ARSAT
Laboratorio de sistemas embebidos
Gerencia de Desarrollo
Tecnológico e Innovación

Librería de workarounds

Fecha: 19-08-2015
Revisión: 1

Contenido

1 Objetivo 2
2 Librería de workarounds Ginga 2
2.1 **Canvas** 2
2.1.1 **Canvas**:attrFont(face, size, style) 2
2.1.2 **Canvas**:drawText(x, y, text) 2
2.1.3 **Canvas**:measureText(text) 2
2.1.4 **Canvas**:clear(x, y, width, height) 2
2.1.5 **Canvas**:attrClip(x, y, width, height) 2
2.1.6 **Canvas**:compose(x, y, srcCanvas, srcX, srcY, srcW, srcH) 3
2.1.7 **Canvas**:flush() 3
2.1.8 **Canvas**:attrScale(w, h) 3
2.2 **Event** 3
2.2.1 **Event**.register(nPos, fHandler) 3
2.2.2 **Event**.unregister(fHandler) 3
2.3 **TCPConnection** 3
2.3.1 **TCPConnection**:connect(sHostIP, sHostPort, fCallback) 3
2.3.2 **TCPConnection**:disconnect() 4
2.3.3 **TCPConnection**:send(sMessage) 4
2.3.4 **TCPConnection**:setCallbackReceive(fCallback) 4
2.4 Encoding 4
3 Hello World 5
3.1 Archivo main.lua 6

1 Objetivo

En este documento se expone una librería de workarounds, que conforma un conjunto de soluciones a los problemas que surgen a partir de las diferencias encontradas en las distintas implementaciones del middleware del parque de receptores.

2 Librería de workarounds Ginga

La librería presentada cumple con la especificación de los objetos procedurales LUA presentados en la norma ABNT NBR 15606-2:2011. Se propone utilizarla en reemplazo de los objetos nativos provistos por la implementación de middleware particular.

Para cada función de los módulos especificados en la norma, se enumeran las diferencias encontradas en los receptores que fueron tratadas por la librería.

2.1 **Canvas**

Reemplaza a **canvas** nativo.

2.1.1 **Canvas**:attrFont(face, size, style)

- Problemas con **canvas**:measureText y **canvas**:attrFont. Si se utilizan ambos múltiples veces en forma alternada, se cierra la aplicación. En este caso, en los dispositivos que presentan este problema, **Canvas**:attrFont solamente configura la fuente la primera vez que se llama a la función para setearla.
- **canvas**:attrFont devuelve el estilo de la fuente con la primera letra en mayúscula en lugar de todo minúscula. **Canvas**:attrFont devuelve todo en minúscula (el set de **canvas**:attrFont recibe todo en minúsculas).

2.1.2 **Canvas**:drawText(x, y, text)

- En algunos receptores, **canvas**:drawText no soporta graficar textos con múltiples líneas. **Canvas**:drawText grafica en una nueva línea el texto encontrado luego del caracter “\n”.

2.1.3 **Canvas**:measureText(text)

- **canvas**:measureText en algunos receptores no soporta medir textos con múltiples líneas. Análogo al problema anterior.

2.1.4 **Canvas**:clear(x, y, width, height)

- En algunos receptores, **canvas**:clear ignora los parámetros.

2.1.5 **Canvas**:attrClip(x, y, width, height)

- En algunos receptores los parámetros se interpretan como dos puntos (esquina

superior izquierda y esquina inferior derecha del rectángulo) en el plano gráfico de Ginga, en lugar de ser un punto y un ancho y alto.

2.1.6 **Canvas**:compose(x, y, srcCanvas, srcX, srcY, srcW, srcH)

- Algunos receptores no soportan los parámetros opcionales srcX, srcY, srcW, srcH.
- Algunos dispositivos afectan el clipping (attrClip) al hacer compose.

2.1.7 **Canvas**:flush()

- Necesidad de doble flush en un receptor.

2.1.8 **Canvas**:attrScale(w, h)

- No se encuentra implementado en algunos equipos. El workaround no lo implementa, pero evita que se cierre la aplicación por la falta de la función.
- En algunos equipos no se puede obtener los parámetros de escala del **canvas**, pero sí se puede modificarlos. Para estos equipos, se implementa la función **Canvas**:attrScale() para obtenerlos.

2.2 Event

Reemplaza a **event** nativo.
Esta clase toma control sobre el registro a eventos NCL. En conjunto con el esquema utilizado para la atención de eventos de teclas en NCL, resuelve diferencias encontradas entre receptores en el comportamiento respecto del foco de medias NCL.

2.2.1 **Event**.register(nPos, fHandler)

- nPos y fHandler corresponden a los parámetros de posición de registro y función de callback descriptos en la norma (sección 10.3.3.2).
- Los parámetros opcionales de clase y dependientes de clase de evento quedan por fuera de la implementación de la librería (sección 10.3.3.2 de la norma).
- Forma parte del workaround para el problema con la función event.unregister.

2.2.2 **Event**.unregister(fHandler)

- Problemas con la función event.unregister que no se encuentra implementada en algunos dispositivos.

2.3 TCPConnection

Reemplaza a los eventos TCP nativos.
Resuelve diferencias encontradas en receptores en cuanto al tipo de dato recibido como identificador TCP (string, number, userdata) en la conexión.

2.3.1 **TCPConnection**:connect(sHostIP, sHostPort, fCallback)

- Realiza un intento de conexión TCP a un **IP** y **puerto** dados y luego llama a fCallback de

modo asincrónico. Devuelve **true** si pudo iniciar el intento de conexión. Si no, devuelve además un texto con una descripción del error sucedido.
- Parámetros de fCallback: tTCPConnection, bSuccess, sError, donde tTCPConnection es el objeto que llamó al callback.

2.3.2 **TCPConnection**:disconnect()

- Realiza la desconexión TCP del **IP** y **puerto** asignados en la conexión. Si hubo algún error (no había conexión hecha, etc.) devuelve **false** y un texto de con una descripción. Sino devuelve **true**.

2.3.3 **TCPConnection**:send(sMessage)

- Envía datos por la conexión previamente establecida. Si hubo algún error (no había conexión establecida, etc.) devuelve **false** y un texto de con una descripción. Sino devuelve **true**.

2.3.4 **TCPConnection**:setCallbackReceive(fCallback)

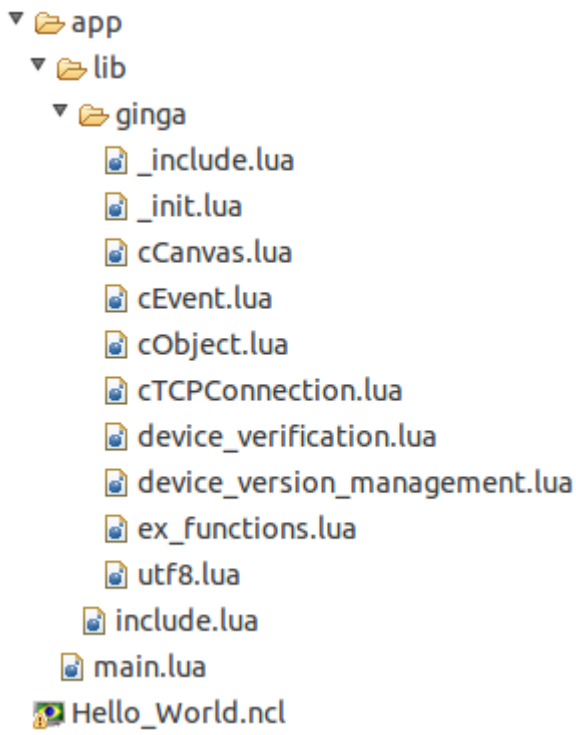
- Setea la función donde se desean recibir los datos que vienen de la conexión TCP realizada.
- La aridad de la función de callback es fCallback(tTCPConnection, sMessage).

2.4 Encoding

Algunos dispositivos trabajan con encoding UTF-8, en lugar de Latin-1 como especifica la norma (ver ABNT NBR 15606-1:2013, sección 11.4).

- **utf8ToLatin1** y **utf8FromLatin1**
- **_strLen**(s) y **_strSub**(s, startIndex, length)
- Problemas con el operador # por el encoding UTF-8 que utiliza caracteres de longitud variable en bytes, a diferencia de Latin-1.

3 Hello World



La librería comprende un archivo NCL y la carpeta “app”, que contiene la carpeta “lib” y el archivo “main.lua”.
La carpeta “lib” contiene los workarounds anteriormente nombrados.
El archivo “main.lua” es el punto de entrada a la aplicación a desarrollar.

3.1 Archivo main.lua

```
require 'lib/include'
lib_ginga_init(function()

------------------------------------------------------
  Canvas:attrFont('Tiresias', 16, 'normal')
  Canvas:attrColor(1, 1, 1, 255)
  Canvas:clear()

  Canvas:attrColor(254, 254, 254, 255)
  Canvas:drawText(50, 50, 'Hello World!')
  Canvas:flush()
------------------------------------------------------

end)
```

Para poder comenzar a utilizar la librería es necesario inicializarla. Se debe realizar un llamado a la función lib_ginga_init(fCallback). Al finalizar la inicialización de la librería, se llamará a la función fCallback.
Luego las clases de la librería se encontrarán listas para su uso.