : Ball_Type :=
     (Shape       => New_Circle (0.0, 0.0, 10.0, Green),
      Angle_Speed => -0.002,
      Angle       => 0.0,
      Distance    => 40.0);
begin
   loop
      Iterate (B1);
      Iterate (B2);

      delay 0.001;
   end loop;
end Main;
```

```
with Display;                                use Display;
with Display.Basic;                          use Display.Basic;
with Ada.Numerics.Elementary_Functions;      use Ada.Numerics.Elementary_Functions;

procedure Main is
   type Ball_Type is record
      Shape       : Shape_Id;
      Angle_Speed : Float;
      Angle       : Float;
      Distance    : Float;
   end record;

   procedure Iterate (V : in out Ball_Type) is
   begin
      V.Angle := V.Angle + V.Angle_Speed;
      Set_X (V.Shape, Cos (V.Angle) * V.Distance);
      Set_Y (V.Shape, Sin (V.Angle) * V.Distance);
   end Iterate;

   B1 : Ball_Type :=
     (Shape       => New_Circle (0.0, 0.0, 10.0, Blue),
      Angle_Speed => 0.001,
      Angle       => 0.0,
      Distance    => 70.0);

   B2 : Ball_Type :=
     (Shape       => New_Circle (0.0, 0.0, 10.0, Green),
      Angle_Speed => -0.002,
      Angle       => 0.0,
      Distance    => 40.0);
begin
   loop
      Iterate (B1);
      Iterate (B2);

      delay 0.001;
   end loop;
end Main;
```

Gives access to sin / cos

```
with Display;                           use Display;
with Display.Basic;                     use Display.Basic;
with Ada.Numerics.Elementary_Functions; use Ada.Numerics.Elementary_Functions;

procedure Main is
   type Ball_Type is record
      Shape       : Shape_Id;
      Angle_Speed : Float;
      Angle       : Float;
      Distance    : Float;
   end record;

   procedure Iterate (V : in out Ball_Type) is
   begin
      V.Angle := V.Angle + V.Angle_Speed;
      Set_X (V.Shape, Cos (V.Angle) * V.Distance);
      Set_Y (V.Shape, Sin (V.Angle) * V.Distance);
   end Iterate;

   B1 : Ball_Type :=
     (Shape       => New_Circle (0.0, 0.0, 10.0, Blue),
      Angle_Speed => 0.001,
      Angle       => 0.0,
      Distance    => 70.0);

   B2 : Ball_Type :=
     (Shape       => New_Circle (0.0, 0.0, 10.0, Green),
      Angle_Speed => -0.002,
      Angle       => 0.0,
      Distance    => 40.0);
begin
   loop
      Iterate (B1);
      Iterate (B2);

      delay 0.001;
   end loop;
end Main;
```

Declares a structure of 4 fields

```
with Display;                           use Display;
with Display.Basic;                     use Display.Basic;
with Ada.Numerics.Elementary_Functions; use Ada.Numerics.Elementary_Functions;

procedure Main is
   type Ball_Type is record
      Shape       : Shape_Id;
      Angle_Speed : Float;
      Angle       : Float;
      Distance    : Float;
   end record;

   procedure Iterate (V : in out Ball_Type) is
   begin
      V.Angle := V.Angle + V.Angle_Speed;
      Set_X (V.Shape, Cos (V.Angle) * V.Distance);
      Set_Y (V.Shape, Sin (V.Angle) * V.Distance);
   end Iterate;

   B1 : Ball_Type :=
     (Shape       => New_Circle (0.0, 0.0, 10.0, Blue),
      Angle_Speed => 0.001,
      Angle       => 0.0,
      Distance    => 70.0);

   B2 : Ball_Type :=
     (Shape       => New_Circle (0.0, 0.0, 10.0, Green),
      Angle_Speed => -0.002,
      Angle       => 0.0,
      Distance    => 40.0);
begin
   loop
      Iterate (B1);
      Iterate (B2);

      delay 0.001;
   end loop;
end Main;
```

Declares a nested subprogram

```
with Display;                           use Display;
with Display.Basic;                     use Display.Basic;
with Ada.Numerics.Elementary_Functions; use Ada.Numerics.Elementary_Functions;

procedure Main is
   type Ball_Type is record
      Shape       : Shape_Id;
      Angle_Speed : Float;
      Angle       : Float;
      Distance    : Float;
   end record;

   procedure Iterate (V : in out Ball_Type) is
   begin
      V.Angle := V.Angle + V.Angle_Speed;
      Set_X (V.Shape, Cos (V.Angle) * V.Distance);
      Set_Y (V.Shape, Sin (V.Angle) * V.Distance);
   end Iterate;

   B1 : Ball_Type :=
     (Shape       => New_Circle (0.0, 0.0, 10.0, Blue),
      Angle_Speed => 0.001,
      Angle       => 0.0,
      Distance    => 70.0);

   B2 : Ball_Type :=
     (Shape       => New_Circle (0.0, 0.0, 10.0, Green),
      Angle_Speed => -0.002,
      Angle       => 0.0,
      Distance    => 40.0);
begin
   loop
      Iterate (B1);
      Iterate (B2);

      delay 0.001;
   end loop;
end Main;
```

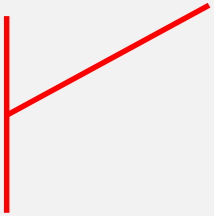
The parameter value may be modified

```
with Display;                            use Display;
with Display.Basic;                      use Display.Basic;
with Ada.Numerics.Elementary_Functions; use Ada.Numerics.Elementary_Functions;

procedure Main is
   type Ball_Type is record
      Shape       : Shape_Id;
      Angle_Speed : Float;
      Angle       : Float;
      Distance    : Flo
   end record;

   procedure Iterate (V : in out Ball_Type) is
   begin
      V.Angle := V.Angle + V.Angle_Speed;
      Set_X (V.Shape, Cos (V.Angle) * V.Distance);
      Set_Y (V.Shape, Sin (V.Angle) * V.Distance);
   end Iterate;

   B1 : Ball_Type :=
     (Shape       => New_Circle (0.0, 0.0, 10.0, Blue),
      Angle_Speed => 0.001,
      Angle       => 0.0,
      Distance    => 70.0);

   B2 : Ball_Type :=
     (Shape       => New_Circle (0.0, 0.0, 10.0, Green),
      Angle_Speed => -0.002,
      Angle       => 0.0,
      Distance    => 40.0);
begin
   loop
      Iterate (B1);
      Iterate (B2);

      delay 0.001;
   end loop;
end Main;
```

Access to the field Angle of the parameter V

```
with Display;                           use Display;
with Display.Basic;                     use Display.Basic;
with Ada.Numerics.Elementary_Functions; use Ada.Numerics.Elementary_Functions;

procedure Main is
   type Ball_Type is record
      Shape       : Shape_Id;
      Angle_Speed : Float;
      Angle       : Float;
      Distance    : Float;
   end record;

   procedure Iterate (V : in out Ball_Type) is
   begin
      V.Angle := V.Angle + V.Angle_Speed;
      Set_X (V.Shape, Cos (V.Angle) * V.Distance);
      Set_Y (V.Shape, Sin (V.Angle) * V.Distance);
   end Iterate;

   B1 : Ball_Type :=
     (Shape       => New_Circle (0.0, 0.0, 10.0, Blue),
      Angle_Speed => 0.001,
      Angle       => 0.0,
      Distance    => 70.0);

   B2 : Ball_Type :=
     (Shape       => New_Circle (0.0, 0.0, 10.0, Green),
      Angle_Speed => -0.002,
      Angle       => 0.0,
      Distance    => 40.0);
begin
   loop
      Iterate (B1);
      Iterate (B2);

      delay 0.001;
   end loop;
end Main;
```

Gives a value by aggregate to an object



```
with Display;                           use Display;
with Display.Basic;                     use Display.Basic;
with Ada.Numerics.Elementary_Functions; use Ada.Numerics.Elementary_Functions;

procedure Main is
   type Ball_Type is record
      Shape       : Shape_Id;
      Angle_Speed : Float;
      Angle       : Float;
      Distance    : Float;
   end record;

   procedure Iterate (V : in out Ball_Type) is
   begin
      V.Angle := V.Angle + V.Angle_Speed;
      Set_X (V.Shape, Cos (V.Angle) * V.Distance);
      Set_Y (V.Shape, Sin (V.Angle) * V.Distance);
   end Iterate;

   B1 : Ball_Type :=
     (Shape       => New_Circle (0.0, 0.0, 10.0, Blue),
      Angle_Speed => 0.001,
      Angle       => 0.0,
      Distance    => 70.0);

   B2 : Ball_Type :=
     (Shape       => New_Circle (0.0, 0.0, 10.0, Green),
      Angle_Speed => -0.002,
      Angle       => 0.0,
      Distance    => 40.0);
begin
   loop
      Iterate (B1);
      Iterate (B2);

      delay 0.001;
   end loop;
end Main;
```

Calls the nested subprogram on the two objects

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 Ball_Type;

   procedure Iterate (V : Ball_Type) is
   begin
      if V.X > 100.0 then
         V.Step := -1;
      else
         V.Step := 1;
      end if;

      V.X := V.X + V.Step;
   end Iterate;

   B : Ball_Type :=
     (Shape => New_Circle (0.0, 0.0, 10.0, Blue),
      Step  => 1.0);
begin
   loop
      Iterate (B);

      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 Ball_Type;

   procedure Iterate (V : Ball_Type) is
   begin
      if V.X > 100.0 then
         V.Step := -1;
      else
         V.Step := 1;
      end if;

      V.X := V.X + V.Step;
   end Iterate;

   B : Ball_Type :=
     (Shape => New_Circle (0.0, 0.0, 10.0, Blue),
      Step  => 1.0);
begin
   loop
      Iterate (B);

      delay 0.001;
   end loop;
end Main;
```

‘end record’ closes a record

```
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 Ball_Type;

   procedure Iterate (V : Ball_Type) is
   begin
      if V.X > 100.0 then
         V.Step := -1;
      else
         V.Step := 1;
      end if;

      V.X := V.X + V.Step;
   end Iterate;

   B : Ball_Type :=
     (Shape => New_Circle (0.0, 0.0, 10.0, Blue),
      Step  => 1.0);
begin
   loop
      Iterate (B);

      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;

   procedure Iterate (V : Ball_Type) is
   begin
      if V.X > 100.0 then
         V.Step := -1;
      else
         V.Step := 1;
      end if;

      V.X := V.X + V.Step;
   end Iterate;

   B : Ball_Type :=
     (Shape => New_Circle (0.0, 0.0, 10.0, Blue),
      Step  => 1.0);
begin
   loop
      Iterate (B);

      delay 0.001;
   end loop;
end Main;
```

Mode should say "in out" for modifying parameter

```
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;

   procedure Iterate (V : in out Ball_Type) is
   begin
      if V.X > 100.0 then
         V.Step := -1;
      else
         V.Step := 1;
      end if;

      V.X := V.X + V.Step;
   end Iterate;

   B : Ball_Type :=
     (Shape => New_Circle (0.0, 0.0, 10.0, Blue),
      Step  => 1.0);
begin
   loop
      Iterate (B);

      delay 0.001;
   end loop;
end Main;
```

Floating-point should be written -1.0

```
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;

   procedure Iterate (V : in out Ball_Type) is
   begin
      if V.X > 100.0 then
         V.Step := -1.0;
      else
         V.Step := 1.0;
      end if;

      V.X := V.X + V.Step;
   end Iterate;

   B : Ball_Type :=
      (Shape => New_Circle (0.0, 0.0, 10.0, Blue),
       Step  => 1.0);
begin
   loop
      Iterate (B);

      delay 0.001;
   end loop;
end Main;
```

Values for X and Y are missing

```
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;

   procedure Iterate (V : in out Ball_Type) is
   begin
      if V.X > 100.0 then
         V.Step := -1.0;
      else
         V.Step := 1.0;
      end if;

      V.X := V.X + V.Step;
   end Iterate;

   B : Ball_Type :=
     (Shape => New_Circle (0.0, 0.0, 10.0, Blue),
      Step  => 1.0, X => 0.0, Y => 0.0);
begin
   loop
      Iterate (B);

      delay 0.001;
   end loop;
end Main;
```

university.adacore.com