



Universidad de
SanAndrés

Rogue Game

Pensamiento Computacional

Trabajo Práctico N°3

1. Objetivos

En este trabajo práctico van a implementar un mini juego al estilo de los originales *roguelike games*.

2. Introducción

Circa el año 1980, conoció la luz un juego llamado **Rogue**. Este juego fue pionero en muchos sentidos, al punto que muchos videojuegos pueden trazar su esencia hasta el Rogue, siendo en conjunto llamados *roguelike games*, o *juegos similares al Rogue*, en español. Un clásico ejemplo de video juego moderno, que puede trazar su estilo hacia el Rogue, es el **Diablo**, y toda su saga, cuya última entrega fue presentada en 2021, y está confirmada una entrega en el futuro. Es decir, es un estilo de juego que, al día de hoy, sigue estando vigente.

En la figura 1. se muestra cómo se ve un juego (**Nethack**) en la terminal de GNU/Linux, en 2022 (obviamente, existen otros juegos, por ejemplo, a través de **Steam**, con otro tipo de gráficos).



Figure 0.1: Nethack gameplay

4. Desarrollo del TP / Especificaciones

Comenzaremos con un juego simple, 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, en un calabozo de varios niveles.

El juego básico debe estar compuesto por:

- Dos personajes:
 - Un personaje principal, el jugador (denotado por el caracter @).
 - Un gnomo (denotado con el caracter g).
- Algunos objetos:
 - Un arma, que debe ser mostrada en la pantalla con el caracter /.
 - Un pico, que se debe mostrar como ^.
 - Un tesoro a recolectar, que se debe mostrar con el caracter *.
- Un calabozo, con múltiples niveles, dentro de una pantalla de 25 filas por 80 columnas.

Los distintos niveles del calabozo estarán compuestos por:

- Paredes de roca, que se verán como █.
- Paredes indestructibles que se verán como los siguientes caracteres: - o |.
- Escaleras ascendentes (<) y escaleras descendentes (>), que permiten cambiar de nivel.

Dados estos elementos, en la siguiente sección se explica el objetivo del juego.

4.1 Objetivo del juego

El personaje principal comienza en el nivel 1 de los calabozos y debe descender al menos 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.

4.2 Personajes

En el juego hay, al menos, 2 personajes: un personaje principal y un gnomo. Todos los personajes comienzan con una cierta cantidad de puntos de vida, también llamados *Hit Points* o HP.

Personaje Principal (Jugador) Es el personaje que el jugador controla. Este personaje debe poder moverse hacia los lados —usando las teclas `w`, `a`, `s`, y `d`— y subir y bajar escaleras, usando las teclas `<` para subir y `>` para bajar.

Este personaje, al moverse hacia un casillero que contiene un objeto puede levantarlo, en cuyo caso, pasa a estar en posesión del mismo. El objeto puede ser levantado automáticamente o se puede optar por levantarlo presionando una tecla, por ejemplo, la letra `p` (de `pick-up`).

Si el personaje se mueve hacia un casillero donde se encuentra otro personaje, lo ataca.

Gnomo Es un personaje que aparece en alguno de los niveles del calabozo, que se puede mover, y puede atacar al jugador si este se encuentra cerca.

Opcionalmente puede haber múltiples gnomos, u otros personajes.

Otros personajes Es posible agregar otros personajes, si se desea, y la lógica de su inserción en el juego y el comportamiento del personaje queda a criterio del equipo que desarrolla el juego. Por ejemplo, pueden agregarse personajes que ayudan al jugador, que atacan al jugador, que hablan con el jugador, etc. Hacerlo es completamente optativo, pero en caso de hacerlo, es recomendable utilizar clases para representarlos.

4.3 Objetos

El personaje principal puede encontrarse con los siguientes objetos, ubicados en algún lugar del calabozo:

- Un arma
- Un pico
- Un tesoro


Arma El arma le sirve al personaje principal para atacar al gnomo. No debe ser obligatorio atacar al gnomo para ganar el juego.

Pico El pico es un objeto que, de tenerlo, el personaje principal puede romper paredes de roca y, consecuentemente, moverse al espacio que se genera al eliminar la pared. De romper una pared, la misma deja de estar en el calabozo, es decir, no se gana la capacidad de atravesarlas, sino de abrirse paso a través de ellas.

Tesoro El tesoro es el único objeto que el personaje principal debe recolectar, obligatoriamente, para poder ganar.

4.4 Calabozo

Cada nivel del calabozo, como se indica en la sección anterior, está compuesto por paredes de roca y paredes indestructibles, escaleras y espacio libre. Tal como se listó anteriormente, el contenido del calabozo es:

- Paredes de roca, que se verán como .
- Paredes indestructibles que se verán como los siguientes caracteres: - o |.
- Escaleras ascendentes (<) y escaleras descendentes (>), para cambiar de nivel.

En el archivo `mapping.py`, provisto por los docentes, se encuentra una función para generar el calabozo de manera aleatoria, compuesto únicamente por paredes de roca y espacios libres.

La siguiente figura muestra un calabozo donde además se muestra el jugador.

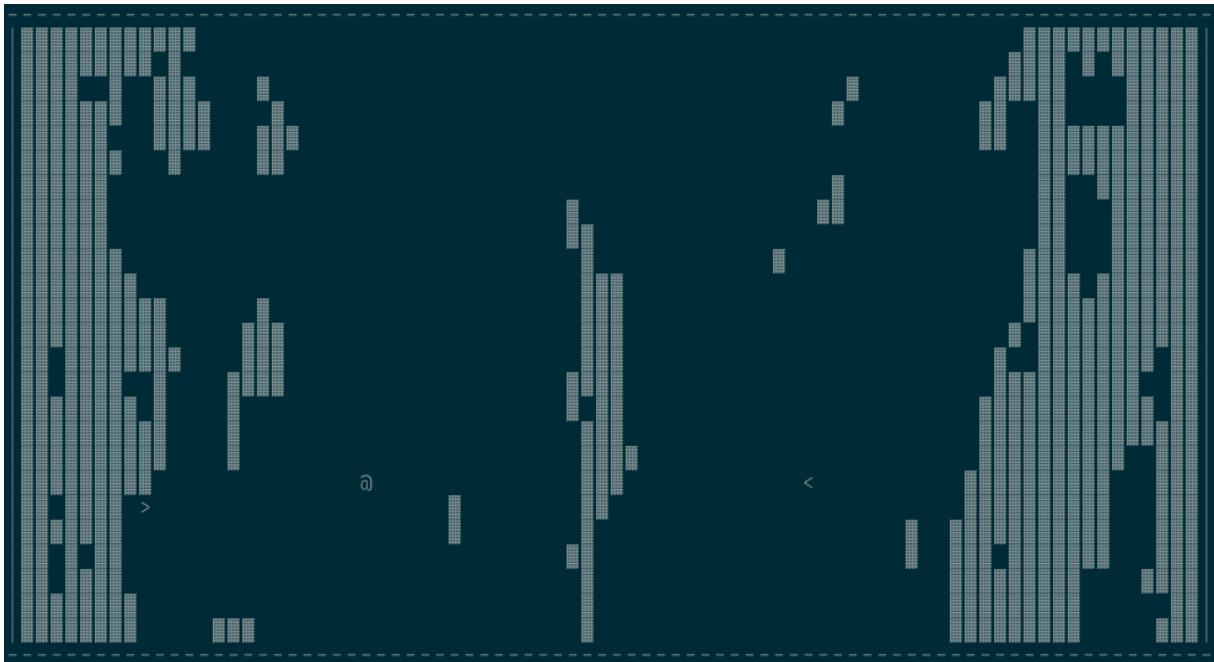


Figure 0.2: Vista del calabozo durante el juego

Al calabozo se le deben agregar otros personajes (gnomo, mínimamente) y objetos, que aquí no se muestran.

4.5 Plantillas

Junto con este enunciado se encuentran unas plantillas en las que se pueden basar para resolver el trabajo práctico. Hay 2 plantillas distintas, cada una compuesta por múltiples archivos. Una de las plantillas, ubicada en el subdirectorio `oop`, está basada en la programación orientada a objetos (OOP, por las siglas en inglés). La otra de las plantillas, ubicada en el subdirectorio `nooop`, está basada en los temas vistos hasta el momento, sin incluir OOP.

Cada grupo tiene la libertad de elegir la plantilla que desee y utilizarla, o comenzar desde 0.

Ambas plantillas tienen los siguientes archivos:

- `mapping.py`: contiene lo necesario para generar el calabozo y utilizarlo, sin embargo, hay algunas cuestiones de uso que se detallan abajo.
- `actions.py`: este archivo debe ser editado para poder hacer una prueba mínima. Actualmente contiene la firma de distintas funciones, y deben ser completadas.
- `human.py`: contiene el código para crear al personaje.
- `items.py`: contiene el código para crear items.

Todos los archivos pueden ser modificados y, como se mencionó, son plantillas, se pueden modificar a discreción del grupo de desarrolladores.

En particular, el archivo `mapping.py` contiene 3 funciones a ser implementadas:

- `is_free()`: dados los parámetros de la función, nos indica si una ubicación del calabozo está libre de otras entidades, por ejemplo, de un gnomo (pero sin considerar paredes, escaleras, items, etc).

```
1 def is_free(..., xy: Location) -> bool:
2     """Check if a given location is free of other entities."""
3     # completar
4     raise NotImplementedError
```

- `get_path()`: es similar a la anterior. Dadas dos ubicaciones en el calabozo, si se puede llegar de una a la otra, debe retornar el camino que las une (la secuencia de `Locations` que las une), no solamente indicar si el camino existe. La implementación más simple de esta función, es recursiva.

```
1 def get_path(..., initial: Location, end: Location) -> bool:
2     # completar
3     raise NotImplementedError
```

- `are_connected()`: dadas dos ubicaciones en el calabozo, indicar si se puede llegar de una a la otra, es decir, si no hay paredes que nos bloqueen el camino. La implementación más simple de esta función, es recursiva.

```
1 def are_connected(..., initial: Location, end: Location) -> bool:
2     """Check if there is walkable path between initial location
3         and end location."""
4     # completar
5     raise NotImplementedError
```

5. Restricciones

La realización de este trabajo comprende las siguientes restricciones.

Integrantes

El trabajo puede realizarse en grupos de hasta 2 (dos) integrantes.

Implementación

La implementación cumple con las especificaciones desarrolladas en el punto **4. Desarrollo del TP / Especificaciones**.

Reusabilidad

El o los módulos desarrollados tiene que poder ser *importados* en python, poniendo a disponibilidad de un tercero las funciones desarrolladas, pero sin ejecutar el juego. El juego debe ser ejecutado únicamente mediante la ejecución del programa o la ejecución de una función. Por ejemplo, para jugar se puede ejecutar el siguiente comando desde una consola de Python:

```
1 >>> from rogue import game
2 >>> game()
```

Módulos

Sólo está permitido utilizar, además de los módulos estándar de python, los siguientes módulos:

- módulos propios
- `numpy`
- `scipy`
- `matplotlib`
- `tcod`
- `getch`

Si se desea utilizar cualquier otro módulo, por ejemplo `pytorch`, se debe consultar a los docentes antes y brindar una justificación de su uso.

6. Entrega

Se debe realizar una entrega digital a través del campus de la materia, compuesta por un único archivo comprimido con todos los contenidos del trabajo.

El nombre de dicho archivo debe cumplir el siguiente formato:

```
1 tp2_legajoA_legajoB.zip
```

donde `legajoA` y `legajoB` son los legajos de ambos integrantes del grupo (ordenados de menor a mayor) y `.zip` es la extensión del archivo comprimido (también puede ser `.zip`, `.tar`, `.gz`, o `.tar.gz`, pero no puede ser `.rar`).

A su vez, el archivo comprimido debe contener los siguientes elementos:

1. El código fuente en (posiblemente) varios archivos de python (`.py`).
2. La documentación del desarrollo **en formato pdf**, que contiene los siguientes items:
 1. Carátula del trabajo práctico, nombre y apellido del alumno y dirección de correo electrónico.
 2. Objetivos del trabajo.
 3. Diseño del programa. Esto incluye una explicación sobre cómo funcionan las distintas partes del programa y las alternativas de diseño que fueron consideradas.
 4. Reseña sobre problemas encontrados y soluciones.
 5. Indicaciones para ejecutar correctamente el programa y las pruebas.
 6. Resultados de ejecuciones, incluyendo capturas de pantalla (como imagen o texto), bajo condiciones normales e inesperadas de entrada.
 7. Bibliografía.

Nota: el informe debe estar redactado en *correcto* castellano.