Algoritmos y Programación I (75.40 - 95.14)
Trabajo práctico n.º 3

Primer cuatrimeste 2016

Consigna

A todo informático le gusta alardear de sus conocimientos. Qué mejor forma de hacerlo que con un programa que pueda disfrutar toda la familia. Imagina la sorpresa de tus amigos cuando vean lo que programaste. Con este Kit puedes programar un reproductor de música y ser la envidia de tus amigos.

Eso decía el kit que compró Jimmy, pero aun no lo logró programar. Es entonces que nos encargó la tarea de completar la implementación de su reproductor de música, que quedó a medio terminar.

Funcionamiento del reproductor

La ventana del reproductor se encuentra dividida en tres secciones: + reproducción: muestra el estado del reproductor (*Reproduciendo* o *Detenido*), y el título y artista de la canción actual + modificación de la cola de reproducción: permite agregar y remover canciones. Contiene un cuadro de texto donde se puede introducir la ruta del archivo de la canción a agregar/remover, y al realizar una acción muestra un mensaje que indica si se pudo realizar la modificación con éxito o no. + lista de canciones de la cola: muestra el título y artista de las próximas canciones de la cola de reproducción.

Como en todo programa moderno, uno debe contemplar que el usuario puede equivocarse. *Agregar canción* y *remover canción* se deben poder deshacer,

o rehacer en caso que alguien haya deshecho algo por error. Por ejemplo, agregamos una canción que no nos gusta; en ese caso podemos deshacer la operación y que se elimine de la lista. Luego podemos cambiar de idea, y en vez de volver a buscarla para agregar, simplemente podemos rehacer la última operación que deshicimos.

Al ejecutar el reproductor, pasando como argumento un directorio con unas pocas canciones, se abrirá una ventana como la siguiente:

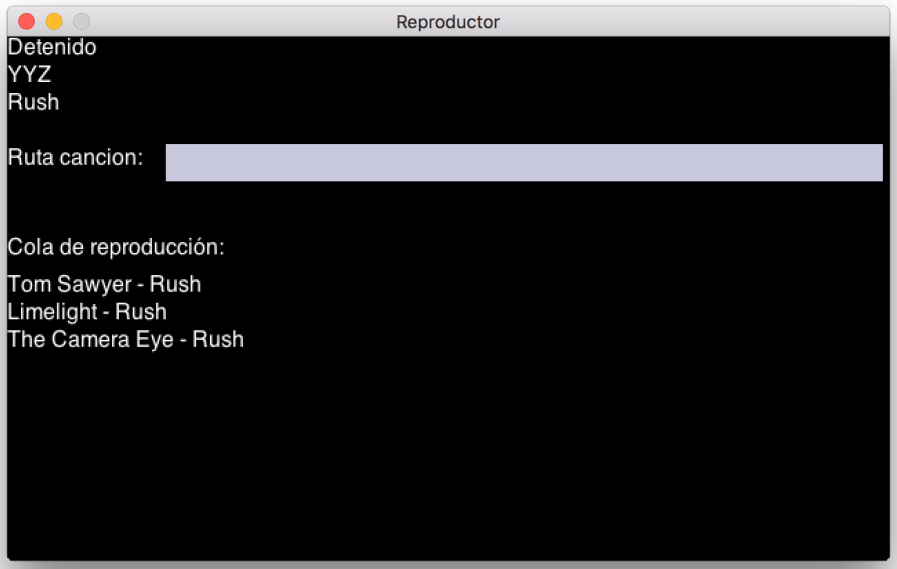
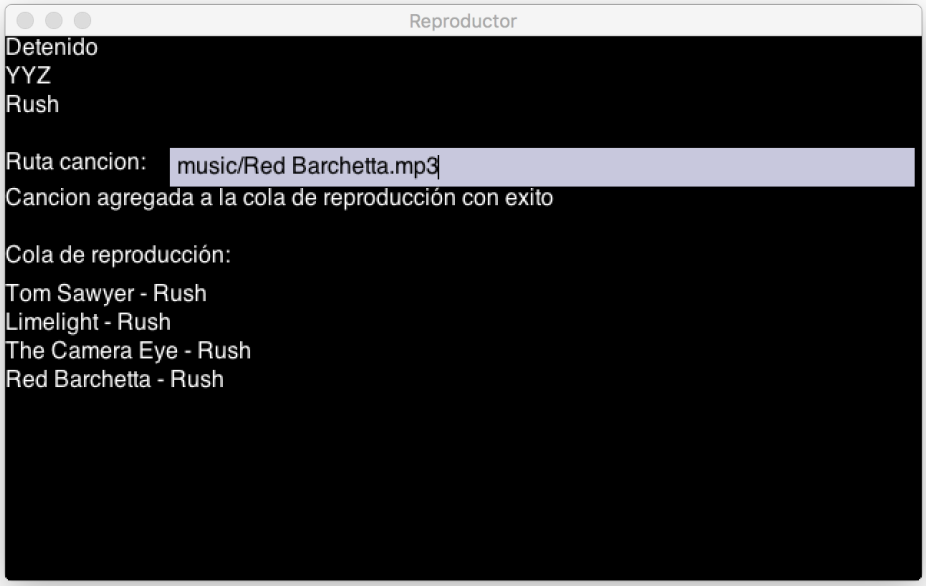
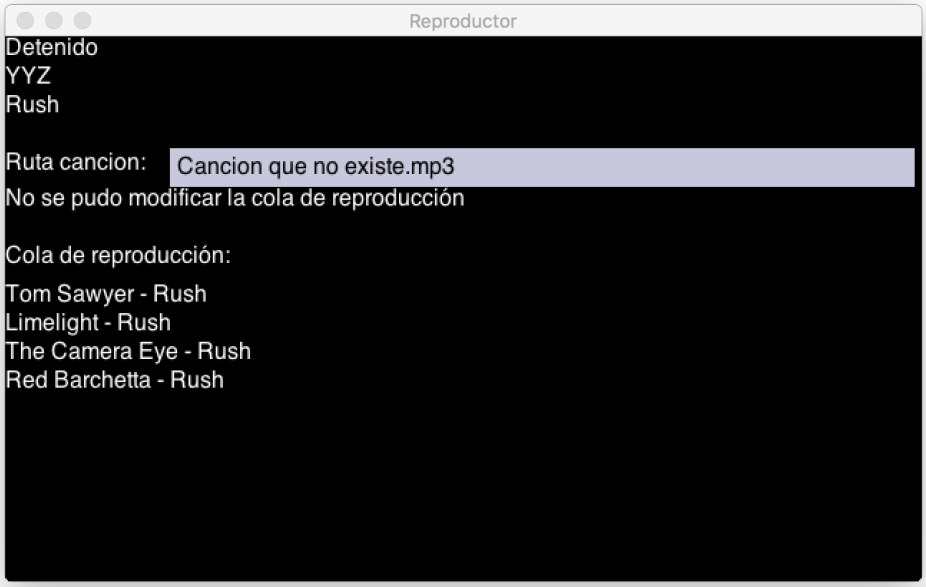


Figure 1: Vista inicial del reproductor

Después queremos agregar una canción, para lo que introduciremos la ruta al archivo en el cuadro de texto y presionamos el atajo para *Agregar canción*. Se verá el mensaje de exito y se agregará la canción si existe, o se verá un mensaje que indica que hubo un error y no se agregó. Obtendremos algo como lo que se ve en las siguientes imagenes:


Reproductor
Detenido
YYZ
Rush
Ruta cancion:
music/Red Barchetta.mp3
Cancion agregada a la cola de reproducción con exito
Cola de reproducción:
Tom Sawyer - Rush
Limelight - Rush
The Camera Eye - Rush
Red Barchetta - Rush


Reproductor
Detenido
YYZ
Rush
Ruta cancion:
Cancion que no existe.mp3
No se pudo modificar la cola de reproducción
Cola de reproducción:
Tom Sawyer - Rush
Limelight - Rush
The Camera Eye - Rush
Red Barchetta - Rush

Por último, queremos agregar una canción más, que es una archivo *wav*. Al añadirlo podemos ver que se agregó a la lista pero el archivo no tenía los tags de *título* ni *artista*, por lo que el reproductor no pudo obtener esa información y lo muestra como desconocida:

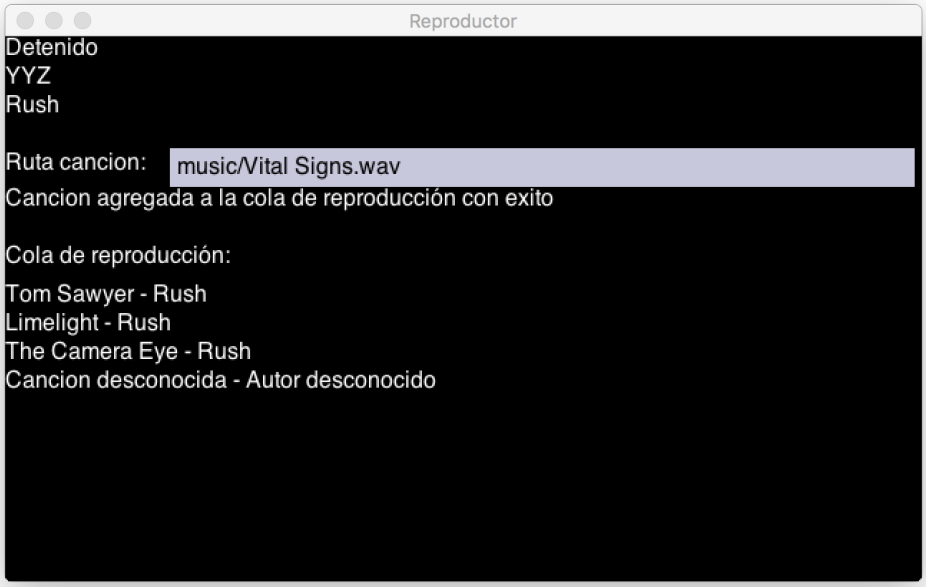


Figure 2: Agregada una canción sin tags

Decidimos no dejar esa canción, por lo que deshacemos la acción de agregar, y vemos lo siguiente:

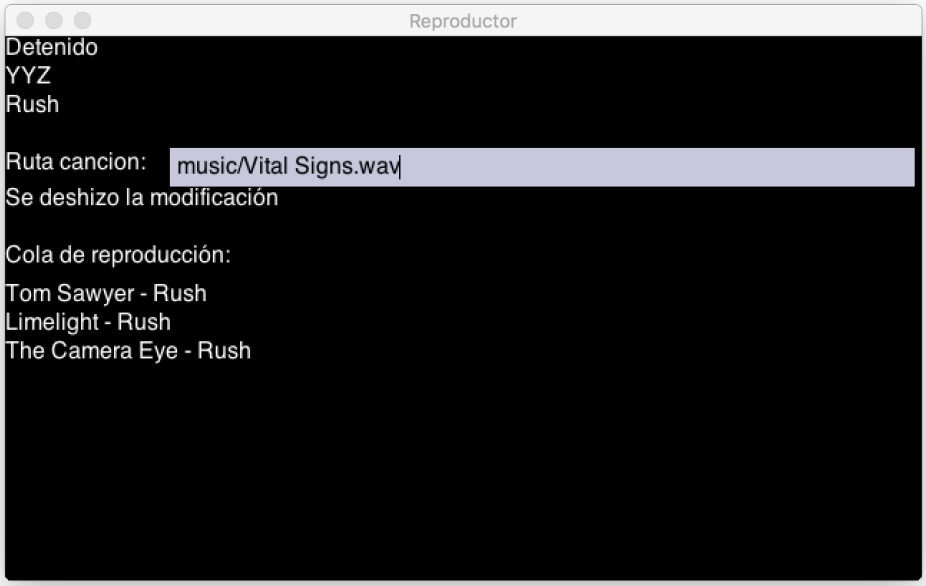


Figure 3: Deshacer una acción de agregar

Uso

Para reproducir archivos *wav*, solo se necesita Python 3.5.1+. Para reproducir archivos en otros formatos, se necesita además AVbin[1]. Ya se encuentra entre los archivos entregados *avbin.dll*, por lo que *los usuarios de Windows no deberán instalar nada extra.* Para *usuarios de Linux y Mac, deberán descargar e instalar la versión de AVbin correspondiente a su sistema operativo.*

El programa recibe un solo argumento, la ruta del directorio donde se encuentran las canciones que se quieren cargar. La carga de archivos se realizará de manera recursiva sobre los subdirectorios que se encuentren en la ruta dada, por lo que también se agregaran las canciones que se encuentren en dichos subdirectorios, y sus subdirectorios, y así sucesivamente.

Suponiendo que queremos reproducir todos los archivos que se encuentran en la carpeta `C:/Users/user/Musica`, debemos ejecutar:

```
$ python algoReproductor.py 'C:/Users/user/Musica'
```

Atajos

Reproducción

- Espacio: Reproducir/Pausa
- Flecha derecha: Avanzar a la próxima canción
- Flecha izquierda: Retroceder a la canción anterior

Modificación de cola de reproducción

- Ctrl+A: Agregar a la cola la canción que se encuentra en la ruta especificada en el cuadro de texto.
- Ctrl+R: Remover de la cola la canción cuya ruta se encuentra especificada en el cuadro de texto
- Ctrl+Z: Deshacer
- Ctrl+Y: Rehacer

[1] https://avbin.github.io/AVbin/Download.html

Agregar y remover canciones durante la ejecución

Para agregar y remover canciones debe especificarse la ruta de la misma en el cuadro de texto del reproductor. Para escribir, si no se encuentra activo, hacer click sobre el cuadro de texto.

Cuando está activo el cuadro de texto no se toman los atajos de reproducción. Para que vuelvan a funcionar debe hacerse click en cualquier parte del reproductor fuera del cuadro de texto.

Código recibido

Recibirán un código parcialmente terminado, en el que deberán completar las funciones y clases no implementadas. Las mismas son: + En el archivo *algoReproductor.py*, deberán completar la función `agregar_canciones`. + En el archivo *ventana*reproductor.py_, falta implementar la clase `WidgetColaReproduccion`. + En el archivo *cola*reproduccion.py_, hay que implementar la clase `ColaDeReproduccion` en su totalidad. + En *cancion.py* se encuentra sin implementar la clase `Cancion`. + Por último, si necesitaran una pila o una cola, deberán implementar el TDA como una clase con sus métodos correspondientes.

Todas las clases y funciones ya se encuentran definidas y documentadas. Además de lo pedido, *pueden agregar cualquier función, atributo y/o método que consideren necesario, pero no pueden modificar la interfaz definida* (o sea, no pueden cambiar la firma de los métodos de las clases o hacer que se comporten diferente a lo que dice la documentación).

Criterios de aprobación

A continuación se describen los criterios y lineamientos que deben respetarse en el desarrollo del trabajo.

Informe

El informe debe consistir en una descripción del **diseño** del programa.

Debe recordarse que la etapa de diseño es *anterior a la implementación*, por lo tanto debe describirse, utilizando texto y/o diagramas, cómo se va a estructurar el código para cumplir con las especificaciones de la consigna.

Algunas preguntas que deberían responderse:

- A grandes rasgos, ¿cómo será el flujo del programa?
- ¿Qué estructura(s) de datos se usan para almacenar y representar los datos en memoria?
- ¿Qué clases se usaron para modelar la solución?
- ¿Encontraron alguna dificultad al trabajar con una parte del código ya implementado?

Código

Además de satisfacer las especificaciones de la consigna, el código entregado debe cumplir los siguientes requerimientos:

- El código debe ser claro y legible.
- El código debe estructurarse en funciones y, cuando corresponda, módulos. Las funciones deben definirse de la manera más genérica posible.
- Todas las funciones deben estar adecuadamente documentadas, y donde sea necesario el código debe estar acompañado de comentarios.
- De la misma forma, las clases deben tener sus atributos y métodos definidos correctamente, y documentados.

Entrega

La entrega del trabajo consiste en:

- El informe y código fuente impresos. Para el código fuente utilizar una tipografía `monoespaciada`.
- El informe en formato *PDF*.
- Una versión digital de todos archivos `.py` de código, separados del informe. En el caso de ser más de un archivo, comprimidos en un `.zip`.

El informe impreso debe entregarse en clase. Los dos últimos (PDF y código fuente) deben enviarse a la dirección electrónica `tps.7540rw@gmail.com` con el asunto *TP3 - PADRON1 - PADRON2* .

Este trabajo práctico se desarrolla en forma **grupal**. El plazo de entrega vence el **viernes 3 de Junio de 2016**.