CC1001 Computación I
Tarea 4: Pac-Man

Prof. Romain Robbes Auxiliar: Boris Romero

Fecha de entrega: 11/11/2013

Objetivos

El objetivo de esta tarea es implementar una version del videojuego Pac-Man. En Pac-Man el objetivo es de comer todas las pastillas o "pac-puntos" en un laberinto. Para hacer la tarea mas complicada, hay fantasmas que persiguen a Pac-Man: si tocan Pac-Man, muere, excepto si Pac-Man ha comido una "super pastilla". En este caso, los fantasmas se devuelven azul y huyen Pac-Man; si Pac-Man toca un fantasma en este estado, se lo come. Sobreviven los ojos del fantasma: estos tienen que volver al centro del nivel para regenerarse. Al comer todas las pastillas, el nivel de juego se termina y Pac-Man puede pasar al nivel siguiente.

En caso de dudas sobre el juego, una version se puede probar acá:

`http://www.freepacman.org`

Si la versión original de Pac-man tomó a los alrededores de un año de desarollo, ustedes están ahora capaz de hacerlo en mucho menos tiempo, siguiendo las estapas siguientes. En la versión que desarollamos, no es necesario hacer las animaciones o graficos del juego original, como Pac-Man abriendo y cerrando la boca cuando avanza: es suficiente dibujar una forma sencilla. De misma manera los muros del laberinto y los graficos de los fantasmas pueden ser mas sencillos.

Como en la tarea precedente, usamos `world.py` como base. Usamos tambien datos compuestos, pero se permite usar mutación. Queda a su apreciación hacer un uso razonable de la mutación.

La tarea consiste en implementar las funciones necesarias para que funcione el juego. En esta tarea se describen los pasos a seguir y el módulo de funcionalidad base necesaria para implementar estas funciones. Ver tambíen el esqueleto de código adjunto que contiene informacíon adicional.

Etapas de desarrollo de Pac-Man

Recomendamos seguir las siguientes etapas de desarollo para llevar a cabo este proyecto. Por cada etapa se indica el tiempo de dedicación recomendado,

de tal manera que la carga de trabajo sea equilibrada y Ud. pueda entregar su tarea a tiempo.

Etapa 1: creacion y dibujo de un nivel basico Crear manualmente una estructura de datos representando un estado inicial de juego, según el plan dada abajo. Esta estructura tiene todos los elementos necesarios del juego: un Pac-Man (`P`), dos fantasmas (`F`), pastillas (`.`), muros (`#`), super-pastilla (`@`). Las coordendas empiezan a (0,0), y varian hasta (ancho, alto); en este caso, (9, 6). Distinto de la tarea precedente, el juego tiene "celdas" de tamaño fijos: la coordenadas son coordenadas de celdas, no de pixeles.

```
##########
#........#
#.######.#
#.#.FF.#.#
#.##.###.#
#.@..P...#
##########
```

Se recuerde consultar la estructura descrita en el esqueleto de código. Se destaca el uso de listas: una lista de fantasmas, y una lista de objetos en el nivel. Cada item tiene coordenadas describiendo su posición. **tiempo: 2 días**

Etapa 2: movimiento de Pac-Man Pac-Man se mueve de manera constante, siempre en la misma dirección. Esto se cumple hasta que al presionar una de las flechas (up, down, left, right), se cambia la dirección de movimiento. Implementar este comportamiento. A cada etapa de juego, Pac-Man se mueve de una celda en la dirección indicada (o no se mueve, si la dirección es "none").

Hay que tomar en cuenta los muros: Pac-Man no puede cruzar muros. Si mover Pac-man lo haría entrar en una celda donde hay un muro, no se mueve, y se para (su dirección se cambia en "none").

tiempo: 2 días

Etapa 3: pastillas y ganar Si Pac-Man entra en una celda donde hay una pastilla, esta desaparece, y se agregan puntos al juego. Al final de un turno, si no quedan pastillas, Pac-Man gana, y el nivel se acaba.

tiempo: 1 día

Etapa 4: fantasmas y perder Los fantasmas tambien se mueven. Hacer que se mueven de la siguiente manera:

Dado una posición de objetivo (la posición corriente de Pac-Man), una posición actual, y la lista de objetos (e.g., los muros), un algoritmo posible es: (1) determinar las direcciones que van a acercar el fantasma a su objetivo (por ejemplo, si estamos en (2, 2), y que Pac-Man es en (5, 5), podemos ir arriba o a derecha); (2) descartar las direcciones imposibles (porque hay muros cercanos,

por ejemplo si hay un muro en (2,3), no podemos ir arriba); (3) elejir entre las direcciones restantes una aleatoriamente. Si despues del paso (2), no quedan direciones validas, elejir aleatoriamente una de las restantes.

Si un fantasma entra en la misma celda que Pac-Man, este pierde una vida. Si no quedan vidas, pierde el juego.

tiempo: 2 días

Etapa 5: super-pastilla Si Pac-Man come una super-pastilla, se devuelve invincible por un tiempo. Los fantasmas se cambian a la colar azul (de miedo). En vez de tratar de acercarse de Pac-Man, tratan de huír a Pac-Man. Pac-Man puede comer a los fantasmas: estos se trasladan a su posicion inicial, al centro de la pantalla, antes de devolver normal.

Para implementar esto, puede ser necesario modificar las estructuras de datos. Para implementar el tiempo limitado de invincibilidad, se recomienda agregar un contador, al cual se resta uno a cada turno de juego.

tiempo: 2 días

Etapa 6, Bonus: leer niveles desde archivos Tener solo un nivel es un poco aburrido. Idealmente, podemos definir niveles arbitrarios. Definir una funcion que, dado una string que es un nombre de archivo, crea la estructura de datos representando el nivel asociado. Para eso se puede ocupar la función `file_contents(file_name)` definida en **world.py**:

```
# file_contents: string -> lista(char)
# devuelve los contenidos de un archivo como una lista de caracteres.
```

Seguir las mismas convenciones que en el primer ejemplo. Se nota el caracter especíal `"\n"` que indica una nueva linea en el archivo. Se puede probar con el archivo `nivel.pac` que contiene el mismo nivel que el descrito al principio de la tarea. Después de hacer eso, se puede diseñar niveles arbitrarios de Pac-Man, haciendo el juego mucho mas interesante.

Recordatorio: módulo world.py

Para implementar el juego, tiene que usar el módulo `world.py`. Este módulo se encarga de aspectos básicos de dibujo grafico y de interacción con el computador. El módulo usa el paradigma de *event-based programming* para controlar el juego. En este paradigma, el módulo llama periódicamente a funciones definidas por el programador en respuesta a eventos. Para esta tarea, los eventos son: el hecho que pasó tiempo, y el hecho que los jugadores apretan algunas teclas específicas del teclado.

La idea de la libreria es de tener un estado del "mundo", que contiene todos los elementos del juego en una estructura de datos. A cada paso de tiempo o acción del usuario, el progrma genera una nueva versión del mundo.

En particular, el módulo que tienen que implementar debe proveer las siguientes funciones:

- `first_world()`, que devuelve el estado inicial del mundo.
- `draw(mundo)`, que toma un mundo y lo dibuja en la pantalla.
- `step(mundo)`, que se llama cada cierto tiempo para similar el paso del tiempo. Esta función toma un mundo, y devuelve el mundo modificado considerando que ha transcurrido un instante de tiempo.
- `step_time()`, que devuelve la frecuencia del paso del tiempo, en milisegundos.
- `up(world), down(world), right(world), left(world), space(world)`, que son funciones llamadas cuando se apreta la tecla respectiva del teclado. Retornan un nuevo mundo que se modificó según la acción del usuario.
- `width()` que devuelve el ancho de la ventana gráfica.
- `height()` que devuelve el alto de la ventana gráfica.

Ademas, el módulo provee las siguente funciones de dibujos gráficos:

- `rectangle(p1, p2, color)` que dibuja un rectángulo dado dos puntos (esquina superior izquierda y esquina inferior derecha), y un color (un String).
- `circulo(p1, radius, color)` que dibuja un círculo dado el punto correspondiente al centro, su radio, y su color
- `line(start, stop, color)` que dibuja una línea dados sus dos puntos extremos y su color.
- `text(pos, contents, color)` que dibuja (escribe) en un punto un String.
- `file_contents(file_name)` devuelve la lista de caracteres contenidos en un archivo.