PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

**IIC2233 - Programación Avanzada (I/2016)**
**Tarea 5**

# 1. Objetivos

- Aplicar conocimientos de creación de interfaces.
- Aplicar conocimientos de threading en interfaces.

# 2. Introducción

Despiertas angustiado. Te das cuenta que el apocalipsis fue solo una pesadilla dada por el trabajo excesivo realizado en LCJ. Esta situación te hace ver que tu vida podría acabar en cualquier momento, y no quieres morir siendo un simple empleado de consultas[1]. Todo esto te motiva a tirar todo a la basura[2], y decides crear desde cero un nuevo juego[3] que te dejará a ti y a tu billetera en la cima.

Si bien partes sin ideas para tu fabuloso juego, recuerdas los malos momentos pasados dentro de tu formación como programador, y encuentras una forma de desquitarte de los personajes que hicieron de esos días una miseria: convertirlos en pedacitos. Creas, finalmente: **¡Ayudantados Trouble!**

# 3. Ayudantados Trouble

El juego consiste en una persona controlada por el usuario mediante el teclado, la que se encuentra en un espacio cerrado con piso, techo y 8 etapas distintas. En cada una, hay burbujas que de impactar al jugador, disminuye su cantidad de vidas. El jugador claramente no quiere ser impactado por ellas, por lo que cuenta con una pistola que le permite ir dividiendo las burbujas, hasta llegar a su unidad mínima y hacerlas desaparecer para pasar a la siguiente etapa.

## 3.1. Lógica del Juego

Al correr el programa, se deberá mostrar un menú que tenga al menos un botón para iniciar el juego, y otro que diferencie la modalidad a jugar (uno o dos jugadores). Además, mientras se está jugando, el usuario podrá terminar el juego en cualquier momento al apretar el botón `QUIT GAME`, o bien presionando la tecla `G`. Por otra parte, el jugador debe tener la posibilidad de pausar el juego al apretar el botón `PAUSE`, o bien presionando la tecla `P`. Notar que el jugador debe poder cerrar o pausar el juego con **ambas posibilidades** (es decir, deben implementar el botón en la interfaz, y además incluir la funcionalidad para dichas teclas).

[1] Además, necesitas un sueldo mejor... recuerda que estás en la miseria.
[2] RIP John Router. Volverá a aparecer... dentro de otros semestres.
[3] Mejor que el anticuado e incompleto SixPYx.

Además del juego normal, en donde se va avanzando en las etapas partiendo de la primera, deberás dar la opción de elegir la etapa a jugar para ambas modalidades (uno o dos jugadores). En este caso, si bien las teclas de movimiento y disparo cambian (especificado en la sección **Dos jugadores**), las que permiten pausar y salirse del juego deben ser las mismas.

## 3.2. Personaje Principal

### 3.2.1. Movimiento

El movimiento del personaje principal es horizontal, y puede ir a la izquierda o a la derecha, respetando los bordes de la ventana del juego.

### 3.2.2. Disparos

El jugador tiene una cantidad infinita de disparos. Estos salen desde el jugador de manera vertical y pueden reventar una sola burbuja a la vez. La única restricción es que para poder realizar otro disparo, se debe haber reventado una burbuja, o esperar a que este haya llegado hasta el "techo" del juego. Existe más de un tipo de disparo con características específicas, los que serán explicados en la sección **Power-Ups**.

### 3.2.3. Vida

El personaje debe contar con una cantidad inicial de vidas. Las dos formas de perder una vida son: Ser impactado por una burbuja, o que se acabe el tiempo de juego. Debes representar gráficamente la cantidad de vidas que tiene el jugador.

## 3.3. Burbujas

Cuando el personaje principal da con una burbuja, esta se debe dividir en dos burbujas más pequeñas, y así sucesivamente hasta llegar a la unidad mínima. Cada burbuja tiene una cantidad definida de niveles que se especifican para cada etapa del juego más adelante. Para la parte gráfica, cuentan con la carpeta `assets` que contiene burbujas de diferentes colores y tipos que deben utilizar para desarrollar su juego. Notar que ustedes deben escalar estas imágenes correspondientes al nivel en que se encuentre cada burbuja.

### 3.3.1. Tipos y jerarquía

Para cada color, existen tres tipos de burbujas: `color_boss`, `color_mentor` y `color_tpd`. Debes respetar la jerarquía de estas, ya que por ejemplo, una burbuja de tipo `tpd` no puede subdividirse en dos burbujas de tipo `boss`. La jerarquía es la siguiente:

- `color_boss` : La burbuja de mayor nivel en la jerarquía. Puede dividirse en dos burbujas de tipo `boss` o dos burbujas de tipo `mentor`. **No puede dividirse en dos burbujas de tipo tpd**.
- `color_mentor` : Esta puede dividirse en dos burbujas de tipo `mentor` o dos de tipo `tpd`. **No puede dividirse en dos burbujas de tipo boss**.
- `color_tpd` : Esta solo puede dividirse en dos burbujas de tipo `tpd`.

Dependiendo de la cantidad de niveles, pueden combinar los diferentes tipos respetando la jerarquía. Sin embargo, la mínima unidad siempre debe ser un tpd y si el nivel es mayor a 2, la burbuja inicial debe ser un boss. En caso de que tenga 2 niveles, la burbuja inicial debe ser un mentor y si tiene un solo nivel, debe ser un tpd.

A continuación, se muestran ejemplos de la división de burbujas. Como notarán, la primera imagen tiene una burbuja con cuatro niveles, por lo que la primera burbuja debe ser un `boss` y la última un `tpd`. En la segunda ocurre lo mismo, y en la tercera, dado que tiene solo dos niveles, la primera es un `mentor` y la segunda un `tpd`.

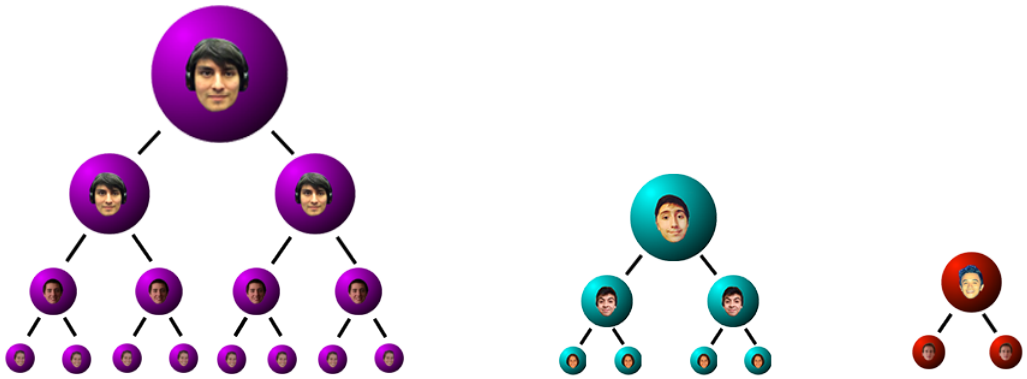

Figura 1: Ejemplo de jerarquía de burbujas.

### 3.3.2. Trayectoria

Al comienzo, las burbujas se deben dejar caer de una altura determinada por usted y deben rebotar describiendo **curvas continuas**. Puede asumir que los rebotes de las burbujas de un mismo nivel con el piso son choques perfectamente elásticos, por lo que la altura máxima de cada rebote siempre es la misma. Sin embargo, entre los distintos niveles de las burbujas las alturas sí deben ir disminuyendo. Pueden asumir que todas las burbujas tienen la misma velocidad. Por otra parte, cuando una burbuja se divide, las divisiones resultantes deben seguir trayectorias en sentidos opuestos (es decir, una debe seguir su trayectoria por la izquierda, y la otra por la derecha). A continuación se muestra un ejemplo de una posible trayectoria de burbujas.

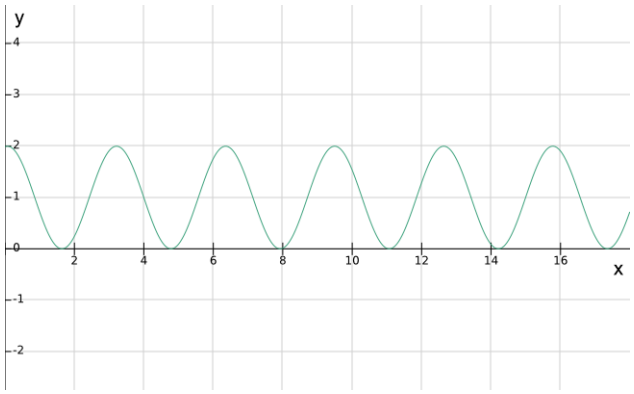


Figura 2: Ejemplo correcto de la trayectoria de las burbujas.

Un ejemplo de un modelamiento incorrecto de la trayectoria se muestra a continuación:

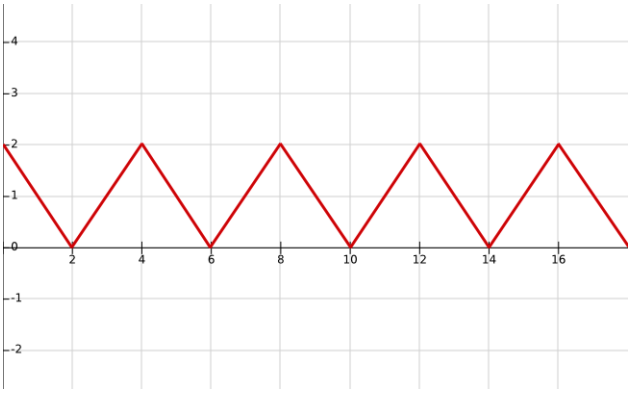


Figura 3: Ejemplo incorrecto de la trayectoria de las burbujas

### 3.3.3. Burbuja Python

Figura 4: Burbuja Python

Existe una burbuja especial que, al igual que las normales, tiene un número determinado de niveles de división. Sin embargo, un elemento diferenciador de esta es que no sigue la trayectoria definida anteriormente, y solo se puede dividir en burbujas de este mismo tipo.

La trayectoria debe ser continua y la deben definir ustedes, pero no puede ser predecible. La idea es que sea lo más aleatoria posible, de forma que el jugador no pueda adivinar su movimiento y sea una verdadera amenaza para él. A continuación, dejamos algunos ejemplos de lo que podría ser la trayectoria de esta burbuja. Deben especificar en el README cómo la modelaron. ¡Sean creativos!

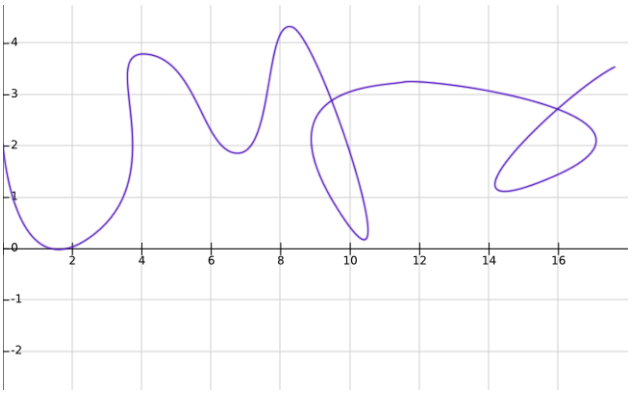


Figura 5: Ejemplo de una posible trayectoria de una burbuja Python.

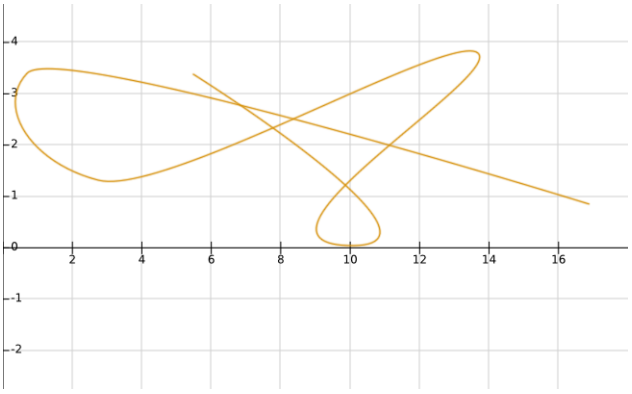


Figura 6: Otro ejemplo de una posible trayectoria de una burbuja Python.

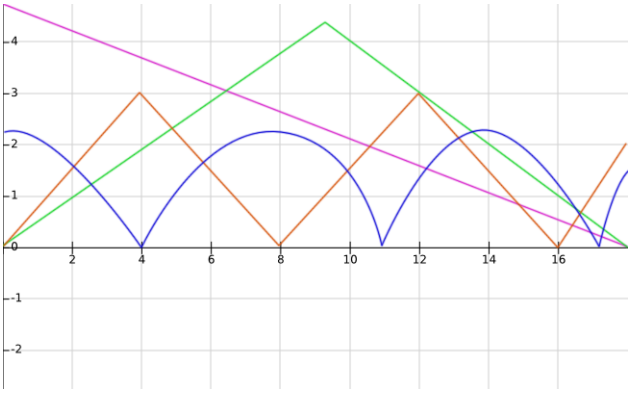


Figura 7: Ejemplos de trayectorias incorrectas de una burbuja Python.

### 3.4. Power-ups

Los power-ups ayudan y dan poderes especiales al jugador. Durante el desarrollo del juego, estos deben aparecer dada una probabilidad definida por ti. Estos pueden caer de forma repentina desde el cielo, o bien pueden aparecer tras dividir una burbuja, cayendo desde la posición donde ésta se dividió. Para que no haya confusiones, deben considerar **ambas posibilidades**. Las probabilidades respectivas, o bien los power-ups que se arrojan en cada situación quedan a su criterio. A continuación, se señalan los distintos power-ups disponibles y sus características:

- **Monedas:** Aumentan el puntaje que el jugador lleva acumulado (la cantidad a aumentar será definida por ti).
- **Billetes:** Aumentan el puntaje que el jugador lleva acumulado (a criterio también, pero debe ser mayor al puntaje dado por las monedas).
- **Vida:** Le entrega al jugador una vida más.
- **Tiempo:** El tiempo restante se ve aumentado (la cantidad a aumentar será definida por ti). Notar que si la suma entre el tiempo restante y lo que otorga el power-up es superior al límite inicial del nivel, el nuevo tiempo tiene que ser igual al límite inicial y no superior a este.

- **Disparo alambre de púas:** Le da a las balas del jugador la forma de alambre de púas. Estas tendrán la propiedad de no desaparecer de inmediato. Si el jugador dispara y la bala toca el techo, el disparo debe quedarse pegado por un tiempo definido por ustedes. El alambre resultante solo puede desaparecer de dos formas: Luego de que el tiempo límite se haya completado, o bien cuando una burbuja lo haya interceptado.
- **Disparo doble:** Permite al jugador disparar dos veces consecutivas. Si este dispara dos veces, podrá volver a hacerlo luego de que la primera bala haya tocado el techo, o bien cuando haya reventado una burbuja.
- **Escudo:** Genera una barrera protectora alrededor del jugador. Esta le permite ser impactado con una burbuja sin perder una vida. Sin embargo, luego de haber chocado con una, la barrera desaparece. Notar que luego de la colisión, la burbuja debe **rebotar** desde el jugador, cambiando el sentido de su trayectoria original. Notas que la burbuja **no se divide al colisionar con el escudo.**

Notar que si el jugador obtiene los power-ups *disparo doble* y *disparo alambre púas* de forma seguida, estos **se mezclan.** Es decir, el jugador podrá disparar dos alambres de púas continuados, y podrá volver a disparar cuando el primero de estos desaparezca (siguiendo la misma lógica explicada en su descripción). Si el jugador pierde una vida o pasa de nivel, vuelve a su condición inicial (es decir, no conserva los disparos adquiridos).

## 3.5. Etapas

Tu juego debe contar con 8 etapas. La etapa es completada cuando el jugador elimina todas las burbujas del escenario. A continuación, se especifica lo que debe tener cada una de ellas:

- **Etapa 1:** Una burbuja de 2 niveles.
- **Etapa 2:** Dos burbujas de 3 niveles.
- **Etapa 3:** Una burbuja de 4 niveles y una de 2 niveles.
- **Etapa 4:** Una burbuja de 3 niveles y una burbuja Python de 2 niveles.
- **Etapa 5:** Una burbuja de 5 niveles.
- **Etapa 6:** Dos burbujas Python de 3 niveles.

Las etapas 7 y 8 cuentan con paredes. Estas dividen la pantalla del jugador e impiden que este pueda cruzar al otro lado, hasta haber reventado todas las burbujas que se encuentran a su lado de la pared. Una vez reventadas todas las burbujas de la primera división, se deben abrir las "puertas"para poder continuar con las burbujas que se encuentren al otro lado de la pared. Una vez abierta la pared, el jugador podrá moverse sobre todo el espacio antes de la división y el que se encuentra hasta la próxima pared, o bien el borde de la pantalla.

- **Etapa 7:** Una pared, la primera sección con una burbuja de 2 niveles, y la segunda sección con una burbuja de 5 niveles.
- **Etapa 8:** Dos paredes, la primera sección con 2 burbujas de 2 niveles, la segunda sección con una burbuja de 3 niveles, y la última sección con una burbuja Python de 4 niveles.

Si el jugador pierde o logra completar las 8 etapas, se debe desplegar un mensaje que ha terminado el juego con su respectivo puntaje.

**Nota:** Queda a su criterio en qué parte de la ventana se inicializan las burbujas y de qué color son, pero deben usar, a lo largo del juego, todos los colores disponibles. Por otra parte, en el caso de las etapas 7 y 8, las burbujas que se encuentren en las secciones inicialmente cerradas **deben estar rebotando desde el principio.**

## 3.6. Colisiones

Dentro del juego, deben existir colisiones entre las siguientes entidades:

- Personaje y bordes.
- Personaje y paredes.
- Burbujas y bordes.
- Burbujas y paredes.
- Personaje y burbujas
- Personaje con escudo y burbujas.

Las colisiones entre el personaje y los bordes o paredes son para evitar que este salga de la ventana (o bien para que este no pueda pasar a una sección de la etapa sin haber reventado las burbujas necesarias). Por otra parte, las colisiones entre burbujas y bordes o paredes, además de respetar los mismos principios, deben considerar los cambios físicos para **cambiar la trayectoria de estas al momento de colisionar.** Si se tiene una colisión entre el personaje y una burbuja, deben reiniciar la etapa desde el principio, poniendo todo como en el inicio (salvo el puntaje, ese se mantiene). En cambio, si el personaje colisiona con una burbuja teniendo un escudo, este desaparece, y la burbuja cambia su trayectoria (aquí también deben considerar la física de la situación).

**BONUS:** Una colisión interesante que no se considera anteriormente, es la colisión entre burbujas. Si usted implementa este tipo de choque (tomando en cuenta las trayectorias que llevan y la altura a la que se encuentran ambas antes de chocar, y un resultado esperable físicamente), tendrá un aumento en su nota del 10 %. Debe indicar en un README si implementó esta parte o no.

## 3.7. Tiempo

Todas las etapas poseen un tiempo límite para destruir todas las burbujas. Este tiempo límite debe ser medido en segundos. El tiempo límite por etapa debe ser definido por usted y explicitado en el README. Puede dejar todas las etapas con el mismo tiempo, si así lo desea, pero siempre dando sus razones de la decisión escogida.

## 3.8. Puntaje

Se debe mantener el puntaje del jugador a medida que avanza el juego. Cada burbuja que reviente le otorgará puntos. Reventar burbujas `boss` debe dar más puntaje que reventar burbujas `mentor`, y reventar estas últimas debe dar más puntaje que reventar burbujar `tpd`. Por otra parte, el tiempo que le quedó de sobra al jugador para completar la etapa también debe ser convertido en puntaje. Si el jugador pasa todas las etapas, debe darle puntaje por cada vida que tenga disponible al terminarlo, recibir una bonificación por haber completado el juego. La cantidad exacta de puntos otorgados debe ser definida por ti y especificada en tu README. Recordar que las *monedas* también dan puntaje. El puntaje se reinicia solo cuando el jugador ha perdido todas sus vidas y reinicia el juego, o cuando se sale del mismo por su cuenta (si pierde una vida, el puntaje que lleve hasta ese momento no se pierde). En caso de haber dos jugadores, el puntaje debe ser individial para cada uno (el tiempo que queda al completar la etapa se divide equitativamente en 2).

## 3.9. Dos jugadores

Tu juego debe dar la opción de multijugador. Se mantienen todos los requerimientos y características del juego de un jugador. El juego termina cuando ambos pierden todas sus vidas, o si se presiona el botón `QUIT GAME` o la tecla `G`. Notar que si uno de los jugadores pierde sus vidas antes que el otro, el que aún tenga vidas deberá seguir jugando solo.

## 3.10. Teclado

Si se está jugando en el modo un jugador, las teclas son las siguientes:

- ← : Moverse hacia la izquierda.
- `SPC` : Emitir el disparo.
- → : Moverse hacia la derecha.

En caso de dos jugadores, las teclas son las siguientes:

- Primer jugador:
  - ← : Moverse hacia la izquierda.
  - ↑ : Emitir el disparo.
  - → : Moverse hacia la derecha.
- Segundo jugador:
  - `A` : Moverse hacia la izquierda.
  - `W` : Emitir el disparo.
  - `D` : Moverse hacia la derecha.

## 3.11. Representación Gráfica

Las burbujas deben moverse de forma continua, describiendo la trayectoria definida por ustedes. El personaje principal debe tener movimientos animados. Para lograr esto, se debe diferenciar su aspecto al caminar en distintas direcciones. Además, se deben observar al menos 3 estados de movimiento. Por ejemplo, pies juntos, pie derecho adelante y pie izquierdo adelante. De esta forma, el movimiento se verá aún más real. Para esta sección, se le recomienda buscar en internet sprites[4] para el jugador principal, que incluyan las orientaciones y estados de movimiento.

Figura 8: Ejemplo de un Sprite Sheet para el jugador principal

Debes incluir en la ventana del juego las vidas del jugador, el tiempo que le queda para completar la etapa, el tipo de disparo, la etapa en la que se encuentra y el puntaje que lleva hasta el momento.

[4] Si no los conoce, son **estos**. ¡Siéntanse libres de buscar más o de crear los suyos propios para personalizar su tarea!

# 4. Notas

- Todas las magnitudes que no están especificadas en el enunciado quedan a su criterio. Lo importante es que las magnitudes elegidas aporten al correcto desarrollo del juego. Por otra parte, **todas las constantes que hayan escogido deben ir en un archivo (`.txt`, `.csv`, `.py`, `etc`) separado a todo el programa llamado `constantes.{extensión}` e importarlo en los archivos que utilicen.** Esto nos permitirá cambiarlas desde dicho archivo y poder probar que su programa funcione de la forma esperada.
- Pueden basarse en el juego Bubble Trouble[5] (**click aquí**) a modo de ejemplo. Tome en consideración que este juego implementa muchas más cosas de las que se piden en esta tarea, así que asegúrese de leer el enunciado con detención.

# 5. Restricciones y alcances

- Tu programa debe ser desarrollado en Python 3.4
- Esta tarea es estrictamente individual y está regida por el Código de Honor de la Escuela: Clickear para Leer.
- Su código debe seguir la guía de estilos `PEP8`
- Si no se encuentra especificado en el enunciado asuma que el uso de cualquier librería Python está prohibido. Pregunte por foro si se pueden usar librerías específicas.
- El ayudante puede castigar el puntaje[6] de tu tarea si le parece adecuado. Se recomienda ordenar el código y ser lo más claro y eficiente posible en la creación de algoritmos.
- Debe adjuntar un archivo `README.md` donde se comenten los alcances y el funcionamiento de su sistema (*i.e.* manual de usuario) de forma *concisa* y *clara*.
- Cree un módulo para cada conjunto de clases. Divídalas por las relaciones y los tipos que poseen en común.
- Cualquier aspecto no especificado queda a su criterio, siempre que no pase por sobre otro.

# 6. Entrega

- **Fecha/hora:** 3 de Junio - 23:59 hrs
- **Lugar:** GIT - Carpeta: Tareas/T05

Tareas que no cumplan con las restricciones señaladas en este enunciado tendrán la calificación mínima (1.0).

[5] De una u otra forma, tus ideas se filtran y los programadores del resto del mundo siguen copiándotelas.
[6] Hasta −5 décimas.