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

**IIC2233 Programación Avanzada (II/2017)**

# Tarea 5

## 1. Entrega

- `Archivo_entrega`
  - **Fecha/hora**: 26 de octubre del 2017, 23:59 horas.
  - **Lugar:** Cuestionario en el Siding
- `Tarea`
  - **Fecha/hora:** 6 de noviembre del 2017, 23:59 horas.
  - **Lugar:** GitHub – Carpeta: `Tareas/T05/`
- `README.md`
  - **Fecha/hora:** 7 de noviembre del 2017, 23:59 horas.
  - **Lugar:** GitHub – `Carpeta: Tareas/T05/`

## 2. Objetivos

- Aplicar conocimientos de creación de interfaces.
  - Entender concepto de *back-end*
  - Entender concepto de *front-end*
- Aplicar conocimientos de *threading* en interfaces.
- Aplicar conocimientos de señales.
- Aplicar conocimientos de *slots*.

  Complementarios:
- Comprender el uso y aplicación del archivo `.gitignore`

## 3. Introducción

¡Felicitaciones! Debido al gran éxito de tu programa decodificando genomas, Hernán descubrió su triste verdad: no es hijo de Marcelo Lagos. Como la fortaleza de Hernán es, entre otras habilidades, la programación, decidió crear un juego en el que el mundo fue destruido producto de su enojo. Ahora debes completar una parte de su juego, que es el inicio de la primera parte de la vida: las células.

# 4. DCCells

En este juego, jugarás con una célula cuya meta es devorar a las demás células, las cuales se consideran tus enemigos porque también buscarán engullir a tu personaje. La alimentación se basa en la *"La ley del más fuerte"*; es decir, alimentarse de los enemigos más débiles, o pequeños, en este caso. A medida que tu célula se alimenta de las demás células, irá ganando puntos de experiencia. Estos te permitirán aumentar paulatinamente su tamaño y tener la capacidad de devorar enemigos que antes eran más grandes que tu personaje.

El juego consiste en **cinco etapas** en donde tu personaje se enfrentará a diferentes tipos de enemigos que podrás diferenciar debido a su tamaño. Habrá algunos más pequeños que tu célula, quienes intentarán huir de tu personaje; otros más grandes, quienes intentaran devorar a tu personaje; y otros de igual tamaño, quienes decidirán de manera aleatoria si atacan o escapan. A medida que el personaje principal devore a otras células, el personaje irá acumulando experiencia. Las etapas se completan cuando el personaje alcanza los 1.000 puntos de experiencia. El juego termina cuando tu personaje es comido por un enemigo o cuando terminas la última etapa.

## 4.1. Entidades [ 10 % ]

Dentro de DCCells, las únicas entidades presentes son el personaje principal y sus enemigos, quienes poseen los mismos atributos que tu personaje, pero son menos inteligentes y sólo buscan seguir las ordenes de Hernán: destruirte e impedir que reconstruyas el mundo.

Todas las entidades comparten los siguientes atributos:

### 4.1.1. Tamaños

Todas las entidades presentes en el juego poseen un tamaño definido, que tienen que ser diferenciados gráficamente. Este condiciona su velocidad de movimiento, vida máxima y ataque frente a otros individuos. En el juego existen **10 tamaños** diferentes, donde el menor tamaño es 1 y van creciendo de uno en uno hasta 10. La diferencia de tamaño de los primeros cinco se puede apreciar en la siguiente figura 1:

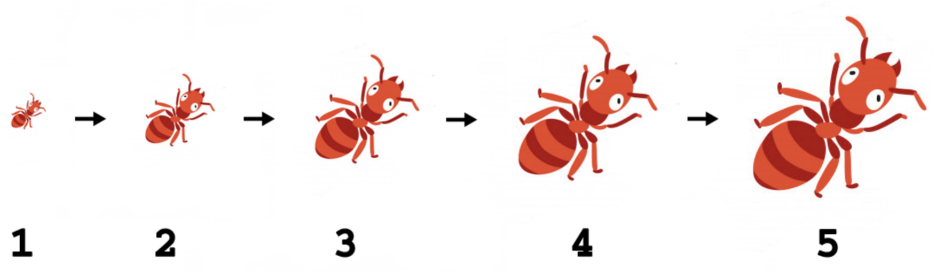


Figura 1: Cinco de los diez tamaños posibles

### 4.1.2. Vida

Cada entidad posee una **vida máxima**, que define cuánto es el daño que puede soportar antes de perecer. Esta vida máxima está condicionada a dos factores: el tamaño actual y bonificaciones externas (ver sección 4.7). De este modo, la vida máxima se define como el $tamaño\_actual \times 20 + 100 + C$, donde C son las bonificaciones externas obtenidas durante el juego. El personaje principal es el único que recibirá bonificaciones durante el juego, por lo que C = 0 para los enemigos.

Cada entidad posee también una **vida actual**. Esta es un número menor o igual a la vida máxima, que se va reduciendo a medida que se reciben los ataques del enemigo. Si la vida actual es menor o igual a cero, implica que esa entidad fue devorada por su último atacante.

Cada entidad debe poseer una forma de visualizar gráficamente la vida actual. En el caso del jugador, debe ser mediante `ProgressBar` (elemento de PyQt que representa una barra que carga) y para los enemigos también debe ser una `ProgressBar` o un `label` (elemento de PyQt que representa una etiqueta) con el porcentaje de vida actual en relación a la vida máxima. Deben restringir el número de decimales a tres para evitar saturar la interfaz.

Ejemplo: Si mi vida máxima es 200 y tengo 1 de vida, entonces si se trabaja con `labels` este mostraria "0.5 %", mientras que si se trabaja con `ProgressBar` Tendra que mostrar la barra cargada en un "0.5 %". Con esta restricción, deben tener presente que este cuociente solamente sirve para determinar la forma de visualizar la vida actual en función de la vida máxima. Este valor no debe ser usado para otros cálculos.

#### 4.1.3. Ataque

Dentro del juego, el ataque se define como el intento de una célula por comer a otra y este está determinado por la colisión de las células en la interfaz. Las células enemigas no se pueden atacar entre sí; en otras palabras, la colisión entre dos enemigos no produce ningún efecto. Sólo una colisión entre el personaje principal y los enemigos es considerado como un ataque.

El ataque se debe ver reflejado en la interfaz mediante la reducción de vida de los participantes. Este mundo se rige por las leyes de Newton, en especial por la tercera ley: "A cada acción siempre se opone una reacción igual". Esto implica que una colisión entre el personaje principal y el enemigo hará que ambos se intenten atacar. Cuando uno de los participantes de la pelea tiene su vida en 0, entonces se dice que muere y el ganador se lo come.

El ataque de cada célula está definido en función de sus atributos como:

$$\text{ataque} = \texttt{round(}\mathit{tamaño\_celula} \times \tfrac{1}{10} \times \mathit{vida\_maxima}\texttt{, 0)}$$

El ataque tiene que ser redondeado a un número entero. Podrás notar que el daño es realizado en función de cuánta es la vida máxima que tiene el atacante y su tamaño. Por lo tanto, si el personaje principal ataca a alguien más grande que él, esa otra célula enemiga también lo hará, pero el ataque que le haga el personaje principal será menor que el ataque que la célula enemiga le haga al personaje.

Por ejemplo, si el personaje principal es de tamaño 5 con vida máxima 200 y el enemigo es de tamaño 6 con vida máxima 220, entonces, el ataque del personaje principal es de 100, mientras que el ataque del enemigo es de 132.

#### 4.1.4. Velocidad de ataque

En la sección anterior se definió el ataque como el efecto de colisión entre el personaje principal y el enemigo. Este ataque se puede realizar cada un segundo. En otras palabras, después de colisionar y atacar a una célula, ambas células involucradas deben esperar un segundo antes de volver a atacar. Este atributo se puede aumentar para el personaje principal mediante bonificaciones externas; es decir, se puede reducir el tiempo entre un ataque y otro.

## 4.2. Personaje principal [ 10 % ]

### 4.2.1. Nivel de personaje

El personaje principal tiene un nivel que está asociado a la etapa en la que se encuentra. Para subir de nivel, debes completar 1.000 puntos de experiencia, los cuales obtendrás a medida que devores a tus enemigos. Se debe mostrar el nivel y el progreso actual de este, como una barra de nivel que sea visible en todo momento. Cuando se sube de nivel, estos puntos de experiencia vuelven a 0 y es necesario obtener 1.000 puntos otra vez para subir al siguiente nivel.


Figura 2: Ejemplo de *progress bar*

Los puntos de experiencia que ganas cuando devoras un enemigo, dependen directamente de la diferencia de tamaños entre tu personaje y el enemigo. De esta manera, los puntos de experiencia que ganas serán:

$$puntos\_experiencia\_ganados = 100 \times max(tamaño\_enemigo - tamaño\_propio + 3, 1) \quad (1)$$

Al llegar al 50 % del progreso de nivel, es decir cuando tengas 500 puntos de experiencia, aumentarás en 1 tu tamaño. También aumentas tu tamaño en 1 al subir de nivel cuando alcances 1.000 puntos de experiencia. De esta forma, por cada nivel, se aumentan 2 unidades de tamaño.

Al comenzar el juego empiezas en nivel 1 y tamaño 2, con 0 % de progreso de nivel. Cada nivel nuevo comienza con 0 puntos de experiencia.

### 4.2.2. Movimiento

El personaje principal es controlado por el jugador a través de distintas teclas. Dos teclas están encargadas del movimiento hacia atrás y hacia adelante, mientras que las otras dos cambian el ángulo de orientación del personaje. Los cambios de ángulos son en sentido levógiro o dextrógiro. Para entender estos conceptos, puedes visitar este enlace.

Los botones del teclado para poder jugar son los siguientes:

- `A`: girar en sentido levógiro.
- `D`: girar en sentido dextrógiro.
- `W`: moverse hacia adelante.
- `S`: moverse hacia atrás.

**Importante:** el personaje debe avanzar en la dirección hacia la cual está orientado.

## 4.3. Enemigos [ 20 % ]

Los enemigos solo buscan seguir las ordenes de Hernán y destruirte para que el mundo no renazca. Estos se comportan automáticamente con una inteligencia básica que debes programar[1]. Debes tener en cuenta

[1]Hernán: Así es, debes programar a quien te destruirá jojojo

que pueden atacar, escapar y moverse aleatoriamente. También tienen un tamaño y una serie de nuevos atributos que le permitirán tomar las decisiones adecuadas sobre atacar o escapar.

Para empezar, es necesario definir una unidad de distancia como la distancia euclidiana entre una coordenada de la interfaz y otra. Por ejemplo, el punto $(0, 0)$ de la interfaz (i.e. la esquina superior izquierda) tiene 5 unidades de distancia con el punto $(4, 3)$.

#### 4.3.1. Rango de visión

Este rango es utilizado por el enemigo mientras no esté presente el personaje principal dentro de él. Se define de forma circular cuyo centro es el centro del enemigo y su radio es: $tamañno\_enemigo \times 30$

#### 4.3.2. Rango de escape

Este rango es utilizado cuando el personaje principal entró a su rango de visión. El rango de escape se calcula como: `rango visión * 1.5`.

#### 4.3.3. Tiempo de reacción

Antes de darse cuenta de que el personaje princiapl está en su rango de visión, el enemigo tendrá un tiempo de reacción aleatorio con distribución uniforme, entre 0 y 1 segundo. Esto significa que si es lo suficientemente lento en reaccionar, podría llegar a no intentar escapar y morir antes de reaccionar.

#### 4.3.4. Inteligencia

Los enemigos deberán saber qué hacer en cada momento dependiendo de lo que está ocurriendo cerca de ellos.

Tendrán tres comportamientos básicos: moverse, escaparse y atacar.

- Cuando los enemigos no detectan al personaje principal dentro de su rango de visión, se mueven aleatoriamente en cualquier dirección, por intervalos de un segundo. Con una probabilidad 0,25 eligen una dirección para moverse y se moverán en dicha dirección por un segundo. Luego volverán a elegir otra dirección cuando transcurra ese segundo.
- Si el personaje principal se encuentra dentro del **rango de visión** de un enemigo, y tiene menor nivel que él, dicho enemigo procederá a acercarse e intentar atacarlo. Para que el enemigo deje de intentar atacar al personaje principal, dicho personaje principal deberá alejarse del **rango de escape** del enemigo.
- Si el personaje principal es de mayor nivel que el enemigo, dicho enemigo escapará por su vida una vez que detecte que el personaje principal está en su **rango de visión**. Dejará de escaparse y volverá a su movimiento normal una vez que el personaje principal ya no se encuentre en su **rango de escape**.
- Si el personaje principal es del mismo nivel del enemigo, este enemigo decidirá al azar, con una probabilidad de 0,5 si atacar o si escapar. Esta decisión sólo se toma la primera vez que el personaje principal entra al rango de visión y se mantiene hasta que el personaje principal sale del rango de escape, muere o el enemigo muere.

Un ejemplo de esto es que el personaje principal estaba a 15 unidades de distancia de un enemigo, quien tiene un rango de visión de 10 unidades y un rango de escape de 15 unidades. Este enemigo por mientras se movía aleatoriamente. Al acercarse 5 o más unidades de distancia, el enemigo activa su rango de escape y solo dejará de atacar/escaparse al estar a 15 o más unidades de distancia del personaje principal. Después de que esté a más de 15 unidades de distancia, tendrá que volver a estar a 10 o menos unidades de distancia para empezar a escapar nuevamente.

#### 4.3.5. Relación entre rango de visión y escape

La relación de estos dos atributos es que, mientras un enemigo no haya visto al personaje principal, está menos pendiente de sus alrededores, por lo que solamente estará observando en su rango de visión, que es menor al de escape.

Una vez que haya visto al personaje principal, se pondrá más pendiente, por lo que en vez de utilizar su rango de visión, utilizará el rango de escape. Esto significa que observará en un rango mayor al que observa comúnmente, hasta que vuelva a estar en condiciones normales.

### 4.4. Colisiones [ 10 % ]

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

- Personaje principal y bordes.
- Enemigos y bordes.
- Personaje principal y enemigos.
- Personaje principal y bombas.
- Enemigos y bombas.
- Personaje principal y vida extra.
- Personaje principal y puntaje extra.
- Personaje principal y zona segura.

Las colisiones entre el personaje principal y los bordes son para evitar que este salga de la ventana. Si el personaje colisiona con otro enemigo, disminuye su cantidad de vidas[2]. Cabe destacar que los personajes no se pueden atravesar entre sí —digamos que funcionarían como murallas. La distancia entre entidades para la colisión queda a criterio del desarrollador, mientras se pueda entender gráficamente.

### 4.5. Etapas

#### 4.5.1. Cambio de tamaño [ 2.5 % ]

Recuerda que en este juego, a medida que avances, tu personaje será cada vez más grande y habrá más enemigos, por lo que la pantalla de juego se podría ver colapsada si es que no tenemos cuidado. Es evidente notar que en los últimos niveles, como todos los personajes serán tan grandes, será difícil el movimiento, pues habrán pocos espacios disponibles. Para evitar esto, deberás iniciar el juego con los personajes muy pequeños, de manera que al ir creciendo de nivel, no se estorben todos los elementos del juego.

Cuando se inicia el juego, debes elegir un tamaño adecuado, como el que se muestra en el ejemplo que sigue:

[2]Sección 4.2.3

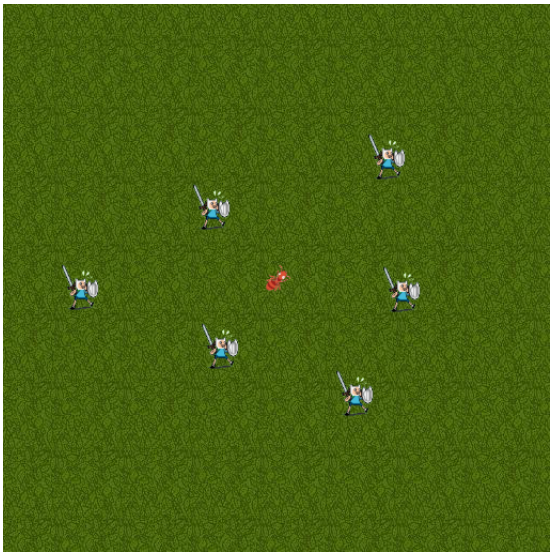

Figura 3: Nivel 1

Mientras que el nivel más grande, debe seguir teniendo espacios disponibles para el movimiento:

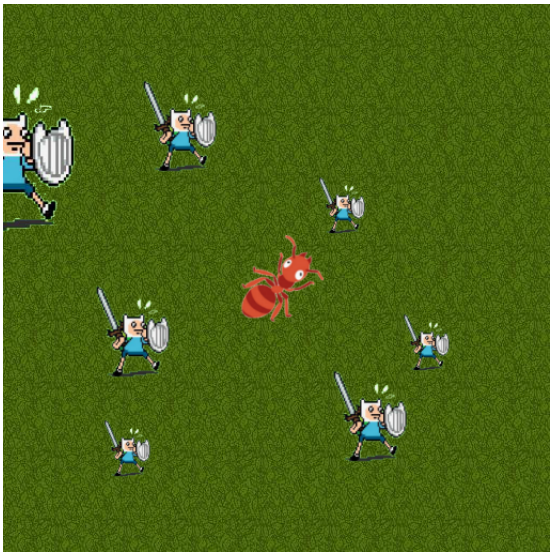

Figura 4: Nivel 5

#### 4.5.2. Elementos del juego [ 15 % ]

Dentro del juego existen distintos elementos repartidos en el campo de juego que se activan al momento de colisionar con la imagen del objeto y otros son se deben comprar. Cada uno tiene distintos efectos en tu personaje. Estos pueden ser:

- **Bomba:** cuando un personaje (ya sea el tuyo o el enemigo) pasa por encima de una bomba, este objeto explotará en 3 segundos, eliminando instantáneamente a los personajes que se encuentren en un rango de 10 unidades de distancia. Luego, la bomba desaparecerá del mapa. Si el personaje principal se encuentra en el radio al momento de la explosión, el juego termina automáticamente. Este objeto debe ser representado por una bomba en la interfaz, mientras que cuando está por explotar, se debe representar el tiempo restante mediante un contador en la posición del objeto.
- ***Safe zone:*** este objeto es representado por una imagen con la palabra *Safe zone* como la figura 5 y define un espacio en donde el personaje principal pueden esconderse. Cada vez que el personaje principal entra a esta zona (i.e. colisiona con la imagen de *Safe zone*), la imagen del personaje debe desaparecer hasta que este decida salir de la zona, es decir, dejar de colisionar. El personaje principal no puede ser atacado dentro de esta zona ni ser reconocido por los enemigos en su rango de visión o escape. Estas zonas son estáticas por el resto de la partida.


Figura 5: *Imagen Safe zone*

- **Puntos Extra:** este objeto te otorga 1.000 puntos y es representado por una moneda. Aparecen espontáneamente en el juego y al pasar por encima, esta desaparecerá y se sumarán 1.000 puntos a tu puntaje total.

- **Vida extra:** cada vez que pases por encima de este objeto, desaparecerá y se llenará tu vida por completo.

  Las apariciones de cada objeto son aleatorias en tiempo uniforme entre 1 y 30 segundos. Respecto a la localidad, queda a criterio del *game developer* (tú) el espacio donde aparecen.

#### 4.5.3. Aparición de enemigos por etapa [ 2.5 % ]

El juego tiene cinco etapas distintas. En cada etapa aparecen enemigos continuamente. El tamaño con el que aparecen y el tiempo entre sus apariciones depende de cada etapa.

El tiempo entre las apariciones de cada enemigo en una etapa, es un tiempo exponencial de parámetro $\lambda_{etapa}$, en el que $\lambda_{etapa}$ depende de la etapa en la que se encuentre. El tamaño con el que aparecen los enemigos se obtiene con la siguiente fórmula: $\lfloor triangular(a_{etapa}, b_{etapa}, c_{etapa}) \rfloor$[3]. En la que $a_{etapa}$, $b_{etapa}$ y $c_{etapa}$ dependen de la etapa en la que se esté.

A continuación, se encuentran los valores de $\lambda_{etapa}$, $a_{etapa}$, $b_{etapa}$ y $c_{etapa}$ para cada etapa del juego.

| Etapa | $\lambda_{etapa}$ | $a_{etapa}$ | $b_{etapa}$ | $c_{etapa}$ |
|---|---|---|---|---|
| 1 | 1/10 segundos | 1 | 5 | 1 |
| 2 | 1/8 segundos | 1 | 6 | 3 |
| 3 | 1/6 segundos | 3 | 7 | 5 |
| 4 | 1/4 segundos | 5 | 9 | 7 |
| 5 | 1/2 segundos | 7 | 10 | 9 |

Cuadro 1: Parámetros de aparición de enemigos por etapa

Los enemigos deben ir apareciendo hasta que completes la etapa. Al cambiar de etapa, debes cambiar los tamaños en la interfaz, y también debes actualizar los tiempos de generación de los enemigos (junto con el tamaño de aparición de estos mismos). Una vez que completes la etapa 5, se acaba el juego.

### 4.6. Puntaje [ 5 % ]

Una característica importante para que tu juego sea divertido es que el jugador siempre sepa si lo está haciendo bien o mal. Para lograrlo, se debe implementar un sistema de puntajes dentro del juego. Se entregará puntaje de tres maneras: un puntaje inicial, un puntaje por segundo de supervivencia, un puntaje por derrotar un enemigo y un puntaje por subir de nivel. Sin embargo, mientras más gente pruebe tu juego, más probable es que a alguien no le gusten los puntajes. Algunas personas querrán algo más fácil, y otras buscarán un desafío. Para darle en el gusto a todo el mundo, se entregará el archivo `constantes.py` con una serie de valores que definirán que tanto vale cada acción. El archivo contiene valores por defecto, pero tu programa debe estar preparado para ocupar cualquier valor que venga en el archivo.

Hay cuatro formas de recibir puntaje en el juego, con cuatro constantes asociadas que las manejan. En las definiciones siguientes, las constantes estarán escritas con `MAYUSCULAS_Y_UNDERSCORES`. Por otro lado, las cosas escritas con `snake_case` son variables del juego.

- Cuando el juego empiece, el jugador debe recibir inmediatamente `PUNTAJE_INICIO` puntos.
- Por cada segundo que el jugador se mantenga con vida, debe ganar `PUNTAJE_TIEMPO` puntos. Recuerda que esto se debe actualizar en tiempo real.

[3]La función triangular está incluida en la librería `random` de Python

- Cada vez que un enemigo sea derrotado debes entregarle `1000 + PUNTAJE_ENEMIGO * (nivel_enemigo - nivel_jugador)` puntos al jugador.
- Por último, cada vez que el jugador suba de nivel debes darle un bonus de `1500 + PUNTAJE_NIVEL * nuevo_nivel` puntos.

No olvides preguntarle el nombre al jugador una vez terminada la partida, ya que deberás guardar los diez mejores puntajes y mostrarlos en el menú principal tal como sale explicado en la sección correspondiente[4]. Esta información debe seguir ahí la próxima vez que juegues. Además, es de libre elección la forma de guardarlo.

## 4.7. Tienda [ 10 % ]

En la *Tienda* puedes comprar objetos para ir mejorando los atributos de tu personaje utilizando tu puntaje. Se debe poder acceder a la tienda mediante un botón, el que se encontrará permanentemente en la ventana principal del juego. Cuando ingreses a ella, el juego se debe pausar hasta finalizar la compra. La tienda debe desplegarse en una nueva ventana, mostrando los objetos disponibles y tu inventario.

### 4.7.1. Objetos de la tienda

En la *Tienda* podrás adquirir tres tipos de objetos, los cuales mejorarán distintos atributos de tu personaje. Cada objeto tiene un precio y una bonificación de atributo. Al comprar el objeto, se descontará su precio de tu puntaje y se sumará su bonificación al atributo correspondiente. Una vez comprado, el efecto durará por el resto de la partida.

Cada uno de los objetos presentes en la tienda te ayudará en uno de estos aspectos:

- **Velocidad de movimiento**: este objeto aumenta la rapidez de tu personaje. Su precio es de 250 puntos y la bonificación que entrega es de 10 % sobre el valor actual.
- **Velocidad de ataque**: este objeto aumenta la velocidad con la cual tu personaje infringe daño a otras entidades. Su precio es de 500 puntos y la bonificación que entrega es de 15 % sobre el valor actual.
- **Vida**: este objeto aumenta la vida total de tu personaje. Su precio es de 750 puntos y la bonificación que entrega es de 20 % sobre el valor actual.

### 4.7.2. Inventario

Tendrás un *Inventario* con espacio para almacenar cinco objetos, y se pueden repetir objetos dentro de estos, pudiendo acumular los poderes. Cuando quieras hacer una compra en la *Tienda*, debes acceder a ésta y **arrastrar** el objeto que deseas comprar hasta uno de los recuadros presente en tu *Inventario*. Si todos los espacios están ocupados y deseas reemplazar un objeto por otro, debes arrastrar el nuevo objeto sobre el que quieras reemplazar. Esto hará que pierdas el objeto antiguo junto a la bonificación que te entregaba, y recibirás el nuevo objeto con la bonificación de éste. El inventario se tiene que poder mostrar en la interfaz en todo momento durante la partida.

[4] Véase sección 4.8


Figura 6: Tienda e inventario

## 4.8. Interfaz del juego [ 15 % ]

Al correr el programa, se deberá mostrar un menú que permita comenzar el juego o mostrar el *ranking* de puntajes más altos alcanzado en partidas anteriores.

La interfaz del juego debe mostrar, al menos, los siguientes elementos:

- Botón de salida: al presionar este botón se reingresa al menú principal con un mensaje del puntaje obtenido hasta ese punto.
- Botón de pausa: al presionar este botón el juego se pausa. También se puede pausar presionando **CTRL + S**.
- Botón de tienda: al ser presionado el juego se pausa y entras al espacio de tienda. También se puede acceder mediante la combinación de teclas **CTRL + T**.
- Barra de vida: barra que representa el porcentaje actual y total de la vida de tu personaje.
- Puntaje: puntaje actual del jugador.
- Progreso del nivel: barra que representa la cantidad de experiencia obtenida en el nivel actual y cuanto falta para llegar a 1.000.

## 4.9. Representacion gráfica

Los personajes deben de tener movimientos animados, tanto para avanzar como para atacar. Para lograr esto, se debe de diferenciar su aspecto al moverse. Considerando al menos tres estados de movimientos para cada acción y dirección, como los mostrados a continuación:

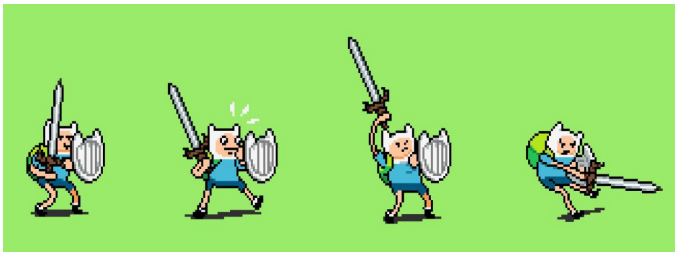

Figura 7: Diferentes estados de movimiento en una dirección

De esta forma, el movimiento se verá aún más real. Para esta sección, se le recomienda buscar en internet *sprites*. Junto al enunciado, se les entregara un conjunto básico de *sprites*. A continuación, dejamos un ejemplo de un *spritesheet* de un personaje cualquiera:

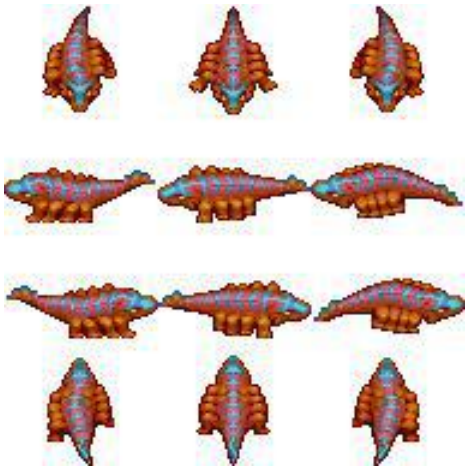

Figura 8: Sprite sheet de un insecto

# 5. Bonus

Para poder optar a bonus, es necesario mínimo tener un 4 como nota de tarea 5, sin contar bonus.

## 5.1. Estiloso (0.5 pts)

Como te habrás dado cuenta, las imágenes entregadas son demasiado básicas y no se detalló ningún requisito en cómo deben verse los elementos de la interfaz gráfica; es decir, qué color usar, dónde poner los botones, etcétera. Es aquí donde entra tu habilidad de diseñador y hacer un cambio total en tu interfaz gráfica para agregarle estilo a esta. Para obtener este bonus, debes cumplir con los siguientes requisitos:

1. Cambio de *sprite*: debes buscar o crear tus propios *sprites* que incluyan la animación de movimiento y que sean acorde a la temática del juego.

2. Fondo: ningún elemento de tu interfaz puede poseer de fondo el predeterminado por PyQt5, es decir, no se pueden ver como la interfaz dada en la T03. Deben agregarle fondo a los *widgets, label* y botones. Recuerden tener presente el uso de espacio y color en la interfaz. Este PDF los podría guiar.

### 5.2. Desarrollador (0.5 pts)

No estás satisfecho con esta pincelada a PyQt5 por lo que te propones agregar nuevas funcionalidades a tu interfaz con material que deberás investigar.

1. Efectos sonoros: cada vez que colisionas con alguien o te lo comes, debe haber un sonido único que notifique al usuario de tal acción. También deben haber al momento de hacer clic en un botón.

2. Búsqueda y selección de música: mediante una ventana de buscador[5], debes permitirle al usuario seleccionar alguna carpeta de interés que contenga su música personal y el juego le permita reproducir tales canciones como música de fondo. En caso de que la carpeta contenga archivos indeseados o la música esté en un formato que tu tarea no pueda procesar, el juego debe alertar al usuario mediante un *pop-up*.

3. *Fullscreen*: debes permitir que el juego se pueda cambiar a pantalla completa lo cual implica re-escalar todos los elementos de la interfaz para no perder la proporción de tamaño y tampoco dejar espacios en blanco. Puedes ver la interfaz de la tarea 3 en donde su tamaño podía variar, pero no se dejaban espacios vacíos, sino que se ajustaban otros componentes al tamaño final.

### 5.3. Cooperativo (1.0 pts)

Qué mejor manera de pasar una tarde que juntarte con tu mejor amigue[6] a disfrutar de una larga partida de DCCells. El problema es que el juego actualmente está en fase *beta* y no dispone de tal modalidad. Debido a lo anterior, se propone como bonus implementar el sistema cooperativo. Los requisitos son:

1. Un jugador debe moverse con WASD, W y S son para avanzar o retroceder mientras que A y D son para rotar a la izquierda o derecha respectivamente, mientras que el segundo jugador debe utilizar las flechas($\leftarrow, \rightarrow, \uparrow, \downarrow$).

2. La barra de progreso y *score* deben ser únicos para cada jugador.

3. Se pasa de nivel solo cuando ambos jugadores lleguen al 100 % de sus barras de progreso. Si uno llega al 100 % antes que otro, ese jugador puede seguir comiendo y ganando puntaje pero no contará como experiencia para crecer más hasta que el otro jugador llegue al 100 % y se pase de nivel.

4. Los jugadores crecen de tamaño de forma independiente cuando su barra de progreso llega al 50 %.

5. Cualquier otro detalle queda a tu criterio.

## 6. Entregable

Un diagrama de secuencia muestra la interacción de un conjunto de objetos de una aplicación a través del tiempo, en el cual se indicarán los módulos o clases que formarán parte del programa y las llamadas que cada uno de ellos harán para realizar una tarea determinada.

Para esta tarea deberán realizar dos diagramas de secuencia. Uno para cuando se colisiona con un enemigo y otro para el cambio de nivel.

A continuación dejamos un ejemplo de un diagrama de secuencia:

[5] Investigar QDialog
[6] No podemos asumir el género de tu amigo o amiga.

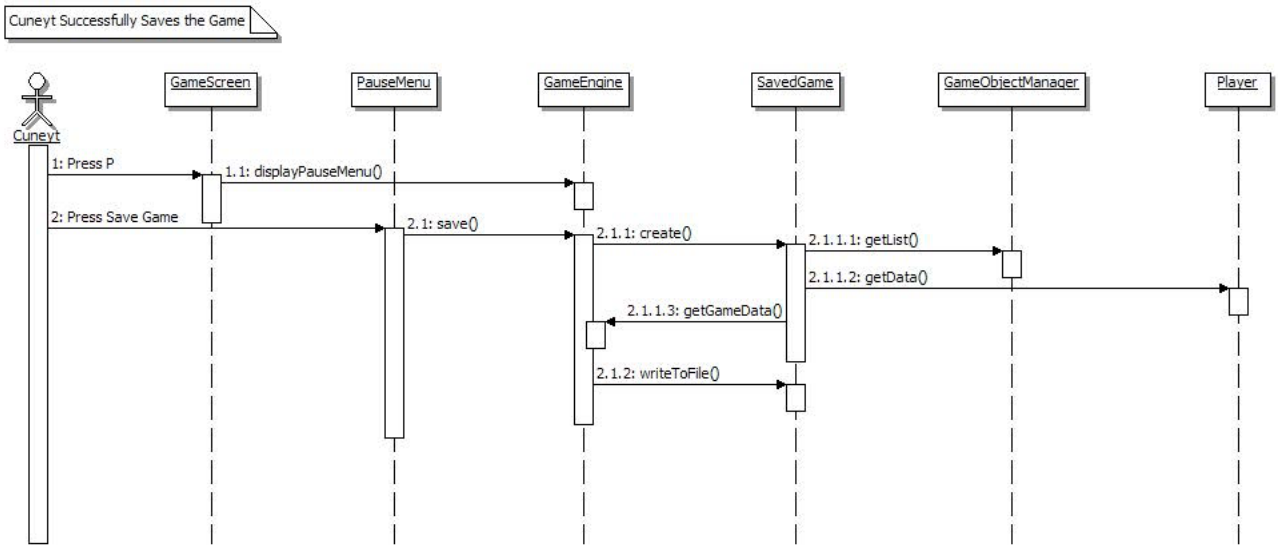


Figura 9: Diagrama de secuencia de guardar una partida de un juego

# 7. Restricciones y alcances

- Tu programa debe ser desarrollado en Python v3.6.
- Esta tarea es estrictamente individual, y está regida por el Código de Honor de la Escuela: Clickear para Leer.
- Tu código debe seguir la guía de estilos descrita en el `PEP8`.
- Si no se encuentra especificado en el enunciado, asume que el uso de cualquier librería Python está prohibida. Pregunta en el foro si es que es posible utilizar alguna librería en particular.
- Si no subes el entregable, tendrás **5 décimas menos** en la nota final de la tarea.
- El ayudante puede castigar el puntaje[7] de tu tarea, si le parece adecuado. Se recomienda ordenar el código y ser lo más claro y eficiente posible en la creación algoritmos.
- Debes adjuntar un archivo `README.md` donde comentes sus alcances y el funcionamiento del sistema (*i.e.* manual de usuario) de forma *concisa* y *clara*. **Tendrás hasta 24 horas después de la fecha de entrega** de la tarea para subir el `README.md` a tu repositorio.
- Crea un módulo para cada conjunto de clases. Divídelas por las relaciones y los tipos que poseen en común. **Se descontará hasta un punto si se entrega la tarea en un solo módulo[8].**
- Cualquier aspecto no especificado queda a tu criterio, siempre que no pase por sobre otro.

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

[7] Hasta −5 décimas.
[8] No agarres tu código de un solo módulo para dividirlo en dos; separa su código de forma lógica