UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

Labor et Sapientia Libertas

Sistemas Operativos

“Kawaii Monsters”

Equipo:
Hernández Montellano Carlos
Martínez Robles Liz Velia
Miguel Sánchez Itzel Mariela

602-A

Fecha de entrega: 30 de Junio de 2017

I.INTRODUCCIÓN

El proyecto final consistió en la implementación de un juego utilizando hilos. Crear un **hilo** es heredar de la clase **Thread** y definir el método **run()**. Luego se instancia esta clase y se llama al método **start()** para que arranque el **hilo**:

```
public Hilo extends Thread{
   public void run( )  {
      // Aquí el código que se desea haga el hilo
   }
};
//declaración y creación de un hilo
Hilo elHilo = new Hilo();
//ejecuta el hilo
elHilo.start();
```

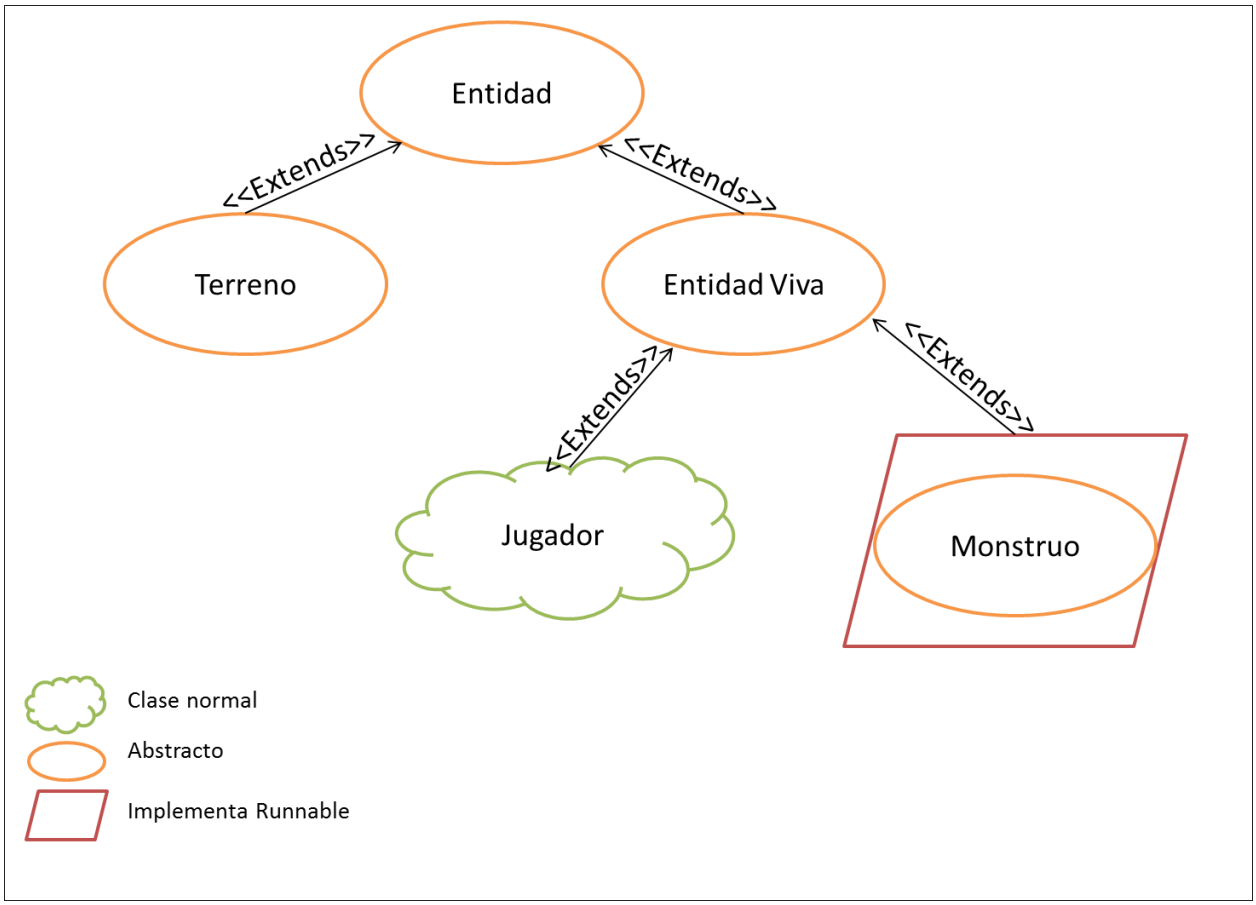
Todos los hilos de un programa comparten el espacio de memoria, haciendo posible que dos hilos accedan la misma variable o corran el mismo método de un objeto al "mismo tiempo". Se crea así la necesidad de disponer de un mecanismo para bloquear el acceso de un hilo a un dato crítico si el dato está siendo usado por otro hilo.

En el desarrollo de este reporte se detalla el uso de los hilos y el funcionamiento del juego.
El juego que se decidió implementar es del género roguelike, el cual se basa en la exploración de mazmorras. Diseñado para un jugador, número limitado de vidas, monstruos como enemigos, premisa del juego sencilla.

II. DESARROLLO

El objetivo del juego consiste en que el jugador debe limpiar la pantalla, es decir, eliminar a todos los monstruos. Cabe destacar que cada uno de ellos, tiene un número de daño predeterminado, por lo cual pueden infligir daño al jugador y este puede perder.

En el siguiente diagrama 1 se muestra la jerarquía de las clases implementadas en el código. Teniendo a Entidad como la clase padre de la cual heredan las demás clases, tomando ciertos atributos de ella.



1:Diagrama del diseño del juego

Cada clase *Monstruo* en el juego es un hilo, es decir, si hay n Monstruos, hay n hilos. Todos ellos implementan el Interface Runnable, con el cual realizan sus tareas asignadas en el momento programado.

Es importante mencionar que todos los hilos / Monstruos se mueven al mismo tiempo, una característica más de los juegos de este género.

El juego hace uso de dos capas:

Primera capa: El motor gráfico es el que se encarga de desplegar la matriz mundo.

Segunda capa: Sobre la matriz mundo, se encuentra la matriz de entidades vivas.

- Mundo:
  En esta capa se crea el mundo que se le muestra al usuario. Es decir sobre él es donde los monstruos y el jugador se mueven.
- Entidades vivas:
  Sobre esta capa se encuentran los monstruos y el jugador. Aquí es dónde al moverse cambian sus posiciones, sin embargo estas acciones se ven reflejadas igualmente en la capa *Mundo*.

La razón por la cual se hicieron dos distintas capas es porque de esta forma las acciones de los monstruos y el jugador no alteran el mundo por el que se mueven, y este se conserva intacto de principio a fin.

Cada *clase Monstruo* tiene su propio comportamiento definido, el cual se repetirá hasta que su atributo vida sea 0 y este desaparezca de la matriz. Esto puede llegar a suceder cuando el jugador lo ataca y la vida del Monstruo comienza a decrementarse. El decremento depende del daño asignado en el atributo Daño del jugador.

La clase Jugador también depende de un atributo Vida y un atributo Daño, es decir, tiene un número limitado de 3 vidas el cual decrementa cuando un Monstruo lo colisiona. Mientras que su atributo Daño, contiene el daño que le puede infligir a los Monstruos.

III. CONCLUSIONES

Para una mejor eficiencia del juego se hizo uso de hilos, ya que de esta forma se pueden realizar varias tareas de manera simultánea.
Usualmente para poder utilizarlos tenemos que crear clases que extiendan de la clase *Thread* y re-escribir el método principal *Run().* Método que se ejecuta primero cuando corre un hilo.

El objetivo del juego si se cumplió, ya que se le pudieron implementar a los Monstruos y al jugador, los atributos deseados. De tal forma que se siguió con el protocolo de implementación de Hilos. De manera aleatoria se generan los Monstruos y sus posiciones iniciales, mostrandolos en el mapa de tal forma que aunque tengan el mismo comportamiento lo hagan en distintas partes del mapa.

Para lograr una buena presentación se implementó una interfaz compuesta por dos capas, World y LifeMap. La primera presenta el fondo sobre el cual los personajes se mueven, mientras que la segunda clase, permite el movimiento de los personajes cambiando sus posiciones iniciales.

IV. BIBLIOGRAFÍA

Harvey M. Deitel. (2004). Como programar en C/C++ y Java. México: PRENTICE HALL.