Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
Segundo Semestre de 2013

IIC 1103 ⋆ Introducción a la Programación

Proyecto 3: Pythonia's Master

Objetivo General

- Diseñar, crear y utilizar clases y sus objetos.
- Aplicar el uso de listas y listas multidimensionales.
- Implementar funciones de ordenación y búsqueda.

Introducción

Las malvadas serpientes de Pythonia están enfurecidas por haberte atrevido a entrar en sus terrenos. Saben que ahora puedes programar casi cualquier cosa en su lenguaje, que creían que sólo ellas dominaban. Es por esto que han decidido poner atajo a tus habilidades y que no puedas humillarlas convirtiéndote, además, en el/la *Pythonia's Master*.

Figura 1: Malvada serpiente enojada de Pythonia

En una de tus incursiones por el lugar te has encontrado con unos enmarañados laberintos, y las serpientes han encontrado este lugar muy propicio para atacarte. Este es su territorio, así que te encuentras en un serio peligro que tendrás que sortear. ¡Nadie dijo que sería fácil!

El miedo te invade

El lugar es tenebroso y te encuentras completamente solo(a), sientes como se acercan las serpientes y una gota de sudor cae por tu frente. Te das cuenta que estás en un laberinto y la presión de salir de él comienza a ser cada vez mayor.

A medida que caminas logras darte cuenta de lo que debes hacer para salir del laberinto: tendrás que pasar por algunos lugares especiales y "dominarlos" o "activarlos a tu favor". Una vez que hayas "dominado" todos estos lugares, te liberarás de este encierro y podrás seguir tu vida pero ahora con el gran título de *Pythonia's Master*.

Las serpientes, tus enemigas, también tratarán de dominar los casilleros especiales. Si logran dominarlos todos, no sólo perderás tu título, sino que te quedarás en los laberintos por toda la eternidad (o por el resto de tu vida, lo que termine primero...).

También existen algunos elementos dentro del laberinto que contienen poderes especiales, con los que te podrás ayudar para derrotar a las serpientes. Pero ¡cuidado! Las serpientes también pueden tomarlos y obtener ventajas que podrían llegar a significar tu fin.

Elementos a considerar

Existen muchos elementos que tienes que considerar en la programación de tu juego. Te los detallamos a continuación:

Laberinto

Los laberintos del juego tendrás que leerlos desde archivos de texto, que contendrán una estructura similar a la que se muestra a continuación:

```
8,8
xxxxxxxx
xeooxxsx
xoxodoox
xdooxoxx
xoooooox
xdoxxoox
xooxoodx
xxxxxxxx
```

La primera línea de texto contiene dos números, separados por una coma, que representan el ancho y alto de casillas de laberinto, respectivamiente. Luego tendrás tantas líneas como el alto indicado, que determinan el contenido de cada casilla. La letra `x` representa un muro de laberinto y la letra `o` representa un casillero disponible para moverse (tanto el jugador como las serpientes no pueden traspasar muros).

La letra `e` corresponde a tu punto de entrada, lugar inicial en donde aparecerá el héroe (avatar, personaje) del jugador, y la letra `s` corresponde a la ubicación inicial de una serpiente (puede haber una o más serpientes inicialmente en el laberinto).

Los casilleros marcados con la letra `d` corresponden a los *puntos de dominación* (PD), los cuales tendrán que ser dominados por ti antes de que sean dominados por las serpientes.

En los archivos que debes descargar para realizar esta tarea podrás encontrar tres archivos de mapa en la carpeta `mapas`. No modifiques estos archivos, pero si tú quieres crear nuevos archivos de mapas, tanto para probar algo específico como por simple expresión de creatividad, puedes hacerlo. Debes considerar, sin embargo, que este juego está pensado para mapas que sean bien restringidos en cuanto a movimientos. La idea es que no existan espacios sin muros que permitan la existencia de cuatro casillas vacías juntas formando un cuadrado. En otras palabras, que el laberinto sólo tenga caminos del ancho de una casilla. Si te fijas bien, los tres mapas incluidos cumplen con esta característica.

Punto de dominación (PD)

Tanto las serpientes como tú tendrán que pasar por estos puntos para poder ganar. Cuando el jugador o una serpiente pasa por un PD que nunca ha sido dominado, automáticamente quedará dominado por quien pasó por él. Sin embargo, si alguien pasa por un PD ya dominado, tendrá que esperar 3 segundos (en el casillero) para poder dominarlo.

El que domina primero todos los PD, ¡gana el juego!

Serpientes

¡Son tus enemigas! Tendrás que esquivarlas a como dé lugar, ya que si te llegan a tocar morirás automáticamente [1].

Las serpientes deben tener algo de inteligencia. No basta con que se muevan por el laberinto de manera aleatoria. Deben intentar llegar hasta el jugador para así ganar directamente, o intentar ir hacia un punto de dominación o hacia algún ítem, dependiendo de qué les parezca más cercano.

¿Y cómo puede una serpiente determinar hacia dónde caminar (o reptar, más bien) de manera de poder llegar hasta un cierto punto? Mmmm... es como tratar de encontrar la salida a un laberinto, ¿no? (sólo que la salida es la posición del jugador...). El requisito mínimo es que la serpiente tenga una meta e intente alcanzarla, aunque por las características del laberinto se termine quedando atrapada. Por ejemplo, una serpiente puede usar la lógica de decidir hacia dónde reptar

[1] No es porque te muerdan pues, de hecho, las serpientes pitón no son venenosas. Se conocen como *constrictoras* pues enrollan su cuerpo en su víctima y la asfixian quitándole hasta su último aliento...

teniendo en cuenta en qué dirección quedará "más cerca" de su objetivo (considerando la distancia absoluta entre dos puntos). Pero hay un bonus especialmente indicado por si quieres crear serpientes estratégicamente superiores.

Power-Ups o ítems

Son poderes que aparecerán en algún casillero que esté disponible para el movimiento (vacío). Esto quiere decir que no podrán aparecer en algún casillero que esté ocupado por el jugador, serpientes, puntos de dominación, murallas u otros power-ups. Al tomarlo, tanto tú como tus enemigos gozarán de una pequeña ayuda para poder ganar.

Hay 3 ítems diferentes, que tendrán un efecto favorable al jugador o a las serpientes dependiendo de quién lo tome:

- Monty Python's Flying Circus: este ítem eliminará a una serpiente (de manera aleatoria) si es adquirido por el jugador o, si una serpiente es la que pone sus colmillos sobre éste, duplicará a dicha serpiente (creando una nueva serpiente en una casilla libre adyacente).
- Thrust SSC [2]: si un jugador gana este ítem podrá aumentar considerablemente la velocidad con la que puede desplazarse. En cambio, si cualquier serpiente se adueña de éste, hará que todas (sí, todas) las serpientes aumenten su velocidad (así que más vale que te desplaces inteligentemente por el laberinto...).
- Santiago 18:00-21:00 hrs: este ítem da la posibilidad al jugador de volver más lentas a todas las serpientes del laberinto, pero también le da la posibilidad a las serpientes de facilitarse alcanzar al jugador volviéndolo a él más lento.

Estos ítems deberán aparecer en el escenario aleatoriamente durante el juego, esto respecto a su posición, a qué ítem es y a los momentos en que aparecen. La decisión respecto a estas probabilidades es tuya, pero ten presente que no es la idea que llenes el laberinto de ítems.

Medidas de los elementos

El laberinto está representado por varios cuadros, formando una grilla de casilleros con un cierto ancho y alto. Pero cada uno de estos casilleros tiene también una medida... ¿Te acuerdas que una pantalla de computador está compuesta por muchos pequeños puntitos llamados *pixeles*? En este juego, todos los elementos tienen un cierto tamaño en pixeles.

Cada cuadro de la grilla tiene 32 pixeles de lado y, para todos los efectos prácticos, todos los elementos que podrán estar dentro del laberinto tendrán exactamente esta misma medida de 32x32 pixeles. Aún cuando la imagen que se vea en pantalla parezca no llenar por completo o sobrepasar los espacios de un cuadro del laberinto, igual debes considerar que ocupa sólo una región de 32x32 pixeles.

[2] `http://es.wikipedia.org/wiki/Thrust_SSC`

Sistema de coordenadas

El laberinto no es otra cosa que una grilla o matriz de cuadros, en que en cada cuadro puede haber ya sea un muro, punto de dominación o un ítem. El tamaño de la grilla estará dado, entonces, por la cantidad de cuadros de *ancho* y *alto* que tenga el laberinto. Estos mismos cuadros darán lugar a uno de los sistemas de coordenadas del laberinto, en donde el cuadro superior izquierdo será el de las coordenadas $(0, 0)$ y el cuadro inferior derecho será el de coordenadas $(ancho-1, alto-1)$.

Este sistema de coordenadas (basado en casilleros del laberinto) nos servirá para todos los elementos del juego que se posicionan exactamente en un casillero en particular del laberinto. Estos elementos son los muros, los puntos de dominación y los ítems. Para posicionar estos elementos sólo te bastará con especificar las coordenadas de una casilla y eso definirá completamente su ubicación (ocupando exactamente la casilla indicada).

Sin embargo, en el caso de las serpientes y del héroe del jugador, queremos que tengan un movimiento más fluído, y el moverse de cuadro en cuadro de la grilla del laberinto no sería un movimiento que personifique la fluidez precisamante... por ello, para el caso del jugador y las serpientes, usaremos un sistema de coordenadas de mayor precisión: basado en pixeles directamente. Pero no te preocupes, que ambos sistemas de coordenadas están muy relacionados. Cada pixel, de hecho, siempre pertenece a un único cuadro de la grilla y, además, cada cuadro de la grilla está compuesto por 32x32 pixeles.

Al igual que con los cuadros de la grilla, el pixel $(0, 0)$ será el de la esquina superior izquierda, que corresponderá también al extremo superior izquierdo del cuadro $(0, 0)$ de la grilla. De la misma manera, dado que tenemos *ancho* cuadros de ancho y *alto* cuadros de alto, el pixel del extremo superior izquierdo del cuadro $(ancho - 1, alto - 1)$ de la grilla (a su vez, el extremo inferior derecho del laberinto) será el pixel de coordenadas $((ancho - 1) * 32, (alto - 1) * 32)$. El pixel del extremo inferior derecho del mismo cuadro (correspondiente también al extremo inferior derecho del sistema de coordenadas en pixeles), será el pixel $(ancho * 32 - 1, alto * 32 - 1)$

De esta manera, decir que el héroe del jugador se encuentra en la posición $(0, 0)$, significará que la esquina superior izquierda del jugador (o área que ocupa un jugador) estará ubicada en el pixel $(0, 0)$, quedando ubicado, por consecuencia, completamente contenido en el cuadro $(0, 0)$ de la grilla del laberinto. Si decimos, en cambio, que el héroe del jugador está ubicado en el pixel $((ancho-1)*32, (alto-1)*32)$, significará que su esquina superior izquierda estará ubicada en el extremo superior izquierdo de cuadro $(ancho - 1, alto - 1)$ de la grilla (también completamente contenido en ese cuadro). Pero hablar de pixeles nos permite una fineza mucho mayor, sin necesidad de encasillar al jugador sólo en un cuadro específico de la grilla; decir, por ejemplo, que el jugador se encuentra en la posición $(20, 25)$ (su esquina superior izquierda se encuentra en ese pixel) implicará que una parte del jugador estará en el cuadro $(0, 0)$ de la grilla (pues el pixel $(20, 25)$ pertenece a ese cuadro) pero también tendrá otra porción en los cuadros $(1, 0)$, $(0, 1)$ y $(1, 1)$.

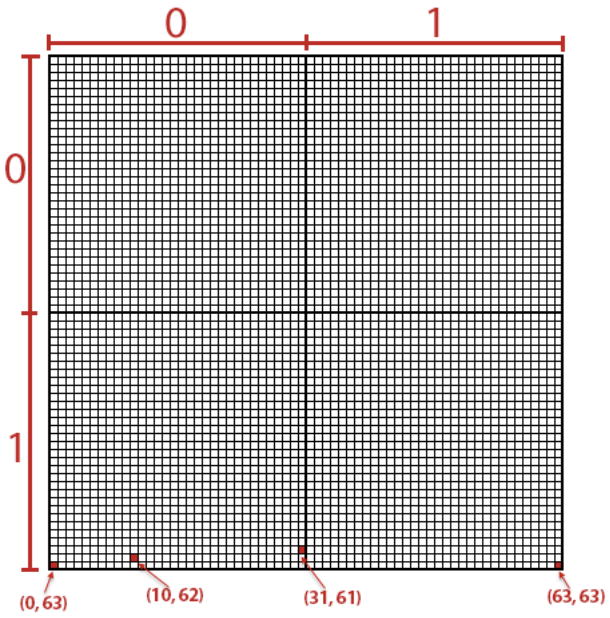

Figura 2: Sistemas de coordenadas del laberinto a nivel de casillas y de pixeles

Si te fijas, dado que el jugador se puede mover a nivel de pixeles, normalmente no estará completamente contenido en una sola casilla del laberinto. Sin embargo, para efecto de algunos cálculos, te convendrá considerar una simplificación de la posición del jugador o de las serpientes y llevarla a un equivalente en el sistema de coordenadas a nivel de grilla de casilleros. Para hacer esta transformación, puedes considerar que el casillero en el cual estará el personaje será aquél en el que tenga la mayor cantidad de la superficie del área del personaje. Así, en el ejemplo anterior, si el jugador está en la posición $(20, 25)$ en pixeles, diremos que equivalentemente se encuentra en la posición $(1, 1)$ de la grilla.

Área ocupada por los personajes

Los personajes del juego son tanto el que maneja el jugador como también todas las serpientes enemigas que puedan estar en el laberinto. Aunque como te lo explicamos anteriormente, desde un punto de vista lógico todos los elementos (personajes incluidos) ocuparán 32x32 pixeles de área, gráficamente verás que eso no se refleja exactamente así. De hecho, si fuésemos precisos, todos los personajes tendrían que ser un gran cuadrado de 32 pixeles de lado, y eso sería muy fome, ¿no?.

Las serpientes, visualmente, ocupan un buen poco menos que 32x32 pixeles. Y en el caso del personaje del jugador, verás que ocupa un poco menos en ancho, pero un poco más en cuanto al alto. No te preocupes por esto. Es sólo el resultado visual final, pero tú internamente tienes que siempre considerar todos los elementos de tamaño 32x32 pixeles. Si ubicas al jugador en las coordenadas $(32, 32)$ (extremo superior izquierdo del casillero $(1, 1)$ de la grilla), aunque visualmente

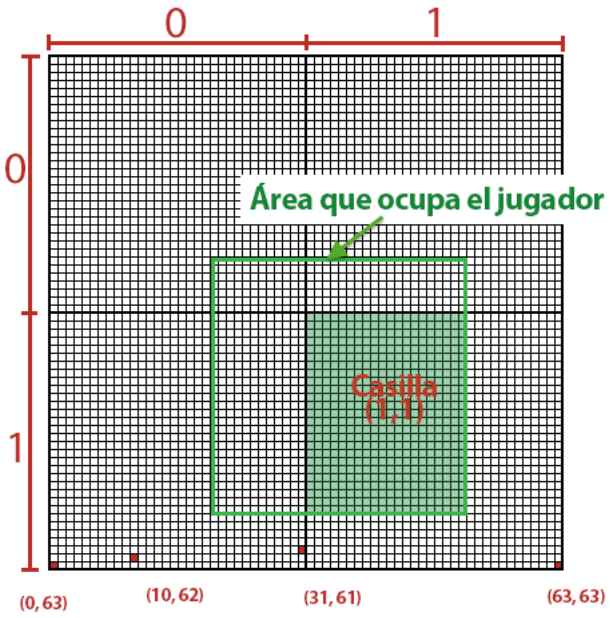

Figura 3: Traducción de jugador desde sistema de pixeles a casillas

parezca ocupar sólo unos 20 y algo pixeles de ancho al centro y unos 15 pixeles adicionales de la casilla $(1, 0)$ (la que está justo arriba) tu siempre tendrás que considerar, para todos los efectos, que ocupa sólo 32x32 pixeles utilizando únicamente, en este caso, la casilla $(1, 1)$. Prueba ubicando al jugador en diferentes coordenadas y verás a lo que nos referimos.

Recuerda tener presente que la posición de un jugador estará dada en pixeles, considerando que esas coordenadas indicarán la esquina superior izquierda del área que ocupa (área de borde verde de la figura 3). Sin embargo, para efectos de una posición válida para un personaje, no puedes considerar sólo esa esquina, sino toda el **área del personaje**. Imagina que en las coordenadas $(0, 0)$ y $(2, 0)$ de la grilla hay un muro, pero la coordenada $(1, 0)$ está vacía; a pesar de que las coordenadas $(40, 10)$ parezcan ser una posición válida para el jugador (pues la mayor parte de su área estará en la casilla $(1, 0)$ que está vacía) esta **no lo será**. Esto pues parte del área que ocupa estará igual en un cuadro que está con un muro y, como te comentamos anteriormente, ni el jugador ni las serpientes pueden traspasar muros.

Movimiento del jugador

Dado que moverás al jugador a nivel de pixeles, hay un "pequeño detalle" que debieras considerar para mejorar la "jugabilidad" de tu juego y no frustrar a tus jugadores. Como verás más adelante en la explicación de la librería, cada vez que muevas al jugador probablemente no lo desplazarás sólo 1 pixel hacia una determinada dirección, sino más de 1 pixel. Si te vas desplazando de esta manera por un camino horizontal y justo te encuentras con un camino que va hacia arriba pero cuyo ancho es de sólo 1 casilla (32 pixeles), aunque tu jugador

quiera moverse hacia arriba probablemente no podrá. Esto pues es muy improbable que haya dejado al personaje justo alineado de manera perfecta, como para que los 32 pixeles de ancho del jugador estén completamente contenidos en el cuadro de la grilla justo abajo del camino (condición necesaria para desplazarse hacia arriba sin que una parte del área del jugador quede en una casilla con muro).

Lo que puedes hacer para remediar esto es lo siguiente: cuando chequees si el jugador puede moverse hacia alguna dirección, sé un poquito más "permisivo". No exijas que los 32 pixeles estén completamente alineados, sino que permite que el jugador "esté corrido" en unos 4 o 5 pixeles y muévelo igual en la dirección indicada. Sin embargo, justo después de moverlo, tendrás que "ajustar" al jugador, como una especie de "encarrillamiento", para que esos 5 pixeles no sigan sobre el muro por el resto de su camino. Aunque no entiendas del todo esta última acotación en este momento, te recomendamos que lo experimentes en la práctica. Tú mismo te darás cuenta que el movimiento será un poco complicado a menos que sigas este consejo.

Implementación

Para la implementación contarás con una librería hecha por nosotros, que te permitirá ver de forma gráfica el laberinto y sus distintos elementos. Además, te permitirán jugar... con las flechas del teclado! (ni una gracia especificar los movimientos ingresando texto, ¿o sí?). A continuación una explicación de cómo utilizar la librería:

Debes descargar el archivo `proyecto3.zip` de la página del curso y descomprimirlo. En él encontrarás cuatro cosas:

- una carpeta `lib`: ésta es la carpeta que contiene todos los archivos de la librería. Debes ubicar esta carpeta en la misma carpeta en la que tengas los demás archivos de tu tarea y, por ningún motivo, debes modificar su contenido. Tu juego debe funcionar con esta carpeta sin absolutamente ninguna modificación.
- un archivo `juego.py`: este archivo es simplemente un ejemplo de un uso muy básico de la librería, con comentarios para explicarte algunos detalles de la misma, como por ejemplo cómo importarla y cómo usar los métodos de su clase principal, `Escenario`. No es necesario que uses este archivo, salvo para revisar lo que hace y cómo lo hace, y extraer esas ideas para tu propio programa. Aunque si te sientes más cómodo partiendo con ese mismo archivo, también puedes hacerlo.
- una carpeta `mapas`: dentro de esta carpeta se encuentran 3 archivos de nombre `mapaX.txt` (para X de 1 a 3). Éstos son los archivos que tienes que usar para cargar los laberintos.
- un archivo `escenario.html`: si abres este archivo se abrirá con tu browser favorito (que ojalá NO sea Internet Explorer) y te mostrará una documentación de todos los métodos de la clase `Escenario` que tiene la librería. Podrás

revisar los nombres de los métodos, los argumentos que reciben y una descripción de qué hacen y qué entregan como resultado. **Es imprescindible que revises a fondo este documento**, sólo así podrás familiarizarte con las posibilidades que te da la librería.

Hay una acotación importante que tienes que considerar (y que está explicado también dentro del archivo `juego.py`). Cuando uno usa una interfaz gráfica y captura las teclas que se presionan para hacer un juego como éste, es la interfaz gráfica (y, por lo tanto, en este caso la librería) la que, en algún momento, tiene que tomar el control de la ejecución del programa. Tú puedes crear todos los objetos que quieras y realizar todas las instrucciones que quieras antes de llamar al método `corre` de la clase `Escenario`, pero una vez que llames a ese método, durante todo el resto del programa el control del mismo pasará a manos de la librería (pues ella tendrá la tarea de estar, en **todo momento**, atenta al teclado y a refrescar la pantalla).

Sin embargo, la librería sabe que tú necesitas hacer cosas mientras transcurre el programa (como hacer que los elementos aparezcan, desaparezcan y se muevan). Por ello, lo que hará es que, cuando construyas un objeto de la clase `Escenario`, su constructor te pedirá que le entregues un objeto. Da lo mismo la clase a la que pertenezca ese objeto. El único requisito es que tenga un método que se llame `despierta` y que reciba (además de `self`, obviamente), un parámetro que será **un número entero**.

La librería lo que hará entonces será, muuuuuchas veces por segundo, llamar al método `despierta` del objeto que le hayas entregado, para darte la posibilidad de que hagas lo que necesites hacer. No tienes por qué tener tooooodo tu código en este método (por favor, no hagas eso!)... recuerda que desde un método de un objeto podemos llamar también a otros métodos, e incluso interactuar con otros objetos que, por ejemplo, estén guardados en atributos del mismo objeto. Así que de todas maneras puedes repartir responsabilidades en distintas clases de tu programa y escribir un buen código.

¿Recuerdas que el método `despierta` debe recibir un número entero? Lo que sucede es que la librería no puede saber exactamente cada cuánto tiempo podrá llamar al método `despierta`. Y esto puede ocurrir por diversas razones. Por ello, la librería le entregará a tu método `despierta` un número entero que indica la cantidad exacta de **milisegundos** (milésimas de segundo) que transcurrieron desde la última vez que llamó a ese método. ¿Que para qué podrías querer saber eso? Para dos razones, principalmente.

La primera es simplemente para poder ir llevando la cuenta del tiempo de juego que ha transcurrido. El mejor indicador de cuánto tiempo lleva corriendo tu juego es la suma de todos los tiempos que se le han entregado a tu método `despierta` (que estará en milisegundos, recuerda...).

La otra utilidad que tiene el saber este tiempo es para hacer que el movimiento de tu jugador y las serpientes sea más realista. Si tu dijeras, por ejemplo, que

cada vez que se llame al método despierta, si el jugador tiene la flecha "abajo" presionada, harás que avance 10 pixeles, ¿cuál sería la velocidad a la que se movería jugador, en pixeles por segundo? No tienes cómo saberlo, pues no sabes cada cuánto tiempo se llamará a este método y, como te dije, ni siquiera será siempre cada una cierta cantidad constante. En cambio, al saber el tiempo que ha pasado desde la última vez que se llamó a este método, en lugar de moverte de manera fija 10 pixeles, podrías ponderarlo por esta cantidad de tiempo que ha pasado. De esta manera, si ha pasado más tiempo, el jugador se desplazará más pixeles, pero si ha pasado menos tiempo, entonces serán menos pixeles. El hacer el movimiento de esta manera hará que, independiente de lo irregular que sean los tiempos transcurridos entre cada llamado al método `despierta`, tu jugador parecerá moverse a una velocidad (en pixeles por segundo) completamente constante.

1. Resumen

Como sabemos que en un enunciado de este tamaño es difícil que se te quede todo en la cabeza, te haremos un pequeño resumen de lo que tiene que hacer tu programa:

- Inicializar el laberinto (cargarlo desde un archivo, crear el `Escenario` y todos los elementos iniciales).
- Comenzar el juego llamando al método `corre` de `Escenario`.
- El jugador, junto a las serpientes, deberán moverse en el laberinto. El jugador controlado por las flechas del teclado, las serpientes controladas directamente por la inteligencia de tu programa.
- Las serpientes se moverán tratando de llegar al jugador por todo el laberinto (o algún otro objetivo).
- Si el jugador domina todos los PDs, gana!
- Si las serpientes dominan los PDs, el jugador pierde!
- Si las serpientes tocan al jugador, pierde!
- Si el jugador toma un poder, o las serpientes lo hacen, se aplicarán los efectos del mismo.

Muchos detalles del juego no están especificados en este enunciado (un ejemplo de la sección justo anterior: la velocidad del jugador y las serpientes). ¡Esto es a propósito! Además de que ya era un enunciado bastante extenso (:P), quicimos dejarlo un tanto abierto para que tú puedas decidir aspectos del juego y lo hagas único, a tú criterio (o descriterio ;)). De todas maneras tendrás también la rúbrica para guiarte respecto a la evaluación, pero siéntente libre de darle un toque personal a **TU** juego. Hay métodos en la librería que también te pueden ayudar a esto...

Bonus

Porque sabemos que eres todo(a) un(a) *Pythonia's Master*, tienes estos pequeños retos adicionales para demostarle a las serpientes que no tienen ninguna oportunidad frente a ti.

Bonus 1 (8 décimas) - Serpientes súper inteligentes

Advertencia: este bonus es de una dificultad considerable, pues además de cumplir con sus requisitos debes idear alguna manera de hacerlo suficientemente eficiente como para que no te arruine tu juego por su lentitud...

Las serpientes han descubierto una fórmula que las hace más inteligentes[3] y ahora no sólo seguirán al jugador por la pantalla de cualquier manera, sino que lo harán encontrando la **ruta óptima** (aquélla que tiene **la menor distancia en cuadros** para llegar hasta el jugador). Además, ahora tendrán otras variables a considerar a la hora de moverse.

La prioridad será:

1. Si están más cerca de un PD que no ha sido tomado por las serpientes, tomará ese camino.
2. Si está más cerca de un poder, intentará tomarlo.
3. Si está más cerca del jugador, continuará persiguiendo al jugador.

Con esto, el reto para el jugador se volverá bastante más interesante...

Bonus 2 (3 décimas) - Hall of Fame

Aquí la cosa no es sólo ganar, sino que ser el/la mejor de la historia. Después de cada partida deberás consultar el nombre del jugador y guardar en un archivo su nombre y el tiempo que se demoró en salir del laberinto. El ranking debe quedar ordenado desde aquel que se demoró menos tiempo. Obviamente, la idea de que lo guardes en un archivo es para que, cada vez que ejecutes el juego, el ranking persista y nunca pierdas ese valioso podio!

Restricciones

No hay restricciones específicas para este proyecto. Pero sí debes considerar que parte importante de lo que se evaluará en esta tarea es el uso de clases, objetos e interacción entre los objetos.

[3] No es a base de cereal y leche

Informe

Además de entregar el código de tu programa, deberás entregar un informe llamado [numero_de_alumno]_informe_proyecto_3.pdf en el cual se explique, usando no más de una plana, qué fue lo más difícil a la hora de desarrollar el proyecto y cómo lo resolviste. También podrás agregar a este informe una nueva plana con una sección que explique en qué aspectos superaste lo que se pide en este enunciado o cualquier supuesto o decisión que hayas considerado en tu solución (siempre y cuando no contradiga el enunciado).

Entrega y Consultas

Antes del día 19 de noviembre a las 23:59:00 debes entregar un archivo llamado [numero_de_alumno]_proyecto_3.zip en el sitio web del curso. Este archivo debe contener exclusivamente el archivo [numero_de_alumno]_proyecto_3.py, módulos que hayas programado adicionales a este archivo principal, y el informe en formato PDF.

Recuerda asegurarte de haber realizado una entrega correcta. Para esto, puedes descargar lo que subiste al buzón y verificar que el contenido esté correcto.

Las consultas respecto al proyecto deben ser realizadas en el foro del curso. Este foro será el **único** medio oficial para dar respuesta a las preguntas sobre el proyecto.