Práctica 3

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

1. Paso 1

Para esta práctica implementaremos tareas periódicas de dos maneras: usando *Ada.Calendar* y *Ada.Real_Time*.

1.1. *Ada.Calendar*

Usando este paquete de *Ada* podemos implementar la tarea de la siguiente manera. Crearemos un paquete llamado *pkg_retardos* y le añadiremos el siguiente código al *.ads*:

```
1 with Ada.Calendar; use Ada.Calendar;
2 with PKG_graficos; use PKG_graficos;
3
4 package pkg_retardos is
5    task Tarea_Retardo;
6 end pkg_retardos;
```

Aquí, estaremos importando el paquete de los gráficos para poder usar el *Actualiza_Cronometro* que se comenta en la práctica y el paquete *Calendar* de *Ada*. También, crearemos la tarea que actualizará el cronómetro cada segundo con el nombre de *Tarea_Retardo*.

El *body* del paquete es el siguiente:

```
1 package body pkg_retardos is
2    task body Tarea_Retardo is
3       Inicio, Siguiente : Time;
4       Frecuencia : constant Duration := 1.0;
5    begin
6       Inicio := Clock;
7       Siguiente := Clock + Frecuencia;
8
9       loop
```

```
10          Actualiza_Cronometro(Clock - Inicio);
11          delay until(Siguiente);
12          Siguiente := Siguiente + Frecuencia;
13       end loop;
14    end Tarea_Retardo;
15 end pkg_retardos;
```

Hemos implementado los retardos mediante *delay until*. En el siguiente paso veremos las diferencias entre este y *delay*.

Creamos variables de tipo *Time* y otra para la frecuencia de activación de la tarea. Podemos observar en la figura 1 que a los diez segundos apenas hay retardo acumulado.

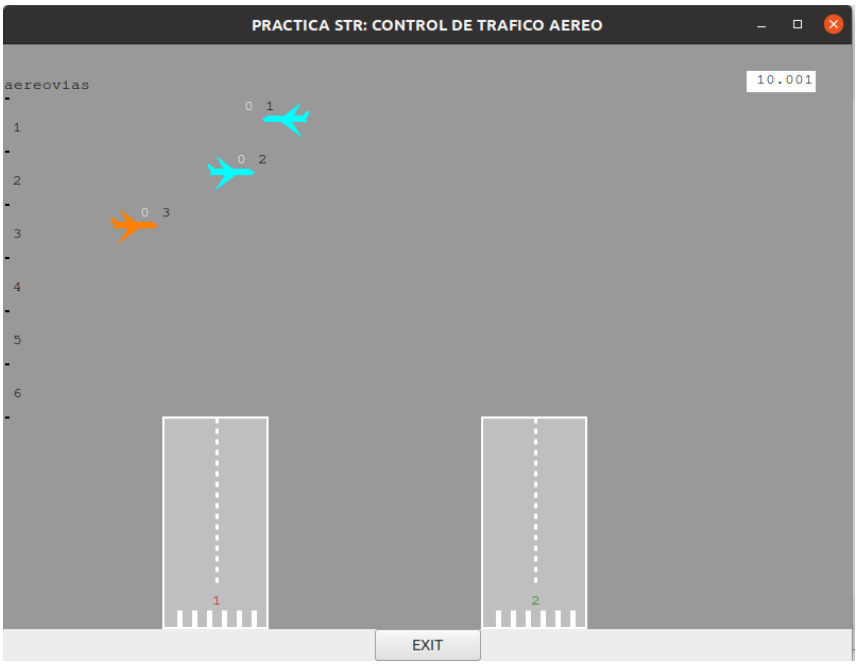

Figure 1. Retardo con *Calendar*.

Para ejecutar el programa y que funcione tendremos que añadir a nuestro *main.adb* lo siguiente:

```
1 with pkg_retardos; use pkg_retardos;
```

1.2. *Ada.Real_Time*

Para realizar este paquete seguiremos prácticamente el mismo procedimiento que con el *Calendar* pero cambiando unas cosas. Creamos el paquete *pkg_retardos2* y añadimos el siguiente código al *.ads*:

```
1 with Ada.Real_Time; use Ada.Real_Time;
2 with PKG_graficos; use PKG_graficos;
3
4 package pkg_retardos is
```

```
5     task Tarea_Retardo;
6  end pkg_retardos;
```

Aquí simplemente cambiamos el paquete importado.

El *body* del nuevo paquete deberá quedar de la siguiente manera:

```
1  package body pkg_retardos is
2     task body Tarea_Retardo is
3        Inicio, Siguiente : Time;
4        Frecuencia : Time_Span := Milliseconds(1000);
5     begin
6        Inicio := Clock;
7        Siguiente := Clock + Frecuencia;
8
9        loop
10          Actualiza_Cronometro(To_Duration(Clock - Inicio));
11          delay until(Siguiente);
12          Siguiente := Siguiente + Frecuencia;
13       end loop;
14    end Tarea_Retardo;
15 end pkg_retardos;
```

Observamos que la diferencia se encuentra en la manera de instanciar la frecuencia.

En la figura 2 se observa que no hay tampoco retardo acumulado gracias a la implementación.

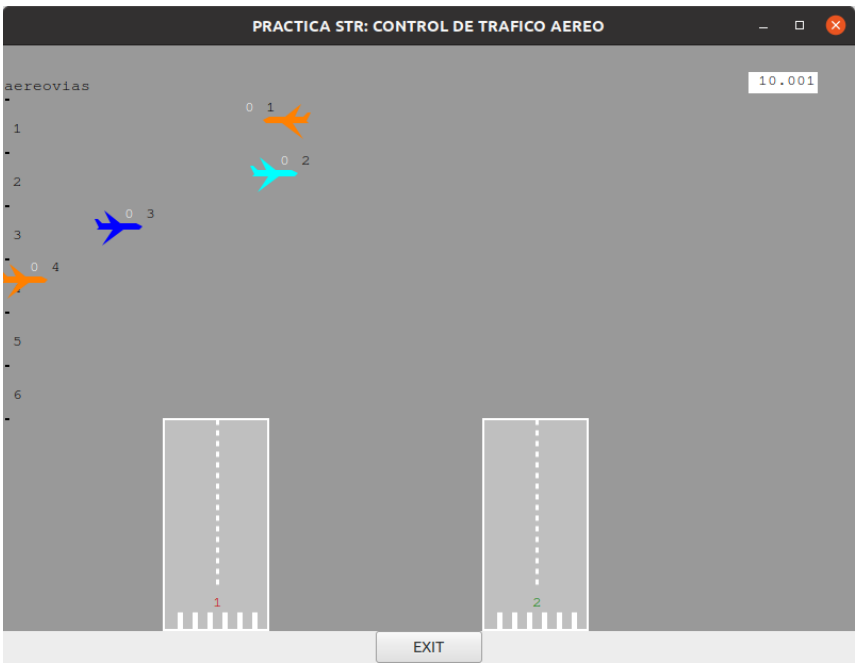

Figure 2. Retardo con *Real_Time*.

2. Paso 2

En este paso veremos la diferencia entre usar *delay until* y *delay*. Simplemente cambiaremos las líneas donde aparezca *delay until* por *delay*:

```
1 delay 1.0;
```

La principal diferencia que observamos es que prácticamente nunca se inicia la tarea en el segundo exacto .000 y tiene milésimas más altas que con *delay until*. Esto se debe a que la tarea espera un segundo y luego se ejecuta, causando que el tiempo de ejecución sea superior al del *delay until* porque:

$$1 + tiempo\_ejecucion > tiempo\_hasta\_siguiente + tiempo\_ejecucion$$

Siendo, en la fórmula anterior:

- *tiempo_ejecucion*: Tiempo que tarda la tarea en sí en ejecutarse.
- *tiempo_hasta_siguiente*: Tiempo que tarda *delay until* hasta activar la tarea.