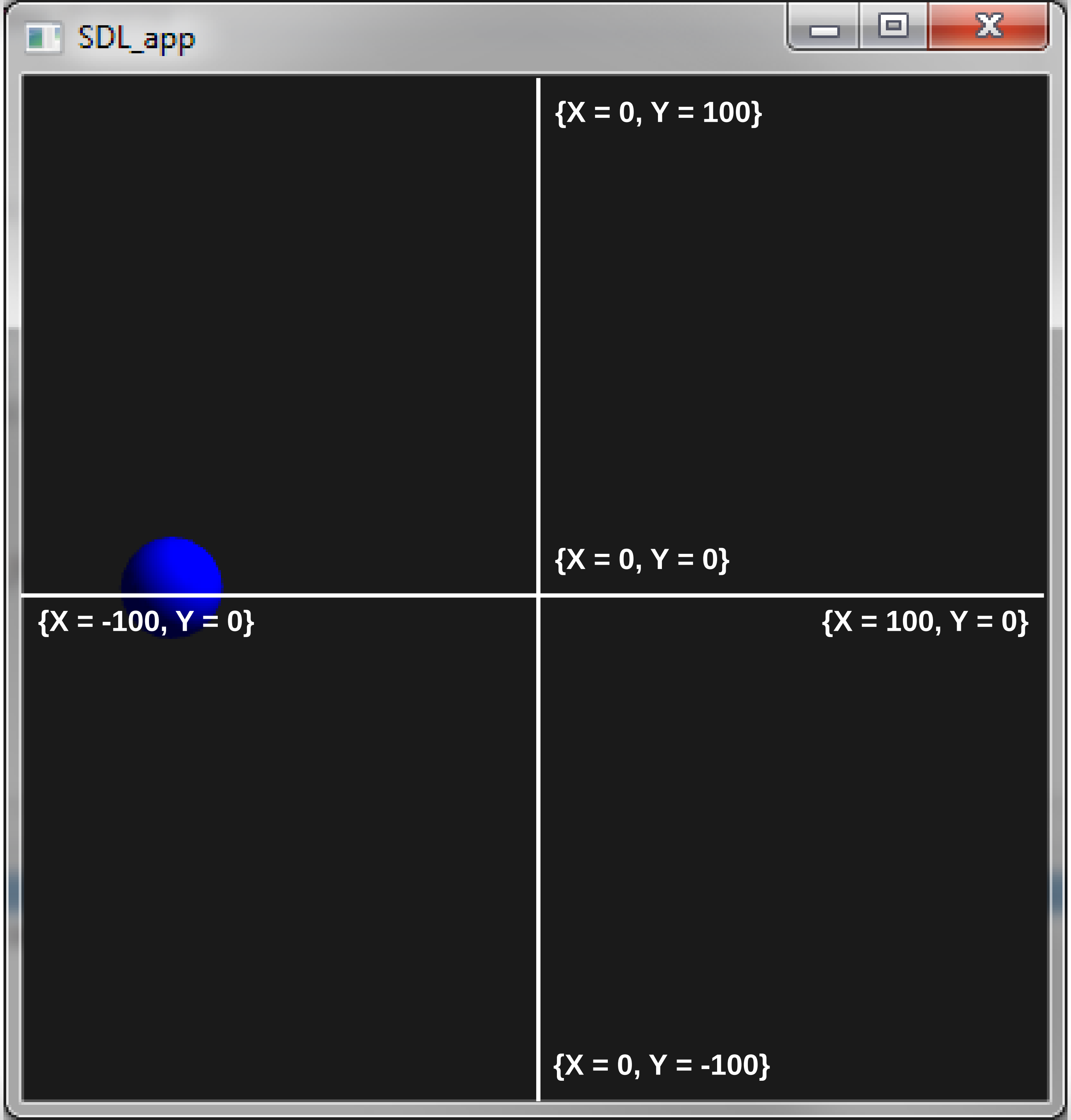
A Bouncing Ball

Presented by Quentin Ochem

University.adacore.com


SDL_app
{X = 0, Y = 100}
{X = 0, Y = 0}
{X = -100, Y = 0}
{X = 100, Y = 0}
{X = 0, Y = -100}

```
with Display;       use Display;
with Display.Basic; use Display.Basic;

procedure Main is
   Ball : Shape_Id := New_Circle
     (X      => 0.0,
      Y      => 0.0,
      Radius => 10.0,
      Color  => Blue);
   Step : Float := 0.05;
begin
   loop
      if Get_X (Ball) > 100.0 then
         Step := -0.05;
      elsif Get_X (Ball) < -100.0 then
         Step := 0.05;
      end if;

      Set_X (Ball, Get_X (Ball) + Step);

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

References to the graphical library

```
with Display;       use Display;
with Display.Basic; use Display.Basic;

procedure Main is
   Ball : Shape_Id := New_Circle
     (X      => 0.0,
      Y      => 0.0,
      Radius => 10.0,
      Color  => Blue);
   Step : Float := 0.05;
begin
   loop
      if Get_X (Ball) > 100.0 then
         Step := -0.05;
      elsif Get_X (Ball) < -100.0 then
         Step := 0.05;
      end if;

      Set_X (Ball, Get_X (Ball) + Step);

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

Declaration of a shape object with an initial value

```
with Display;       use Display;
with Display.Basic; use Display.Basic;

procedure Main is
   Ball : Shape_Id := New_Circle
     (X      => 0.0,
      Y      => 0.0,
      Radius => 10.0,
      Color  => Blue);
   Step : Float := 0.05;
begin
   loop
      if Get_X (Ball) > 100.0 then
         Step := -0.05;
      elsif Get_X (Ball) < -100.0 then
         Step := 0.05;
      end if;

      Set_X (Ball, Get_X (Ball) + Step);

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

Parameter value given by name

```
with Display;       use Display;
with Display.Basic; use Display.Basic;

procedure Main is
   Ball : Shape_Id := New_Circle
     (X      => 0.0,
      Y      => 0.0,
      Radius => 10.0,
      Color  => Blue);
   Step : Float := 0.05;
begin
   loop
      if Get_X (Ball) > 100.0 then
         Step := -0.05;
      elsif Get_X (Ball) < -100.0 then
         Step := 0.05;
      end if;

      Set_X (Ball, Get_X (Ball) + Step);

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

X is a float, so it needs a floating point literal (0.0) not integer (0)

```
with Display;       use Display;
with Display.Basic; use Display.Basic;

procedure Main is
   Ball : Shape_Id := New_Circle
      (X      => 0.0,
       Y      => 0.0,
       Radius => 10.0,
       Color  => Blue);
   Step : Float := 0.05;
begin
   loop
      if Get_X (Ball) > 100.0 then
         Step := -0.05;
      elsif Get_X (Ball) < -100.0 then
         Step := 0.05;
      end if;

      Set_X (Ball, Get_X (Ball) + Step);

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

```
with Display;       use Display;
with Display.Basic; use Display.Basic;

procedure Main is
   Ball : Shape_Id := New_Circle
     (X      => 0.0,
      Y      => 0.0,
      Radius => 10.0,
      Color  => Blue);
   Step : Float := 0.05;
begin
   loop
      if Get_X (Ball) > 100.0 then
         Step := -0.05;
      elsif Get_X (Ball) < -100.0 then
         Step := 0.05;
      end if;

      Set_X (Ball, Get_X (Ball) + Step);

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

Infinite loop

```
with Display;       use Display;
with Display.Basic; use Display.Basic;

procedure Main is
   Ball : Shape_Id := New_Circle
     (X      => 0.0,
      Y      => 0.0,
      Radius => 10.0,
      Color  => Blue);
   Step : Float := 0.05;
begin
   loop
      if Get_X (Ball) > 100.0 then
         Step := -0.05;
      elsif Get_X (Ball) < -100.0 then
         Step := 0.05;
      end if;

      Set_X (Ball, Get_X (Ball) + Step);

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

Wait for 1 milisecond

If the ball gets out of the boundaries, then invert the step

```ada
with Display;       use Display;
with Display.Basic; use Display.Basic;

procedure Main is
   Ball : Shape_Id := New_Circle
     (X      => 0.0,
      Y      => 0.0,
      Radius => 10.0,
      Color  => Blue);
   Step : Float := 0.05;
begin
   loop
      if Get_X (Ball) > 100.0 then
         Step := -0.05;
      elsif Get_X (Ball) < -100.0 then
         Step := 0.05;
      end if;

      Set_X (Ball, Get_X (Ball) + Step);

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

```
with Display;       use Display;
with Display.Basic; use Display.Basic;

procedure Main is
   Ball : Shape_Id := New_Circle
     (X      => 0.0,
      Y      => 0.0,
      Radius => 10.0,
      Color  => Blue);
   Step : Float := 0.05;
begin
   loop
      if Get_X (Ball) > 100.0 then
         Step := -0.05;
      elsif Get_X (Ball) < -100.0 then
         Step := 0.05;
      end if;

      Set_X (Ball, Get_X (Ball) + Step);

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

Move the X position of the ball

Quiz

Identify the Errors

```
with Display;       use Display;
with Display.Basic; use Display.Basic;

procedure Main is
   Shape_Id : Ball := New_Circle
     (X      => 0.0,
      Y      => 0.0,
      Radius => 10.0,
      Color  => Blue);
   Float : Step := 0.05;
begin
   loop
      if (Get_Y (Ball) > 100.0) then
         Step := -0.05;
      else if Get_Y (Ball) < -100 then
         Step := 0.05;
      end if;

      Set_Y (Ball, Get_X (Ball) + Step);

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

```
with Display;       use Display;
with Display.Basic; use Display.Basic;

procedure Main is
   Shape_Id : Ball := New_Circle
      (X      => 0.0,
       Y      => 0.0,
       Radius => 10.0,
       Color  => Blue);
   Float : Step := 0.05;
begin
   loop
      if (Get_Y (Ball) > 100.0) then
         Step := -0.05;
      else if Get_Y (Ball) < -100 then
         Step := 0.05;
      end if;

      Set_Y (Ball, Get_X (Ball) + Step);

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

Variable are declared like name : type;

```
with Display;       use Display;
with Display.Basic; use Display.Basic;

procedure Main is
   Shape_Id : Ball := New_Circle
      (X      => 0.0,
       Y      => 0.0,
       Radius => 10.0,
       Color  => Blue);
   Float : Step := 0.05;
begin
   loop
      if (Get_Y (Ball) > 100.0) then
         Step := -0.05;
      else if Get_Y (Ball) < -100 then
         Step := 0.05;
      end if;

      Set_Y (Ball, Get_X (Ball) + Step);

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

elsif introduces an alternative

```
with Display;       use Display;
with Display.Basic; use Display.Basic;

procedure Main is
   Ball : Shape_Id := New_Circle
      (X      => 0.0,
       Y      => 0.0,
       Radius => 10.0,
       Color  => Blue);
   Step : Float := 0.05;
begin
   loop
      if (Get_Y (Ball) > 100.0) then
         Step := -0.05;
      else if Get_Y (Ball) < -100 then
         Step := 0.05;
      end if;

      Set_Y (Ball, Get_X (Ball) + Step);

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

```
with Display;        use Display;
with Display.Basic; use Display.Basic;

procedure Main is
   Ball : Shape_Id := New_Circle
     (X      => 0.0,
      Y      => 0.0,
      Radius => 10.0,
      Color  => Blue);
   Step : Float := 0.05;
begin
   loop
      if (Get_Y (Ball) > 100.0) then
         Step := -0.05;
      elsif Get_Y (Ball) < -100 then
         Step := 0.05;
      end if;

      Set_Y (Ball, Get_X (Ball) + Step);

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

100 is not a float literal
100.0 would be

```
with Display;       use Display;
with Display.Basic; use Display.Basic;

procedure Main is
   Ball : Shape_Id := New_Circle
     (X      => 0.0,
      Y      => 0.0,
      Radius => 10.0,
      Color  => Blue);
   Step : Float := 0.05;
begin
   loop
      if (Get_Y (Ball) > 100.0) then
         Step := -0.05;
      elsif Get_Y (Ball) < -100.0 then
         Step := 0.05;
      end if;

      Set_Y (Ball, Get_X (Ball) + Step);

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

These parenthesis are OK
(although useless and not Ada-stylish)

university.adacore.com