Universidad de SanAndrés

Trabajo Práctico N°3

Alumnos: Serena Feldberg y Benjamin Vitale

Correos electrónicos:

- sfeldberg@udesa.edu.ar
- bvitale@udesa.edu.ar

Profesor: Ernesto Mislej y Santiago Bassani

Fecha de entrega: 03/06/2022

Objetivos

El objetivo del trabajo práctico consiste en el desarrollo de un videojuego llamado Rogue o Roguelike games. El mismo funciona con dinámica de turnos, es decir, hay un cuerpo de iteración que se repite en cada turno, compuesto por el movimiento de los personajes y las acciones que se desprenden de este movimiento. El juego está compuesto por una serie de objetos que aparecerán dentro de un mapa (calabozo).

Personajes:

- Personaje principal, el jugador (‘@’)
- Gnomo (‘G’)

Objetos:

- Pico mostrado con el caracter (.
- Espada mostrada con el caracter /.
- Amuleto mostrado con el caracter “.

Elementos del mapa:

- Paredes indestructibles mostradas con los caracteres - o |
- Paredes de roca mostradas con el caracter .
- Escaleras ascendentes (<) y escaleras descendentes (>).

El personaje principal comienza en el nivel 1 de los calabozos y debe descender hasta 2 niveles (llegar al nivel 3) para:

- Encontrar un tesoro.
- Subir las escaleras del nivel 1 para salir del calabozo con el tesoro.

Subir las escaleras del nivel 1 sin el tesoro implica el abandono de la misión, y la finalización del juego.

Diseño

El código consiste en varios archivos distintos para una mejor organización.

Actions: se basa en el desarrollo de algunas funciones para el movimiento del jugador y del gnomo. Una de ellas es la función move que permite que el jugador se mueva dentro del mapa a partir del ingreso por teclado de las letras ‘a’, ‘s’, ‘d’ y ‘w’. Algo similar ocurre con la función move_gnome y gnome_move que permiten el movimiento del gnomo de manera automatizada. La primera de ellas mueve al gnomo al alzar, y la segunda sigue un algoritmo que le permite moverse de manera que persigue al jugador. Este algoritmo se realizó en base a los ángulos de una circunferencia, es decir, utilizando 4 cuadrantes, se mueve de determinada manera. Los cuatro cuadrantes son:

- 45° - 135°
- 135° - 225°
- 225° - 315°
- 315° - 360° / 360° - 45°

Se eligieron estos cuadrantes porque optimizan el movimiento hacia al jugador. Luego las funciones climb_stairs y descend_stairs permiten subir o bajar de nivel respectivamente dentro del mapa una vez que el jugador se posicione por encima del caracter correspondiente. Por último se encuentran las funciones possible_attack y attack, donde la primera evalúa si es posible realizar un ataque dependiendo de la distancia entre el gnomo y el jugador, y devuelve un booleano. La segunda, permite realizar el ataque quitándole puntaje de vida a quien haya sido atacado. Por último, se desarrolló la función pickup, que sirve para levantar objetos, y borrarlos del mapa una vez que ya hayan sido obtenidos.

Mapping: En este archivo se desarrollan 3 objetos, el primero es Tile que se usa para representar el tipo de cada caracter del calabozo. El segundo es Level que devuelve una instancia del calabozo. Dentro de ella se desarrollan algunos métodos que ayudarán con la representación de algunas cosas y con la obtención de algunos datos, por ejemplo la ubicación de cierto elemento, o la implementación de un render para visualizar el mapa.
Y por último se encuentra el objeto Dungeon que retorna una instancia del calabozo. Muchos de sus métodos son muy parecidos a los de Level. Esta clase tiene el método are_connected implementado, que sirve principalmente para evaluar las posiciones de los jugadores, la del

jugador principal con la del pico, y la del gnomo con la del jugador principal. Esto se realiza para que el juego se pueda jugar, por ende es necesario que ninguno aparezca encerrado entre rocas.

Ítems: en este archivo se desarrolla el objeto padre Item que inicializa las tres variables principales de los ítems que son el nombre, la cara y el tipo de ítem. Además tienen el protocolo de representación implementado, de modo de que se printeen de la manera deseada. Luego, se desarrollan los objetos Sword, Amulet y Pickaxe que solo se diferencian en el tipo, por lo tanto, el resto lo heredan de la clase Item.

Player: Este archivo contiene principalmente el objeto de Player, donde se inicializan los atributos básicos de ambos jugadores (el humano y el gnomo), como el nombre, la cara y el número de vida (hit points). Uno de sus métodos implementados es move_to que permite cambiar la ubicación del jugador. A partir de este objeto se desarrollan Human y Gnome.

- Human: por un lado, esta clase es específica al jugador principal (el humano). Dado que el humano puede obtener distintos objetos como la espada o el pico, tiene algunos atributos que permiten saber si los tiene o no. Con respecto a la cantidad de vidas que tiene el jugador, es decir, los hit points, el gnomo deberá atacar dos veces para que el jugador se muera. Por lo tanto, el humano tiene 50 hit points, pero el gnomo solo puede quitar 25 a la vez.
- Gnome: Esta clase corresponde a información relacionada con el gnomo, por lo tanto tiene atributos que facilitan conocer el estado de este, es decir, si está vivo o no.

Game: Por último, game es el archivo donde se inicializan las variables principales, tales como los personajes y el calabozo. Además, se van agregando distintos ítems al mapa. Las variables de los jugadores, es decir la del jugador principal y la del gnomo tienen la condición de que si el jugador no puede acceder al pico, antes de que se renderiza, se cambia la ubicación hasta que sea posicionado correctamente. Lo mismo sucede con el gnomo, solamente que se evalúa el camino entre la ubicación del gnomo y del jugador principal.
La lógica principal del juego consiste en los movimientos por turnos, es decir, primero se mueve el humano y después el gnomo. Todo esto sucede dentro de un bucle, que se ejecutará mientras el nivel del calabozo sea mayor a 0 y mientras que no se oprima la tecla ‘q’, pues esta sirve para salir del juego (‘quit’). Dentro del bucle se imprime el render del calabozo y dependiendo las teclas que el jugador principal elija presionar, se invocan a distintas

funciones. Por ejemplo, si el jugador utiliza la tecla ‘b’, podrá matar al gnomo, siempre y cuando tenga la espada y se encuentre en un rango de cercanía. Por otro lado, si presiona ‘a’, ‘s’, ‘d’ o ‘w’ indica que desea moverse para la izquierda, abajo, derecha o arriba respectivamente. Así pues, si presiona la tecla ‘p’ significa que desea levantar alguno de los objetos que haya en el mapa. Por último, el jugador ganará el juego si logra buscar el amuleto en el último nivel (nivel 3) y volver a subir al nivel 1, y dentro de este acercarse a las escaleras ascendentes.

Algunas alternativas de diseño del código fueron en un principio realizarlo sin clases ni objetos, es decir, solamente con funciones. Sin embargo, después de arrancar a desarrollar el código, nos dimos cuenta que con clases se facilitaba un poco más el trabajo, porque al poder usar herencia, muchos objetos quedan más simples y más organizados. Otra alternativa que terminamos cambiando es que el gnomo no mate al humano en un ataque. Lo terminamos haciendo que el gnomo mata al jugador con dos ataques. Es decir que el juego sea más fácil para el jugador y tenga más chances de sobrevivir. Con respecto al gnomo también solía ser que este perseguía al jugador de manera continua. Pero se hacía muy difícil porque una vez que este se acercaba al jugador no había manera de generar una separación. Entonces que el jugador llegase a la espada y lo matase era muy improbable. Decidimos hacer que el gnomo haga un movimiento aleatorio y otro persiguiendo al jugador.

Reseñas sobre problemas encontrados y soluciones

actions.climb_stair(dungeon[level-1], player, key)
TypeError: 'Dungeon' object is not subscriptable

- Este fue uno de los errores que nos surgió cuando intentamos acceder a otros niveles del calabozo. Para solucionarlo, utilizamos el método level de la clase Dungeon, que nos permite cambiar de nivel más fácilmente

TypeError: can't multiply sequence by non-int of type 'float'

- Este fue otro de los errores que nos encontramos cuando queremos calcular los ángulos de la ubicación del gnomo. Esto es porque se calcula el ángulo de la ubicación del gnomo respecto a la del jugador principal. Había algunos ángulos que no ingresaban en los condicionales, por lo tanto quedaban como None type. Tuvimos que incluir la opción de que los ángulos fueran negativos.

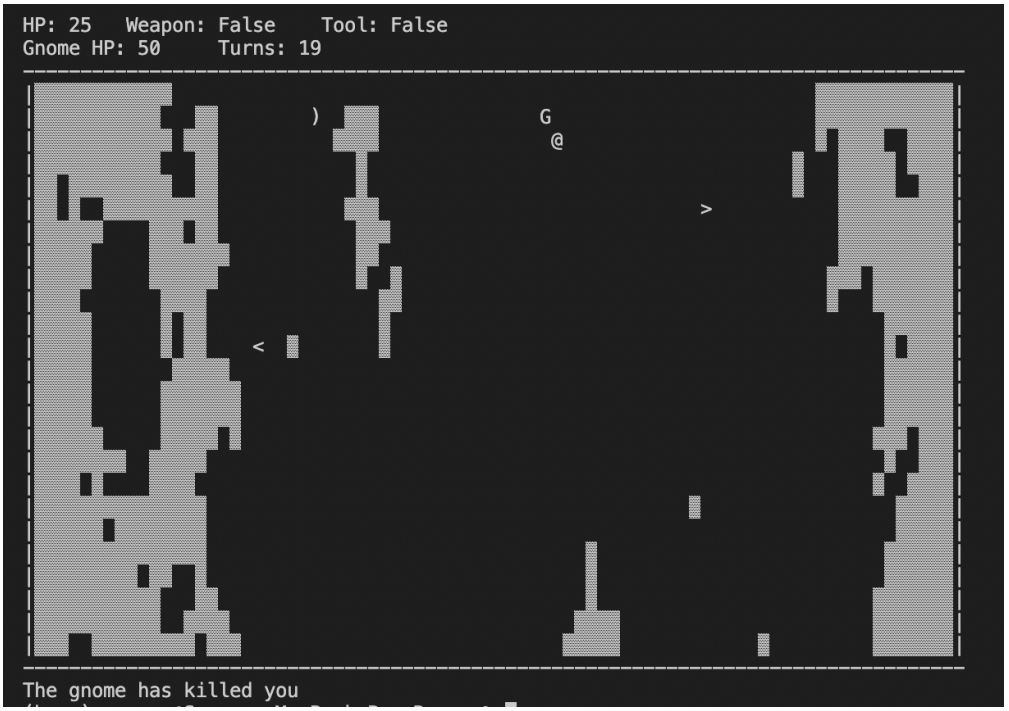
AttributeError: 'PickAxe' object has no attribute 'find_free_tile'

- Este problema lo tuvimos cuando quisimos insertar el pico en el mapa al azar, pero no era este el problema sino que estábamos utilizando la función incorrecta para insertar el pico, pues en vez de utilizar find_free_tile, la función add_item tiene la cualidad de que si no se le pasa ninguna ubicación, elige una al azar directamente, mientras que esa ubicación esté libre.

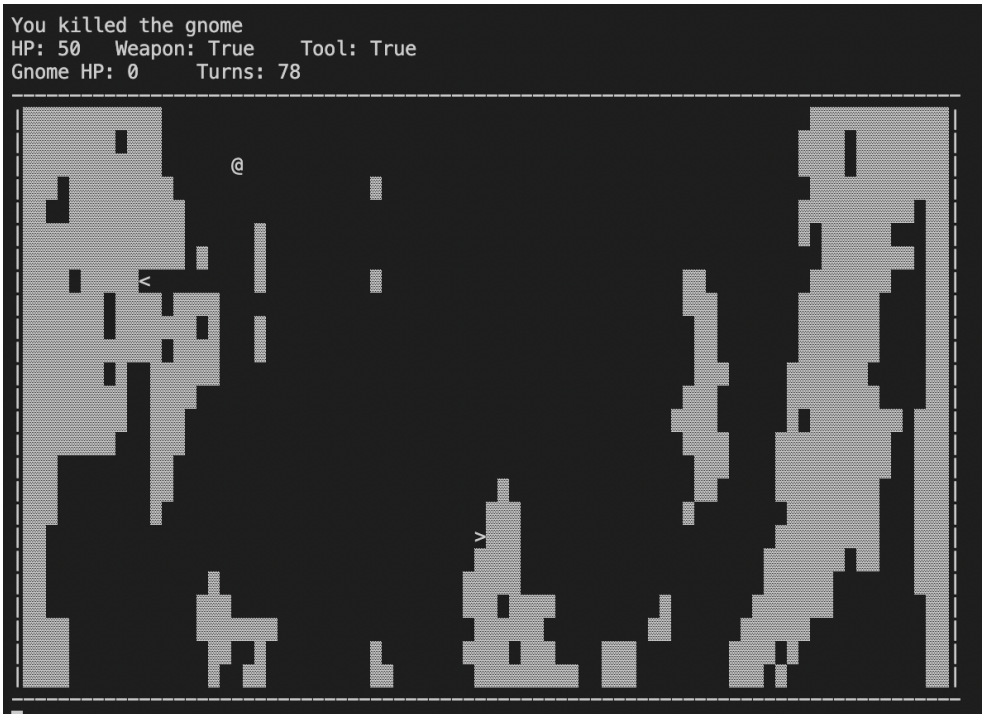
Indicaciones para ejecutar correctamente el programa

El programa se corre desde el archivo game. Dentro de este archivo, se llama a otros archivos que se encuentran en la carpeta ‘src’, por lo tanto es de suma importancia ubicarse en la carpeta Rouge en la terminal para que los importes funcionen correctamente.
Es importante también tener en cuenta que el programa utiliza un método implementado en el archivo Magic que no funciona en el sistema operativo de Windows.
Lo primero que ocurrirá cuando se ejecute el programa es que deberá ingresar su nombre y la tecla enter. Luego, una vez que aparezca el mapa, solo debe ir moviéndose.

Resultado de ejecuciones

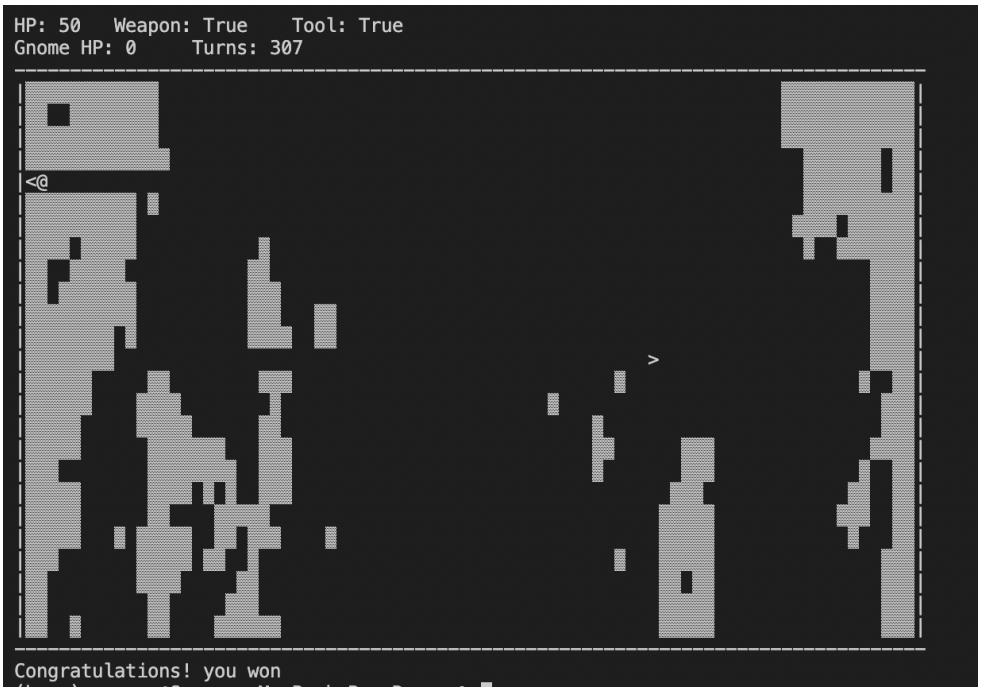


En este caso, el Gnomo mató al jugador principal porque se encontraba en un rango menor a un bloque de espacio. Como el jugador murió, se termina el juego.



En esta imagen, se puede ver como el jugador principal mató al gnomo, y en el render siguiente no aparece más, es decir, una vez que se muere, el gnomo desaparece del mapa.

Además, en la parte arriba de la captura de pantalla, aparece que el jugador posee la espada, lo que significa que era posible matarlo. En cambio, en la anterior captura, como no tenía la espada, no lo podía haber matado.



Por último, se puede ver acá como el jugador ganó dado que tenía el amuleto (bajó hasta el último nivel, lo agarró y volvió a subir) y se direccionó para el lado de las escaleras ascendentes.

En todos los prints del render, se mostrará un pequeño contador de cuantos hit points tiene cada uno, de los turnos y de si tiene o no los distintos objetos.