Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2233 Programación Avanzada (2020-2)

Tarea 02

Entrega

- **Avance de tarea**
  - **Fecha y hora:** miércoles 21 de octubre de 2020, 20:00
  - **Lugar:** Repositorio personal de GitHub — Carpeta: `Tareas/T02/`
- **Tarea**
  - **Fecha y hora:** sábado 31 de octubre de 2020, 20:00
  - **Lugar:** Repositorio personal de GitHub — Carpeta: `Tareas/T02/`
- `README.md`
  - **Fecha y hora:** lunes 2 de noviembre de 2020, 20:00
  - **Lugar:** Repositorio personal de GitHub — Carpeta: `Tareas/T02/`

Objetivos

- Utilizar conceptos de interfaces y PyQt5 para implementar una aplicación gráfica e interactiva.
- Entender y aplicar los conceptos de *back-end* y *front-end*.
- Implementar una separación entre *back-end* y *front-end*.
- Aplicar conocimientos de *threading* en interfaces.
- Aplicar conocimientos de señales.

Índice

1. *DCCumbia*

Finalmente, se ha encontrado una cura para el Coronavirus y se volvieron a abrir las discotecas. Junto a tus amigos, muy emocionados porque por fin podrían salir a desempolvar sus mejores pasos de baile, se dan cuenta de que han pasado tantos meses sin poder ir a un club que ya no recuerdan cómo bailar.

Ante esto, tus amigos recuerdan que les contaste que habías aprendido sobre **Interfaces gráficas** y `PyQt5`. Te piden que, como ya habías desarrollado con éxito una simulación de la *DCCumbre olímpica*, los ayudes ahora desarrollando ***DCCumbia***, un programa que les permita recuperar todos los pasos de baile que tenían y así lucirse nuevamente en la pista.

Figura 1: Logo de *DCCumbia*

Para esto deberás utilizar tus conocimientos de interfaces gráficas y *threading* para modelar un juego de baile, el cual simula una discoteca donde podrás bailar al ritmo de la música.

2. Flujo del programa

DCCumbia es un juego de ritmo cuyo objetivo es impresionar al público con tus increíbles pasos de baile y durar la mayor cantidad de niveles en la pista para llegar a ser un gran *Cumbia Master*.

Antes de empezar a jugar cada nivel, el jugador deberá prepararse para la partida, seleccionando las configuraciones correctas para ella. Una vez iniciado el nivel, el jugador deberá realizar una serie de pasos de baile al ritmo de la música según una combinación de teclas, las cuales aparecerán en pantalla. Finalmente, una vez finalizada la ronda la música se detendrá y el programa entregará un grado de aprobación, que determina el puntaje total obtenido por el jugador, y si el jugador puede mantenerse en el juego o será expulsado del escenario, finalizando la partida. Mientras no sea expulsado del programa, el jugador podrá participar en tantas rondas como lo desee, acumulando puntaje que le permitirá comprar más bailarines, quienes seguirán los mismos pasos de baile.

Al iniciar el programa se mostrará la **ventana de inicio**, en donde el jugador podrá elegir si desea ver el ***ranking* de puntajes** o **iniciar una nueva partida**, eligiendo un nombre de usuario que cumpla las restricciones pedidas. En caso de iniciar una nueva partida, se procederá a abrir la **ventana de juego** en fase de preparación.

En la **fase de preparación** el jugador tendrá la opción de **seleccionar una canción** y una **dificultad**. Además podrá **comprar pingüinos bailarines (pingüirines)** y **situarlos en la pista de baile** para

que muestren la coreografía. Una vez terminadas las configuraciones, el jugador podrá **pasar a la fase de juego** y comenzar un nivel.

Una vez iniciado el nivel se reproducirá la canción seleccionada y, en un sector de la ventana denominado **zona de ritmo**, aparecerán secuencias de flechas en la parte superior que irán descendiendo hasta desaparecer o ser capturadas. El jugador deberá **atrapar estas flechas** presionando la tecla correspondiente cuando una flecha esté en la **zona de captura**, lo que se verá reflejado en distintos **pasos de baile** para los bailarines en la pista.

Al terminar un nivel, la musica se detendrá y aparecerá la **ventana de resumen**, que mostrará los resultados de la ronda recién finalizada. En caso de que la aprobación no supere el mínimo, se avisará al jugador que fue expulsado de la pista de baile y el juego terminará tras registrar su puntaje acumulado obtenido en el *ranking* de puntajes. En cambio, si la aprobación es positiva, el jugador recibirá dinero equivalente al puntaje obtenido en el nivel y tendrá la opción de **jugar un nuevo nivel**, redirigiéndose nuevamente a la **ventana de juego** en fase de preparación.

3. Mecánicas de *DCCumbia*

DCCumbia contiene ciertas mecánicas clave que deben ser implementadas para un correcto funcionamiento del programa. En esta sección se explica el funcionamiento de las principales mecánicas del juego:

3.1. Pasos de baile

Durante el transcurso del nivel aparecerán **flechas** con distintas direcciones en la **zona de ritmo** de la **ventana de juego** e irán descendiendo en la interfaz. El jugador deberá atraparlas en el momento que colisionen con la **zona de captura** para generar un paso de baile y así ganar la aprobación del público.

Los pasos a realizar se compondrán de un grupo de **flechas** que deberán aparecer en la parte superior de la pantalla de juego e ir descendiendo hasta la **zona de captura**. Ahí, deberás apretar las teclas correspondientes en el teclado, presionando simultáneamente las teclas en el caso de que haya más de una en la fila, y según esto determinar si se acertó a las flechas. Cada flecha acertada le dará puntos al jugador. Un paso de baile se considera correcto si se aciertan a todas las flechas del paso sin realizar movimientos extra, por otro lado se considera incorrecto si al presionar alguna tecla no se cumple la condición anterior. En la sección Interacción del usuario con *DCCumbia* se explica con detalle cómo debes verificar si se aciertan a las flechas y la correctitud de los pasos de baile.

Hay 2 tipos de pasos: normales y combinados. Los **pasos normales** están compuestos por solo una tecla mientras que los **pasos combinados** están compuestos por dos o más teclas presionadas en conjunto. La selección de qué paso se deberá seguir se genera ~~entrenando una red neuronal a partir del ritmo de la canción seleccionada~~ de manera aleatoria usando el módulo `random`.

Cada paso de baile se traduce en una animación de movimiento del personaje en la pista de baile junto con todos sus compañeros de baile (¡todos bailan coordinados!). Los pingüirines deben ser capaces de mostrar pasos de baile distintos asociados a las direcciones de las flechas y posibles combinaciones de ellas. Para lograr ésto puedes usar los *Sprites* disponibles en los archivos entregados.

3.2. Tipos de Flechas

Las flechas normales son las más comunes en el juego. Se desplazan a una velocidad de `VELOCIDAD_FLECHA` pixeles por segundo[1] y se traducen en un puntaje de `PUNTOS_FLECHA` al final del nivel si son capturadas.

[1] Por ejemplo, si actualizas la posición de la flecha cada 0,1 segundos, en cada actualización la posición debe variar en 0,1 × `VELOCIDAD_FLECHA`, de éste modo tras 10 actualizaciones (1 segundo) la flecha se movió `VELOCIDAD_FLECHA` pixeles.

La probabilidad de aparición de ésta flecha es `PROB_NORMAL`[2].

Además de las flechas normales existen flechas especiales que podrán aparecer de forma aleatoria durante un nivel, las flechas especiales se subdividen en:

- **Flechas x2:** Como su nombre lo indica, estas flechas dan el doble de puntos al ser atrapadas. Se mueven a la velocidad normal y la probabilidad de aparición de ésta flecha es `PROB_FLECHA_X2`[2].
- **Flecha dorada:** Esta flecha viaja un `50 %` más rápido y si se atrapa se obtiene `10` veces el puntaje normal. La probabilidad de aparición de esta flecha es `PROB_FLECHA_DORADA`[2].
- **Flechas hielo:** Cuando esta flecha es atrapada la velocidad de **todas las flechas** se reduce a la mitad durante `0.2` $\times$ *duración_nivel* segundos. Se mueven a la velocidad normal y la probabilidad de aparición de esta flecha es `PROB_FLECHA_HIELO`[2].

Entre los *Sprites* podrás encontrar múltiples *sprites* para flechas, cada tipo de flecha debe ser distinguible por su apariencia en la interfaz. Recuerda que debes explicar cómo distinguir las flechas de tu programa en tu `README` (por ejemplo: las flechas normales son verdes...).

3.3. Combo

Al realizar múltiples pasos consecutivos se consigue un **combo** como bonificación que será considerada en el cálculo del **puntaje**. El valor del **combo** parte en `0` al comienzo de cada nivel y aumenta en `1` cada vez que se realiza un paso correcto. Por otro lado, si se realiza un paso incorrecto, el valor del *combo* vuelve a `0`. El máximo valor de combo alcanzado en el nivel será el considerado en la bonificación al momento de calcular el Puntaje.

3.4. Dificultad

DCCumbia tiene 3 niveles de dificultad: **Principiante, Aficionado y Maestro cumbia**. A continuación se describen sus diferencias:

- **Principiante:** En este nivel, las partidas duran `30` segundos, y se genera un nuevo paso cada `1` segundo. Como recién te estás iniciando en este grandioso estilo de baile, solo sabes 4 pasos (las 4 flechas del teclado, sin combinaciones entre ellas). Además, para superar un nivel se requerirá una aprobación de al menos `30`.
- **Aficionado:** Ahora que ya estás dentro en el mundo de la *DCCumbia*, estás listo para bailar por más tiempo y mejor. En este nivel, las partidas duran `45` segundos, generando un paso cada `0.75` segundos. Además, ¡Ahora puedes hacer nuevos pasos! (Podrán aparecer en tu zona de ritmo combinaciones de hasta 2 flechas al mismo tiempo). Para superar un nivel en esta dificultad se requiere una aprobación de al menos `50`.
- **Maestro cumbia:** Luego de años de práctica, haz logrado llegar a este nivel. Aquí, las cosas se ponen más complicadas: las partidas duran `60` segundos, y se genera un nuevo paso cada `0.5` segundos. Además, ahora tus habilidades cumbieras no tienen límite: puedes hacer pasos de hasta 3 flechas al mismo tiempo. Para superar un nivel en esta dificultad se requiere una aprobación de al menos `70`.

3.5. Puntaje

DCCumbia posee una agrupación de críticos y críticas de baile de alto nivel. Se dedican a revisar cada paso de baile que das en la pista y al terminar la canción darán una **aprobación** que determinará si puedes

[2]Debes considerar que la suma de estas probabilidades debe ser igual a 1.

seguir bailando o serás expulsado de la pista de baile. Además, con cada nivel completado se obtendrá un puntaje acumulado que te permitirá subir en el *ranking* del *DCCumbia*.

Al terminal el nivel se calculará la **aprobación** dada por el porcentaje de acierto. Si en un nivel se generaron `pasos_totales` pasos, considerando los pasos correctos e incorrectos que haya realizado el jugador la aprobación corresponde a la formula:[3]

$$aprobacion = \left\lfloor 100 \times \frac{pasos\_correctos - pasos\_incorrectos}{pasos\_totales} \right\rfloor$$

La aprobación determina si el juego acaba o continúa, considerando que el jugador solo podrá pasar al siguiente nivel si la aprobación es superior a la que exige la dificultad. Al final de cada ronda se obtendrá una cantidad de puntos determinada por las siguientes fórmulas:

$$puntaje = max\_combo \times suma\_flechas \times \texttt{PUNTOS\_FLECHA}$$

$$suma\_flechas = (flechas\_normales + 2 \times flechas\_x2 + 10 \times flechas\_doradas + flechas\_hielo)$$

Donde:

- $flechas\_\{tipo\}$: corresponde al número de flechas acertadas de cada tipo (se consideran acertadas si se capturan incluso si el paso completo al que pertenecían fue incorrecto).
- max_combo: corresponde al máximo valor del combo alcanzado en el nivel.

La cantidad de puntos obtenidos en la ronda se suma a los **puntos acumulados** y la misma cantidad de puntos se obtienen en **dinero** para realizar compras en la fase de preparación.

3.6. Fin del nivel

Una vez transcurrido el tiempo del nivel se dejarán de generar nuevas flechas y el nivel terminará cuando ya no queden flechas en pantalla, sea porque el jugador las capturó o porque pasaron de largo sin ser capturadas. En éste momento dejará de reproducirse la canción y se mostrará la **ventana de resumen** con las estadísticas del nivel.

3.7. Fin del juego

Si al terminar un nivel la aprobación del jugador no supera el mínimo, en la ventana de resumen se mostrará un mensaje que indique que ha perdido, (junto con las estadísticas del nivel que se explican en la sección Interfaz gráfica) y el jugador ya no podrá jugar un nuevo nivel. El nombre de usuario y el puntaje acumulado en todas las rondas se deberán almacenar en el archivo `ranking.txt` y el jugador tendrá la opción de regresar a la **ventana de inicio** para iniciar una nueva partida.

4. Interfaz gráfica

4.1. Modelación del programa

Para la correcta modelación de *DCCumbia* se deberán tomar en cuenta distintos aspectos, dentro de los cuales se evaluarán:

- Correcta **modularización** del programa, lo que incluye una adecuada separación entre ***back-end*** y ***front-end***, y un **diseño cohesivo** con **bajo acoplamiento**.

[3] La función `floor` la puedes encontrar en la librería `math`.

- Correcto uso de **señales** y ***threading***[4] para modelar todas las interacciones en la interfaz.
- Presentación la información y funcionalidades pedidas (puntajes o avisos, por ejemplo) a través de la **interfaz gráfica**. Es decir, **no se evaluarán** *items* que solo puedan ser comprobados mediante la terminal o por código a menos que se explicite lo contrario.

4.2. Ventanas

Dado que *DCCumbia* se compone de diversas etapas, tendrás que implementar varias ventanas que representen las secciones relevantes del programa y así lograr que el flujo principal se pueda seguir correctamente.

Tu programa deberá contener como mínimo las ventanas indicadas a continuación. Es importante destacar que los *sprites* utilizados y la distribución de elementos dentro de cada ventana quedará completamente a tu criterio, siendo los ejemplos mostrados en esta sección simplemente una ayuda para que tengas una idea de lo que se espera de tu interfaz.

4.2.1. Ventana de Inicio

Esta ventana es la primera que se debe mostrar al jugador al ejecutar el programa. Debe incluir como mínimo:

- Un área para ingresar el **nombre del jugador**, el cual será utilizado más adelante para guardar su puntaje.
- Una opción para **comenzar el juego**.
- Una opción para **Ver el ranking de puntajes**.

En caso en que se seleccione comenzar el juego, el programa deberá verificar si el nombre ingresado es **alfanumérico**. Si el nombre es válido, se cerrará la Ventana de Inicio para dar paso a la **Ventana de Juego**. En caso contrario, se deberá mostrar una alerta[5] en la ventana señalando el error y no permitir comenzar el juego.

Si se elige ver el *ranking* de puntajes, se deberá mostrar la **Ventana de *ranking***. Quedará a tu criterio si en este caso se cierra o mantiene abierta la Ventana de Inicio.

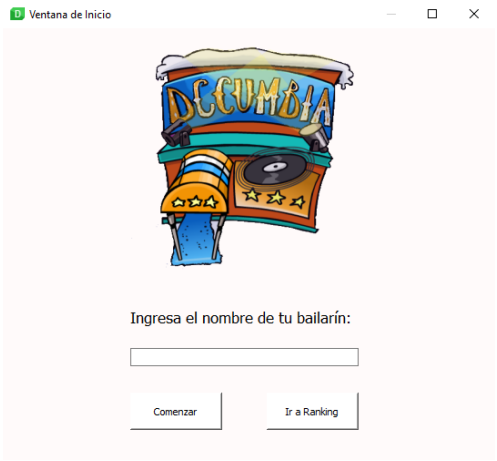


Figura 2: Ejemplo de la ventana de inicio de *DCCumbia*

[4] Esto considera el uso de elementos de *threading* de `PyQt5`, como `Qthreads` o `Qtimer`, entre otros.

[5] Algunas opciones son utilizar un `QLabel` o un `QMessageBox`. Sin embargo, puedes utilizar otro *widget* que cumpla con el objetivo de alertar al usuario de su error mediante la interfaz.

4.2.2. Ventana de Juego

Esta corresponde a la ventana en que se desarrollará el juego. Consta de **cuatro áreas**, las cuales entregarán todas las funcionalidades requeridas para una partida de *DCCumbia*.

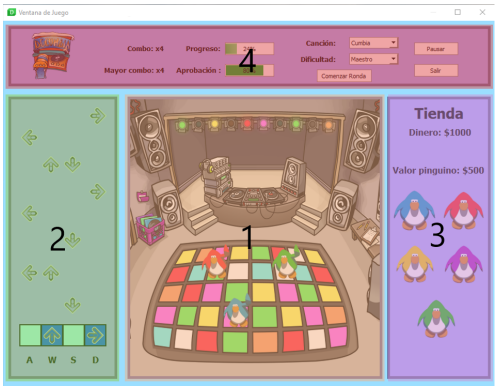
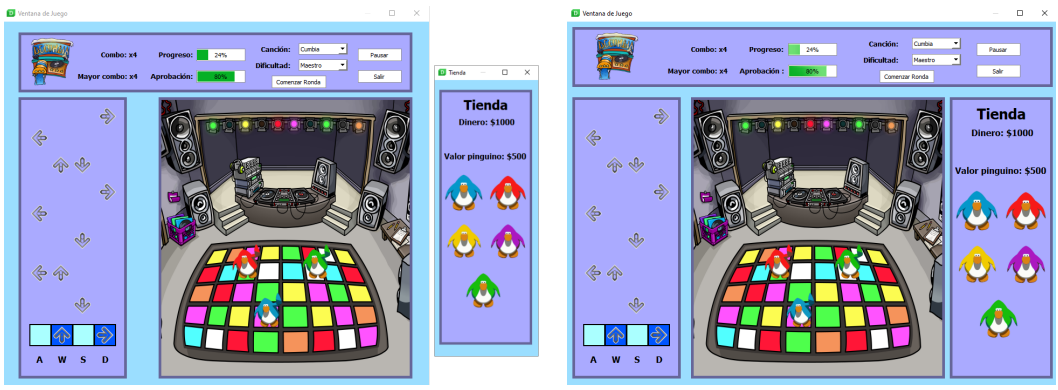


Figura 3: Áreas de la ventana de juego de *DCCumbia*

La primera área representa la **pista de baile**, en la cual se ubicarán los **pingüirines** y se deberán observar sus movimientos debido a la Interacción del usuario con *DCCumbia*. La segunda área representa la **zona de ritmo**, en la cual deberás mostrar las flechas del juego junto con la zona de captura.

La tercera área representa a la **tienda**, sección en la cual se podrán comprar pingüirines que bailen por un valor de `PRECIO_PINGUIRIN`. La tienda estará activa solo durante la **fase de pre-ronda** y el usuario podrá interactuar con ella a través de *drag and drop*, moviendo los pingüirines desde la tienda hasta la pista de baile si posee el dinero suficiente. Esta área se puede implementar de dos maneras distintas: la primera forma es en una ventana separada que interactúe con la ventana de juego (Figura 4a), mientras que la segunda forma es implementar la tienda en la misma ventana del juego (Figura 4b).



(a) Ventana de juego con tienda separada (b) Ventana de juego con tienda integrada

Figura 4: Ejemplos de como implementar la tienda en la ventana de juego

Por último, la cuarta área representa la **zona de estadísticas y de elección de ronda**. En esta parte se deberán mostrar las siguientes estadísticas de la partida, las cuales deberán mantenerse actualizadas a medida que progrese el juego:

- Progreso de la canción.
- Porcentaje de aprobación.

- Combo actual.
- Mayor combo obtenido.

También dentro de esta última área deberás incluir un **selector de canciones** y un **selector de dificultad para la ronda**, junto con un botón para comenzar la ronda. Se deberá contar con una manera de **salir del juego** ya sea con un botón y/o cerrando la ventana manualmente lo cual provocara que se vuelva a la ventana de inicio guardando el puntaje que el jugador lleve en ese momento. Deberás especificar en tu `README.md` las opciones que implementaste. Además, deberás incluir un botón para **pausar el juego**.

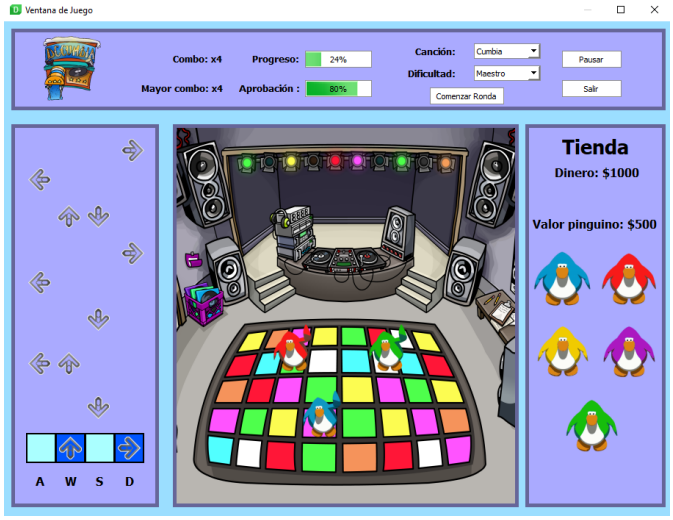


Figura 5: Ejemplo de la ventana de juego de *DCCumbia* con todos sus elementos

La ventana de juego de *DCCumbia* cuenta con tres fases distintas, que dictarán cuales de las áreas anteriores estarán activas en un momento dado. Las fases son:

- **Fase de pre-ronda:** Esta es la primera fase del juego y es el único momento en donde la tienda estará activa para comprar pingüirines. El **primer** pingüirín que compres no tendrá costo asociado. Además se podrá elegir una canción y dificultad para la próxima ronda. Una vez que el jugador esté listo podrá seleccionar el botón de comenzar y dar inicio a la **fase de ronda**. El programa debe permitir pasar de fase si el jugador tiene al menos un pingüirín, la canción a reproducir y la dificultad de la ronda. Si lo anterior no se cumple, se deberá mostrar una alerta al jugador en alguna parte de la ventana y no permitir el cambio de fase.
- **Fase de ronda:** Esta es la fase en que se desarrollan las mecánicas del juego. Deberá reproducirse la canción seleccionada junto con la aparición de las flechas. En esta fase **no** se deberá poder interactuar con la tienda, con las opciones para seleccionar una canción o dificultad ni con el botón para comenzar una ronda. Cuando se acabe la duración de la ronda y no queden más flechas, se debe terminar con la ronda y pasar a la **fase de post-ronda**.
- **Fase de post-ronda:** En esta fase se deberá mostrar en una nueva ventana un resumen de la última ronda, que deberá contener las siguientes estadísticas:
  - Puntaje acumulado.
  - Puntaje de la ronda.
  - El máximo combo obtenido en la ronda.
  - Cantidad de pasos fallados.

- El porcentaje de aprobación.

Si el jugador alcanza el porcentaje mínimo de aprobación, se deberá incluir un botón que permita **continuar jugando** y volver a la **fase de pre-ronda**. Si el jugador no alcanza el mínimo porcentaje de aprobación, se le deberá informar que ha perdido y se deberá incluir un botón para volver a la **Ventana de Inicio**, guardando el puntaje del jugador.

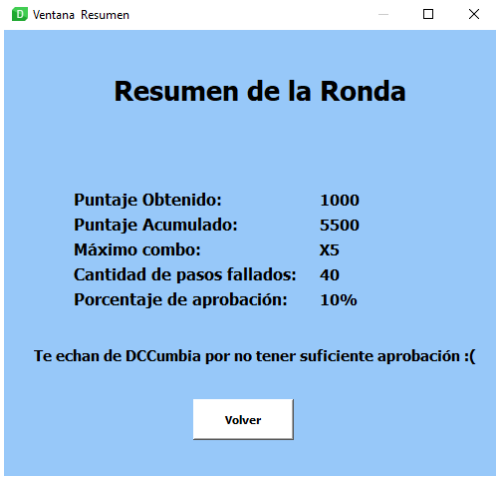


Figura 6: Ejemplo de como mostrar la información en la fase de post-ronda

4.2.3. Ventana de Ranking

En esta ventana se deberán poder ver los 5 mejores puntajes guardados en `ranking.txt`, ordenados de manera **decreciente**. También deberás implementar una forma de volver a la **Ventana de Inicio**, ya sea cerrando la ventana manualmente o utilizando un botón dentro de la interfaz.

Figura 7: Ejemplo de la ventana de ranking de *DCCumbia*

5. Interacción del usuario con *DCCumbia*

DCCumbia posee diversas formas mediante las cuáles el jugador puede interactuar, dependiendo de la funcionalidad y situación. A continuación se presentan las diferentes interacciones presentes en el juego:

5.1. *Click*

Para moverse entre ventanas, o interactuar con ciertos elementos de la interfaz, como botones o selectores **drop-down** necesarios para preparar una ronda, el jugador deberá hacer *click* sobre dichos elementos.

5.2. Atrapar Flechas

El jugador podrá atrapar las flechas cuando estas se encuentren dentro de la zona de captura. Para esto deberá presionar la tecla correspondiente, teniendo como opciones `FLECHA_ARRIBA`, `FLECHA_ABAJO`, `FLECHA_IZQUIERDA` y `FLECHA_DERECHA`.

Figura 8: Ejemplo de flechas con sus respectivas teclas.

Una flecha es acertada si hay una sección de esta dentro de la zona de captura[6] en el momento en que el jugador presiona la tecla, para esto considera que la flecha tiene una altura `ALTO_FLECHA` y la zona de captura `ALTO_CAPTURA`. Deberás cambiar el color de la zona de captura correspondiente a la tecla presionada, ya sea si la jugada es correcta o no (ver la Figura 9), y las flechas acertadas deben desaparecer de la pantalla. Una vez que se deje de presionar la tecla, debe volver a su color original.

Por último, un paso de baile se considera correcto si se aciertan a todas las flechas del paso sin presionar teclas extra.

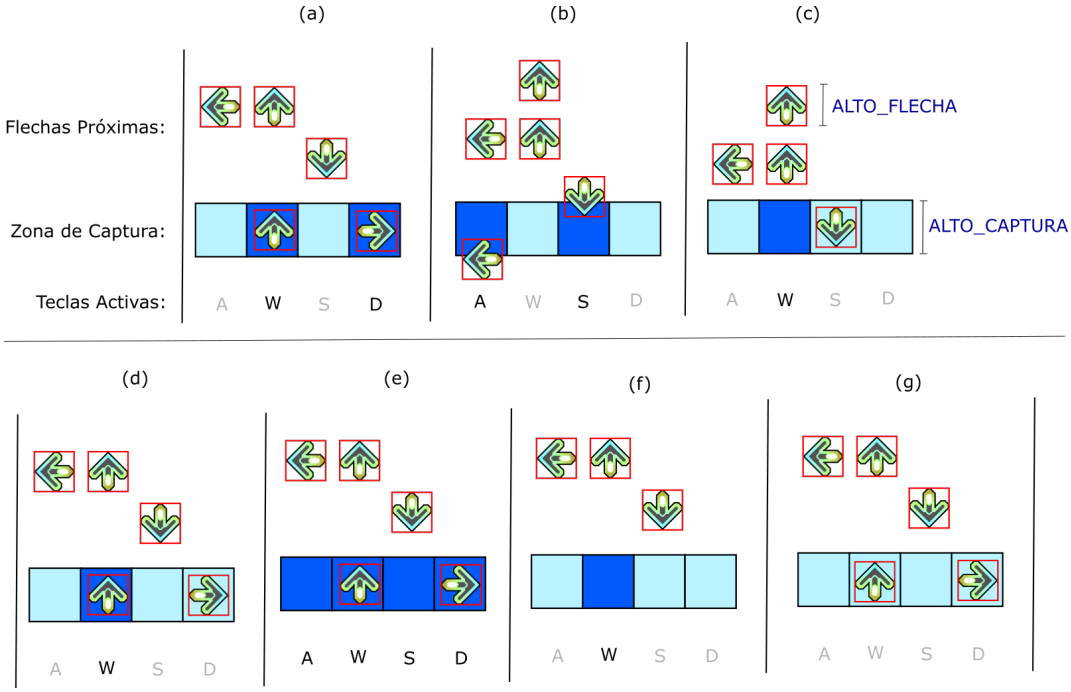


Figura 9: Ejemplo distintos pasos de baile

[6]Una forma para verificarlo es utilizando la clase `QRect` de `QtCore` y el método `intersects`. Todo `QWidget` posee un `QRect` al que puedes acceder con el método `geometry`.

A partir de los distintos pasos que aparecen en la Figura 9, podemos clasificar los distintos casos como correcto, incorrecto o no realizado. A continuación se muestra cada uno de ellos:

(a) **Paso correcto:** el jugador realiza un paso combinado *arriba-derecha* correctamente.

(b) **2 Pasos correctos:** el jugador intenta realizar los pasos *abajo* y *izquierda* (son dos pasos simples y distintos ya que están en distintas filas) y las dos flechas presentan una sección en la zona de captura.

(c) **Paso incorrecto:** el jugador intenta realizar el paso *arriba* cuando debía ser *abajo*.

(d) **Paso incorrecto:** se acierta a solo una de las flechas del paso.

(e) **Paso incorrecto:** se aciertan a las dos flechas del paso pero se aprietan teclas extras.

(f) **Paso incorrecto:** se intenta realizar un paso cuando no correspondía ninguno.

(g) **Paso no realizado:** el jugador no realiza el paso y las flechas siguen de largo, no se considera correcto ni incorrecto.

5.3. Movimiento pingüirines

Los pingüirines deben contar con una **posición neutra**, la cual toman cuando no están haciendo ningún paso de baile. Cada vez que se haga un paso nuevo, los pingüirines deberán **cambiar su *sprite*** para reflejar el movimiento efectuado. Para un movimiento más fluido, entre cada paso de baile distinto, los pingüirines deberán pasar primero por su posición neutra antes de efectuar el paso nuevo.

5.4. Pausa

Es importante contar con un botón de pausa que al ser presionado pause o continúe el juego, el cual también debe ser activable con la tecla `P`. El juego debe estar visualmente pausado, sin ocultar sus elementos. Durante la pausa las flechas deberán dejar de moverse, la música debera detenerse, y los pingüirines deberán parar, como también la barra de progreso no deberá aumentar. Una vez que se vuelva a presionar la tecla `P` o se haga *click* sobre el botón de pausa, todos los procesos reanudarán lo que estaban haciendo, sin reiniciar ni perder la información de la partida actual.

5.5. *Cheatcodes*

Con la finalidad de ~~facilitar la corrección~~ mejorar la experiencia de juego, es posible presionar un conjunto de teclas en orden, o al mismo tiempo (queda a tu criterio)[7], para hacer "trampa":

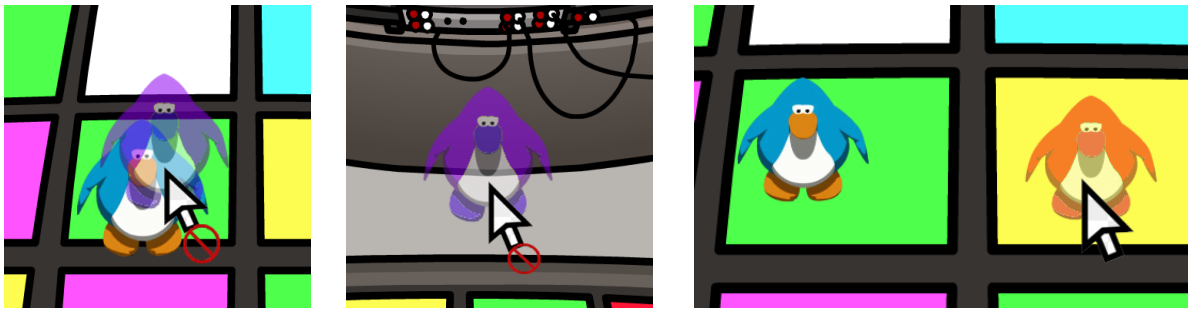
- **M + O + N**: Esta combinación de teclas aumenta tu dinero en `DINERO_TRAMPA`.
- **N + I + V**: Esta combinación de teclas termina el nivel actual, lo que hace que la canción termine y que las flechas dejen de aparecer. Si se usa estando fuera de un nivel, no tendrá efecto. Por otro lado, el puntaje obtenido, el acumulado, el máximo combo, la cantidad de pasos fallidos y el porcentaje de aprobación deberán calcularse con el progreso logrado hasta la última flecha atrapada antes de utilizar la combinación.

5.6. *Drag and drop*

Para conseguir nuevos pingüirines, deberás usar *drag and drop* para **arrastrarlos** desde la tienda a su posición deseada en la ventana de juego. Ten en consideración que la posición en la que quedan se mide usando la esquina superior izquierda de la entidad. Al soltarlos, estos deben quedar en una **posición**

[7] Deberás especificarlo en tu `README.md`.

válida, es decir, no puede quedar por encima de otro pingüino[6] o fuera de la pista de baile (Ver figura 10). En caso de que se intente lo anterior, el bailarín volverá a la tienda sin descontar dinero.



(a) Posición invalida por colisión

(b) Posición invalida por salir de la pista de baile

(c) Posición válida

Figura 10: Ejemplo de *drag and drop* en distintas posiciones

6. Archivos

Con el fin de que puedas poner en marcha a *DCCumbia*, deberás hacer uso de los siguientes archivos:

- Los elementos visuales, que están contenidos en la carpeta `sprites` y que se encuentran descritos en *Sprites*. Puedes usar tanto los sprites entregados junto al enunciado, o crear tus propios sprites para la tarea.
- Los archivos de audio, que están contenidos en la carpeta `songs`, descritos en *Songs*.

Adicionalmente, será tu deber:

- Generar y actualizar un archivo `ranking.txt`.
- Crear y utilizar un `parametros.py`.

6.1. *Sprites*

Para que puedas representar todos los elementos gráficos que *DCCumbia* posee, esta carpeta contiene múltiples imágenes en formato PNG que te serán útiles. En esta encontrarás:

- **`pinguirin_{color}`:** En estas sub carpetas encontrarás los *sprites* de movimiento para pingüirines de diversos colores. Los colores disponibles son amarillo, celeste, morado, rojo y verde. El nombre de cada imagen indicará el paso de baile al que corresponde.

Figura 11: Ejemplos de *sprites* para pingüirines celeste.

- **figuras_ritmo:** Esta carpeta contiene *sprites* útiles para la **pista de ritmo**, como flechas, números y efectos, entre otros:

Figura 12: Ejemplo número, flecha y efecto para flechas

- **`fondo.png`** imagen que representa el fondo de la pista de baile de *DCCumbia*.

Figura 13: Ejemplo pista de baile

6.2. *Songs*

Para que puedas animar el ambiente de *DCCumbia*, esta carpeta contiene 2 canciones en formato `.wav`, que deberás reproducir en tu programa durante cada ronda. Ambas canciones tienen aproximadamente **90 segundos de duración**, por lo que deberás ajustar el tiempo de reproducción de estas acorde a la duración de las rondas, deteniéndolas cuando una ronda haya terminado.

La reproducción de estas canciones se deberá hacer mediante la librería `PyQt5`[8], por lo que el uso de cualquier otra librería de python para reproducirlas será considerado como **inválido** y no obtendrá el puntaje respectivo.

Puedes probar tu tarea usando tus propias canciones, sin embargo **debes ignorar todos los archivos con formato `.wav` al momento de subir tu tarea a tu repositorio remoto.**

6.3. `ranking.txt`

Este archivo se encargará de mantener un registro de todos aquellos jugadores, que han tenido el honor de jugar una partida de *DCCumbia*. Para ser mas precisos, por cada partida se debe almacenar una fila con el formato **usuario, puntaje**, los cuales podrás almacenar en el orden que estimes conveniente. Tu tarea debe encargarse de **crear** el archivo en caso de no existir al momento de ejecución. Un ejemplo de los contenidos del archivo es el siguiente:

```
Roca14, 1200
Tom17, 500
kuki30, 0
```

[8] Las clases `QSound` o `QMediaPlayer` podrían serte útiles.

6.4. `parametros.py`

A lo largo del enunciado se han ido presentando distintos números y palabras en `ESTE_FORMATO`, estos son conocidos como **parámetros** del programa y son valores que permanecerán constantes a lo largo de toda la ejecución de tu código.

En este archivo se deben encontrar todos los parámetros mencionados en el enunciado, además de todos los *paths* y cualquier otro valor constante que vayas a utilizar en tu código. Cada línea de este archivo debe almacenar una constante junto a su respectivo valor.

Asegúrate que los nombres de las constantes sean descriptivos y fáciles de reconocer. Finalmente, este archivo debe ser importado como un módulo y así usar sus valores almacenados. En caso de que no se especifique el valor de un parámetro en el enunciado, deberás asignarlo a tu criterio, procurando que no dificulte la interacción con el programa.

Es posible que los ayudantes modifiquen estos parámetros durante la corrección, pero los que hagan referencia a dimensiones de ventanas, dimensiones de elementos de la interfaz o rutas a *sprites* **no serán modificados** para no complicar la visualización de estos. Sin embargo, debes agregarlos al archivo de igual modo. A continuación se presenta un ejemplo de cómo parametrizar rutas a archivos:

```
CANCIONES = {
    "CANCION_1" : ["ruta", "relativa", "a", "cancion.wav"],
    # Esto sería para almacenar la ruta "ruta/relativa/a/cancion.wav".
    ... # Completar con resto de canciones
}
```

7. *Bonus*

En esta tarea habrá una serie de *bonus* que podrás obtener. Cabe recalcar que necesitas cumplir los siguientes requerimientos para poder obtener *bonus*:

1. La nota en tu tarea (sin bonus) debe ser **igual o superior a 4.0**[9].
2. El bonus debe estar implementado **en su totalidad**, es decir, **no se dará puntaje intermedio**.

Finalmente, la cantidad máxima de décimas de *bonus* que se podrá obtener serán 8 décimas. Deberás indicar en tu `README` si implementaste alguno de los bonus, y cuáles fueron implementados.

7.1. Flecha espacio (2 décimas)

Nadie puede considerarse un *Cumbia Master* si no es capaz de hacer el **ultra mega paso combinado**. Este paso se realiza con un nuevo tipo de flecha, la **flecha espacio**.

Esta flecha no tiene una dirección y ocupa el ancho de las cuatro flechas. A diferencia de las otras flechas debe ser atrapada con la barra espaciadora dando `4` veces los puntos de una flecha normal. La probabilidad de aparición de esta flecha es `PROB_FLECHA_ESPACIO`[2]. Debes tener en consideración que cuando aparece esta flecha no pueden aparecer otras en la misma fila.

Como parte del bonus deberás crear un *sprite* (o utilizar uno de libre uso de internet) que represente esta flecha. Éste archivo no debe ser ignorado por lo que no debe ubicarse en la carpeta `sprites` (ver sección `.gitignore`).

[9] Esta nota es sin considerar posibles descuentos de cualquier tipo.

7.2. *Cheatcodes* adicionales (2 décimas)

Te aburres de que la suerte no te acompañe durante tus rondas de *DCCumbia* y nunca te aparezcan flechas especiales. Para contrarrestar esto, decides implementar **3 *cheatcodes* más**:

- **F + T:** Al presionar esta combinación, harás que automáticamente descienda una **Flecha x2** en la zona de ritmo.
- **F + G:** Al presionar esta combinación, harás que automáticamente descienda una **Flecha dorada** en la zona de ritmo.
- **F + H:** Al presionar esta combinación, harás que automáticamente descienda una **Flecha hielo** en la zona de ritmo.

Para cualquiera de estos *cheatcodes*, la dirección de la flecha será determinada de forma **aleatoria** entre las 4 existentes. Además, siempre aparecerá **una sola flecha** por fila al utilizar estos comandos especiales.

7.3. Segundos gratis (2 décimas)

Es normal que tus dedos se cansen tras jugar muchas horas de *DCCumbia*. Para darle un merecido descanso a tus manos sin tener que perder el título de *Cumbia Master*, decides implementar los **segundos gratis**, donde el programa jugará por tí.

Para este bonus, deberás implementar en la interfaz un botón con nombre **segundos gratis**, el cual solamente estará habilitado durante las **fases de ronda**. Al presionar este botón, se activará por `TIEMPO_GRATIS` segundos el periodo de segundos gratis. Durante este periodo, todas las flechas que pasen por la zona de captura serán atrapadas automáticamente por el programa, otorgándote el puntaje correspondiente. Las celdas de la zona de captura deberán comportarse tal como si un jugador estuviera interactuando con el juego, cambiando de color para reflejar que la flecha está siendo capturada. Una vez finalizado el periodo de tiempo, el jugador volverá a tener control completo del las teclas.

El botón de segundos gratis podrá utilizarse solo **una vez por partida**, y deberá inhabilitarse por el resto de la partida una vez presionado.

7.4. *Puffles* (4 décimas)

Tras estar meses en cuarentena encerrados en casa, los pingüirines han formado una conexión tan profunda con sus mascotas, los ***puffles***, que han decidido llevarlos a participar de *DCCumbia*.

Para este bonus deberás implementar en la **ventana de juego** una sección similar a la **tienda** en donde se presenten los distintos *puffles* junto al precio de adquirir uno, determinado por `PRECIO_PUFFLE`. La compra de un *puffle* debe manejarse igual que la de un pingüirín, debiendo arrastrarse el *puffle* a la zona de baile y siendo soltado en una ubicación válida en la zona, verificando que no colisione con otro *puffle* o pingüirín.

Estas mascotas no son tan coordinadas como sus dueños, pero poseen la misma pasión por el baile. Durante una ronda, reaccionarán solo al presionar correctamente las flechas que apunten hacia la **derecha** o **izquierda**, rotando en **45°** en la dirección de la flecha presionada respecto a su centro. Una vez soltadas las teclas, deberán volver a su posición original.

Dentro de la carpeta *sprites*, se encuentra la subcarpeta *puffles*, la cual contiene los *sprites* necesarios para visualizar los *puffles* en la interfaz.

Figura 14: Ejemplo de *sprites* para *puffles*.

8. Avance de tarea

Para esta tarea, el avance corresponderá a implementar una versión **simplificada** de la **zona de ritmo**.

En particular, deberás programar una Zona de ritmo con ancho de **1 carril**, es decir, de tamaño para una única flecha. Dicha flecha deberá descender dentro de la interfaz de forma automática, y cuando esta colisione con su **zona de captura** correspondiente, la zona de captura deberá cambiar su color mientras **se mantenga la colisión entre la flecha y la zona**. Una vez la flecha salga completamente de la zona de captura, el color de esta debe volver al original.

A partir de los avances entregados, se les brindará un *feedback* general de lo que implementaron en sus programas y además, les permitirá optar por **hasta 2 décimas** adicionales en la nota final de su tarea.

9. `.gitignore`

Para esta tarea **deberás utilizar un** `.gitignore` para ignorar los archivos indicados, este deberá estar dentro de tu carpeta `Tareas/T02/`. Puedes encontrar un ejemplo de `.gitignore` en el siguiente ***link***.

En específico, los archivos a ignorar para esta tarea son:

- Enunciado.
- Archivos de audio (canciones) en formato `.wav`.
- Carpeta de *sprites* entregada junto al enunciado en su totalidad. Si deseas añadir tus propios *sprites* adicionales, recuerda no ignorarlos.

Se espera que no se suban archivos autogenerados por las interfaces de desarrollo o los entornos virtuales de Python, como por ejemplo: la carpeta `__pycache__`.

Para este punto es importante que hagan un correcto uso del archivo `.gitignore`, es decir, los archivos no **deben** subirse al repositorio debido al archivo `.gitignore` y no debido a otros medios.

10. Entregas atrasadas

Posterior a la fecha de entrega de la tarea se abrirá un formulario de Google Forms. En caso de que desees que se corrija un *commit* posterior al recolectado, deberás señalar el nuevo *commit* en el *form*.

El plazo para rellenar el *form* será de 24 horas a partir de la entrega del código. En caso de que no lo contestes en dicho plazo, se procederá a corregir el *commit* recolectado.

11. Importante: Corrección de la tarea

Para esta tarea, el carácter funcional del programa será el pilar de la corrección, es decir, **sólo se corrigen tareas que se puedan ejecutar**. Por lo tanto, se recomienda hacer periódicamente pruebas de ejecución de su tarea y *push* en sus repositorios.

Cuando se publique la distribución de puntajes, se señalará con color:

- **Amarillo**: cada ítem que será evaluado a nivel de código, todo aquel que no esté pintado de amarillo significa que será evaluado si y sólo si se puede probar con la ejecución de su tarea.
- **Azul**: cada ítem en el que se evaluará el correcto uso de señales. Si el ítem está implementado, pero no utiliza señales, no se evaluará con el puntaje completo.

En tu archivo `README.md` deberás señalar el archivo y la línea donde se encuentran definidas las funciones o clases relacionados a esos ítems.

Se recomienda el uso de *prints* para ver los estados del sistema (progreso de la ronda, posición de las flechas, colisiones, etc.) pero no se evaluará ningún ítem por consola. Esto implica que hacer *print* del resultado una función, método o atributo no valida, en la corrección, que dicha función esté correctamente implementada. **Todo debe verse reflejado en la interfaz.** Por ejemplo, si se atrapa una flecha correctamente, que hagan *print* señalando que se capturó correctamente la flecha no será válido para la corrección, sino se evaluará que la captura correcta se refleje en la interfaz.

Finalmente, si durante la realización de tu tarea se te presenta algún problema o situación que pueda afectar tu rendimiento, no dudes en contactar al ayudante jefe de Bienestar a su correo **bienestar.iic2233@ing.puc.cl** o al correo oficial del curso **iic2233@ing.puc.cl**.

12. Restricciones y alcances

- Esta tarea es **estrictamente individual**, y está regida por el **Código de honor de Ingeniería**.
- Tu programa debe ser desarrollado en Python 3.7.
- Tu programa debe estar compuesto por uno o más archivos de extensión `.py`.
- Si no se encuentra especificado en el enunciado, supón que el uso de cualquier librería Python está prohibido. Pregunta en la *issue* especial del **foro** si es que es posible utilizar alguna librería en particular.
- Debes adjuntar un archivo `README.md` **conciso y claro**, donde describas los alcances de tu programa, cómo correrlo, las librerías usadas, los supuestos hechos, y las referencias a código externo. **Tendrás hasta 48 horas después del plazo de entrega** de la tarea para subir el `README` a tu repositorio.
- Tu tarea podría sufrir los descuentos descritos en la **guía de descuentos**.
- Entregas con atraso de más de 24 horas tendrán calificación mínima (1,0).
- Cualquier aspecto no especificado queda a tu criterio, siempre que no pase por sobre otro.

Las tareas que no cumplan con las restricciones del enunciado obtendrán la calificación mínima (1,0).