Guangen Wu

Memoria Puzzle 2

1. Resumen

En este puzzle 2 nos interesa crear una interfaz gráfica para mostrar el uid de de la tarjeta nfc. Utilizando la biblioteca Ruby-GNOME2/gtk3 e importando el puzzle 1 también como biblioteca.
La ventana principal está formada por un Label que pide login con la tarjeta RF (carnet UPC o Mifare) y un Button de Clear que resetea y vuelve a la ventana principal.

2. Bibliotecas

Básicamente vamos a utilizar dos bibliotecas: la gtk3 para desarrollar el entorno gráfico y el puzzle 1 para no tener que reescribir el código de lectura de uid.

Primero instalamos la gtk3, para eso hemos usado los siguientes comandos. Luego procedemos a instalar la gema. No hemos tenido problemas en este paso.

```
pi@guang:~ $ sudo apt update
sudo apt install -y libgtk-3-dev gir1.2-gtk-3.0
```

```
pi@guang:~ $ cd PBE
pi@guang:~/PBE $ gem install gtk3
```

El siguiente paso es utilizar el código del puzzle 1 como librería para el puzzle 2. No hemos modificado nada el código que teníamos del puzzle 1. Sólo tuvimos que ponerlos en la misma carpeta ambos puzzles y escribir la siguiente línea en el puzzle 2.

```
require_relative "puzzle1" # Usar como libreria el puzzle1 para manejar la lectura NFC
```

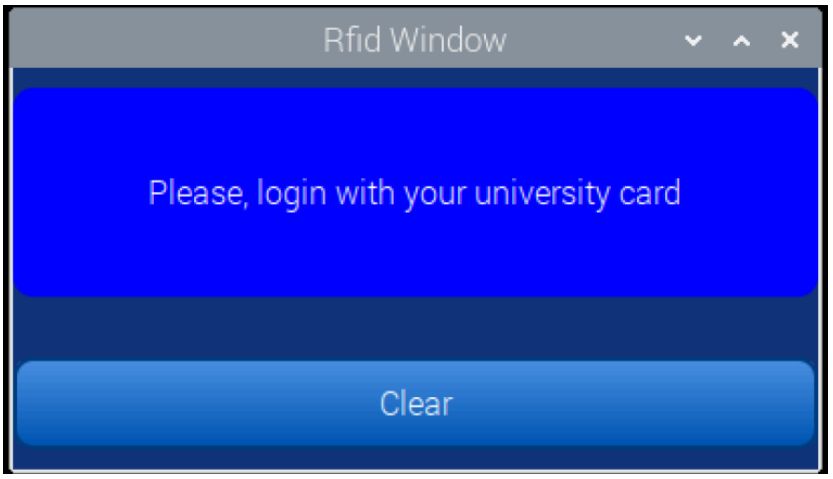
3. Problemas encontrados

Directamente hemos probado una versión simple pero nos dio errores. Nos dimos cuenta de que el problema era que read_uid es bloqueante y el tratamiento de eventos en entornos gráficos NO debe bloquearse.

Para que la interfaz gráfica (GTK) y la lectura NFC trabajen al mismo tiempo sin bloquearse, empleamos un par de técnicas en Ruby:
- Uso de hilos auxiliares (Thread.new) para la lectura NFC.
- Uso de GLib::Idle.add para actualizar la interfaz en el hilo principal de GTK.

De esta manera:
- El hilo principal queda libre para procesar eventos de la interfaz (pulsar botones, refrescar la ventana, etc.).
- El hilo NFC hace su bucle de lectura y, cuando detecta algo, agenda la actualización de la UI con GLib::Idle.add.

Así hemos logrado que funcione sin problemas la interfaz.
Primero aparece esta ventana pidiendo una tarjeta NFC.



Luego al acercar la tarjeta nos muestra su uid en hexadecimal.

Si le damos al botón “Clear” vuelve a la primera pantalla. Y si le damos al “X” de la esquina superior derecha se cierra la ventana.

4. Código

Para personalizar la interfaz gráfica hemos utilizado css escribiendo el código de abajo. El funcionamiento es bien sencillo, he creado un fichero llamado “disseny.css” y lo metí en la misma carpeta que puzzle 2, que luego añadiré un par de líneas de código para importarlo. En esta imagen de abajo muestro el código completo de css.

```css
/* Estilos para los botones */
button {
    background: linear-gradient(to bottom, #4a90e2, #0056b3); /* Degradado azul */
    border-radius: 10px; /* Bordes redondeados */
    border: 2px solid #003d80; /* Borde azul oscuro */
    color: white; /* Texto en color blanco */
    font-weight: bold; /* Texto en negrita */
    padding: 10px; /* Espaciado interno */
    transition: background 0.3s ease-in-out; /* Transicion suave al cambiar el color de fondo */
}

/* Cambio de color cuando el cursor pasa sobre el botón */
button:hover {
    background: linear-gradient(to bottom, #67a8ff, #004080); /* Azul más claro en hover */
}

/* Cambio de color cuando el boton esta presionado */
button:active {
    background: linear-gradient(to bottom, #ff6b6b, #b30000); /* Rojo degradado cuando se presiona *
}

/* Color de fondo de la ventana principal */
window {
    background-color: #133578; /* Azul oscuro de fondo */
}
```


```css
/* Color de fondo de la ventana principal */
window {
    background-color: #133578; /* Azul oscuro de fondo */
}

/* Estilos para las etiquetas de texto */
label {
    border-radius: 10px; /* Esquinas redondeadas en las etiquetas de texto */
}
```


En las tres imágenes siguientes están el código completo del puzzle 2.

```ruby
require "gtk3"
require "thread"
require_relative "puzzle1" # Usar como libreria el puzzle1 para manejar la lectura NFC

# Metodo para aplicar estilos CSS a la interfaz
def apply_css
  css_provider = Gtk::CssProvider.new
  css_provider.load(path: "disseny.css") # Carga el archivo CSS con los estilos personalizados

  style_context = Gtk::StyleContext
  screen = Gdk::Screen.default

  # Aplica el CSS a toda la pantalla
  style_context.add_provider_for_screen(screen, css_provider, Gtk::StyleProvider::PRIORITY_USER)
end

# Clase principal que gestiona la interfaz y la interaccion con el lector NFC
class NFCApp
  def initialize
    apply_css # Aplica los estilos CSS al iniciar la interfaz

    # Crear la ventana principal
    @window = Gtk::Window.new("Rfid Window")
    @window.set_size_request(400, 200) # Establecer el tamano de la ventana
    @window.signal_connect("destroy") { Gtk.main_quit } # Cerrar la aplicacion al cerrar la ventana

    # Contenedor vertical para organizar los elementos
    @vbox = Gtk::Box.new(:vertical, 10)
    @window.add(@vbox)

    # Etiqueta para mostrar mensajes
    @label = Gtk::Label.new("Please, login with your university card")
    @label.override_background_color(:normal, Gdk::RGBA.new(0, 0, 1, 1)) # Fondo azul
    @label.override_color(:normal, Gdk::RGBA.new(1, 1, 1, 1)) # Texto en blanco
    @vbox.pack_start(@label, expand: true, fill: true, padding: 10)
```

En esta primera imagen cargamos primero las librerías para usar gtk3, habilitar el uso de hilos e importar la clase Rfid del puzzle 1, que maneja la lectura de tarjetas NFC. Luego aplicamos los estilos css del archivo “disseny.css”. Finalmente creamos la clase principal que controla la interfaz gráfica y la interacción con el lector NFC.

```ruby
    # Boton para limpiar la pantalla y reiniciar la lectura
    @clear_button = Gtk::Button.new(label: "Clear")
    @clear_button.signal_connect("clicked") { clear_label } # Conectar el boton a la accion de limpiar
    @vbox.pack_start(@clear_button, expand: false, fill: true, padding: 10)

    @window.show_all # Mostrar todos los elementos de la ventana

    # Inicializar el lector NFC
    begin
      @rfid = Rfid.new
    rescue StandardError => e
      update_label("No NFC reader found: #{e.message}", 1, 0, 0) # Mostrar error si no se detecta lector
      return
    end

    # Iniciar la primera lectura NFC
    start_reading_thread
  end

  # Metodo para iniciar un hilo que realice la lectura NFC
  def start_reading_thread
    Thread.new do
      lectura
      Thread.exit # Terminar el hilo despues de la lectura
    end
  end

  # Metodo que realiza la lectura NFC y actualiza la UI
  def lectura
    @uid = @rfid.read_uid # Leer el UID de la tarjeta NFC
    GLib::Idle.add { gestion_UI } # Llamar a gestion_UI en el hilo principal de GTK
  end

```

En esta segunda imagen acabamos de definir el diseño de la ventana y luego creamos una instancia de “Rfid” para inicializar el lector nfc. Luego definimos un método para iniciar un hilo auxiliar que realice la lectura. Finalmente está el método “lectura” que lee la tarjeta y guarda el uid en “@uid” y luego llama al hilo principal.

```ruby
  # Metodo que actualiza la interfaz grafica cuando se detecta un UID
  def gestion_UI
    if @uid && !@uid.empty?
      update_label("UID: #{@uid}", 1, 0, 0) # Muestra el UID en pantalla con fondo rojo
    end
  end

  # Metodo para actualizar el texto y color de la etiqueta
  def update_label(text, r, g, b)
    @label.set_text(text) # Cambia el texto del label
    @label.override_background_color(:normal, Gdk::RGBA.new(r, g, b, 1)) # Cambia el fondo del label
  end

  # Metodo que se activa al pulsar "Clear": limpia la pantalla y reinicia la lectura
  def clear_label
    @uid = "" # Borra el UID almacenado
    update_label("Por favor, acerca tu tarjeta de identidad de la uni", 0, 0, 1) # Vuelve al estado inicial con fondo azul
    start_reading_thread # Relanza la lectura de NFC
  end
end

Gtk.init # Inicializa GTK
NFCApp.new # Crea una nueva instancia de la aplicacion y lanza la interfaz
Gtk.main # Mantiene la aplicacion en ejecucion

```

En esta tercera y última imagen definimos primero “gestion UI” que básicamente comprueba si “@udi” no está vacío, entonces cambia la etiqueta y muestra el uid en pantalla con fondo rojo. Luego hemos definido el método “update_label” que se encarga de cambiar el texto y color de la etiqueta. Después también hemos escrito el método “clear_label” que al darle click limpia el uid almacenado, vuelve al estado inicial con fondo azul y relanza la lectura de nfc.