Práctica 2

Juan Llinares Mauri
74011239E
jlm109@alu.ua.es

1. Paso 1

Para el paso uno basta con dejar el código por defecto de la siguiente manera:

```
with PKG_graficos;

procedure Main is
begin
   PKG_graficos.Simular_Sistema;
end Main;
```

Lo que hará esto será generar el entorno gráfico para poder visualizar los aviones, como se indica en la figura 1.

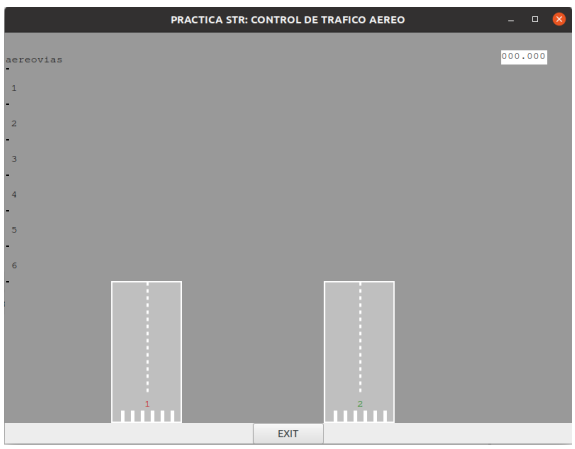

Figure 1. Entorno gráfico.

2. Paso 2

Con este paso dos empezamos a crear los aviones. En primer lugar se nos pide añadir un retardo aleatorio a la aparición de aviones. Esto lo podemos implementar de la siguiente manera en el *body* de la tarea *TareaGeneraAviones* del paquete que se nos facilitó para la práctica 1.

```
package pkg_retardosRandom is new Ada.Numerics.Discrete_Random(
   T_RetardoAparicionAviones);
generadorRetardos : pkg_retardosRandom.Generator;
```

Tendremos que resetear el generador justo después del *begin* de la tarea de generar aviones.

```
pkg_retardosRandom.Reset(generadorRetardos);
```

Luego, al final del *body* añadiremos el retardo:

```
delay(Duration(pkg_retardosRandom.Random(generadorRetardos)))
```

Como los nuevos aviones han cambiado, necesitaremos inicializarlos de distinta manera. Lo haremos de la siguiente forma:

```
ptr_avion := new T_RecordAvion;
ptr_avion.id := id;
ptr_avion.velocidad.x := VELOCIDAD_VUELO;
ptr_avion.velocidad.y := 0;
ptr_avion.aereovia := aereovia;
ptr_avion.tren_aterrizaje := False;
ptr_avion.aereovia_inicial := aereovia;
ptr_avion.pista := SIN_PISTA;
ptr_avion.color := pkg_colorRandom.Random(generadorColores);
ptr_avion.pos := Pos_Inicio(aereovia);

tarea_avion := NEW T_TareaAvion(ptr_avion);
```

Para poder darle un color aleatorio al avión que se genera hemos seguido el mismo procedimiento que hicimos con los retardos:

```
package pkg_colorRandom is new Ada.Numerics.Discrete_Random(
   T_ColorAparicionAvion);
generadorColores : pkg_colorRandom.Generator;
```

```
pkg_colorRandom.Reset(generadorColores);
```

El cuerpo de la tarea *T_TareaAvion* se deberá quedar de la siguiente manera:

```
TASK BODY T_TareaAvion IS
```

```
      avion : T_RecordAvion;
   begin
      avion := ptr_avion.all;

      Escribir(cadena => "Informacion del avion");
      Put_line("TASK Avion: " & T_IdAvion'Image(ptr_avion.id) &
         " _"
               & T_Rango_AereoVia'Image(ptr_avion.aereovia));

      Aparece(avion);

      exception
      when event : DETECTADA_COLISION =>
         Desaparece(avion);
         Escribir("ERROR en TASK ...: " & Exception_Name(
            Exception_Identity(event)));
   end T_TareaAvion;
end pkg_tareas_dinamicas
```

Podemos observar que hemos creado un puntero para vigilar los aviones que se crean y hemos añadido la excepción de cuando se detecta una colisión. Con todo esto, y el paquete *.ads* como se indica en el siguiente código, podemos observar (ver figura 2) que al ejecutar el programa aparecen aviones y sus informaciones.

```
with PKG_tipos; use PKG_tipos;
with Ada.Numerics.Discrete_Random; --paquete random generico
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
with Ada.Text_IO; use Ada.Text_IO;
with PKG_graficos; use PKG_graficos;
with PKG_debug; use PKG_debug;
with Ada.Exceptions; use Ada.Exceptions;

package GeneraAviones is
   TASK TareaGeneraAviones;
   -- Tipo tarea encargada del comportamiento de un avion
   TASK TYPE T_TareaAvion(ptr_avion : Ptr_T_RecordAvion);
   -- Puntero al avion para poder pasarle datos
   type Ptr_TareaAvion is access T_TareaAvion;
end GeneraAviones;
```

3. Paso 3

En este paso haremos que los aviones vuelen y se muevan. Para lograr esto, tendremos que añadir, después de la cadena de información para los *logs*, el siguiente código:

```
if avion.aereovia / 2 = 0 then
    avion.velocidad.X := VELOCIDAD_VUELO;
else
```

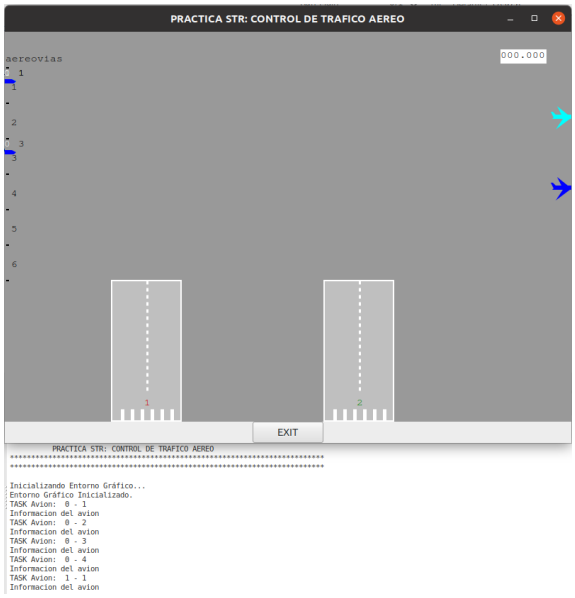

Figure 2. Ejecución del paso 2.

```
4       avion.velocidad.X := -VELOCIDAD_VUELO;
5   end if;
6
7   Aparece(avion); -- Esta sentencia ya la ten amos implementada
        en el paso 2
8
9   loop
10      Actualiza_Movimiento(avion);
11      delay(PKG_tipos.RETARDO_MOVIMIENTO);
12  end loop;
```

Lo que hacemos es revisar en qué aerovía se encuentran los aviones para darles diferente sentido de vuelo a estos[1] y que no todos vayan en la misma dirección, los hacemos aparecer y luego creamos el bucle infinito de actualizar los movimientos con su retardo.

[1] Se puede cambiar el sentido de los vuelos a como se indica en las figuras 3 y 4 cambiando el código de la condición *if* a 'avion.aereovia / 2 /= 0' en lugar de 'avion.aereovia / 2 = 0'.

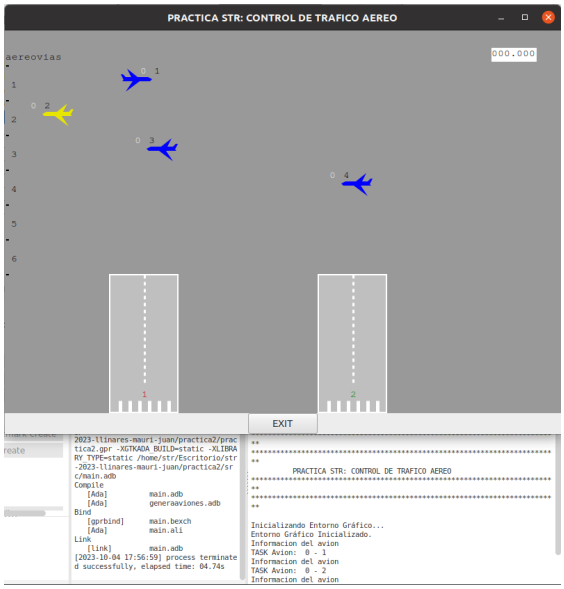

Figure 3. Dirección derecha.

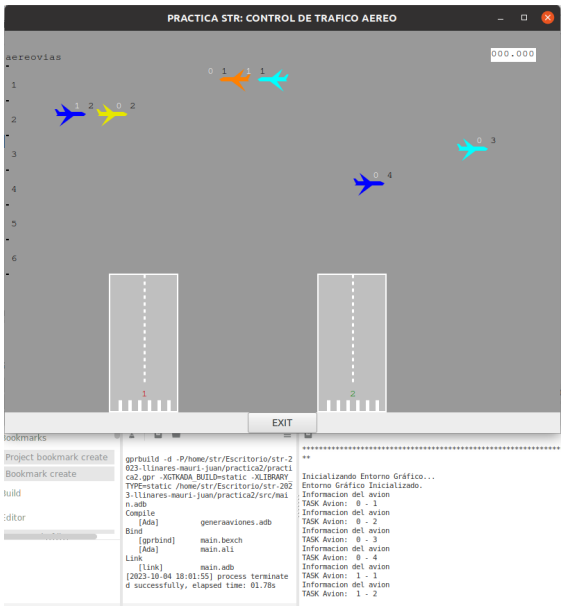

Figure 4. Dirección izquierda.