FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2 — Algocraft

[7507/9502] Algoritmos y Programación III
Curso 1
Primer cuatrimestre de 2019

Alumno	Padron
Cabrera, Mauricio Luca	101334
Piragine, Santiago	100805
Gimenez Melchiore, J. Bautista	101846
Companys, Gonzalo Alejo	103026

Índice

1. Introducción

El presente informe reune la documentación de la solución del trabajo practico final de la materia Algoritmos y Programacion III que consiste en desarollar una version 2D del conocido juego minecraft, poniendo en uso todos los conceptos y tecnicas aprendidas a lo largo del cuatrimestre.

2. Supuestos

Para la resolucion del trabajo practico, como grupo hemos adoptado diferentes supuestos, los cuales seran listados a continuacion:

- Primer supuesto: Al iniciarse el juego, los materiales son distribuidos de manera aleatoria en el mapa.
- Segundo supuesto: El usuario no puede causar una excepción con las acciones que puede realizar en el juego.
- Tercer supuesto: Como cada material disminuye su duración dependiendo si fue golpeada por el tipo de herramienta del material adecuado, los materiales son los que realizan la accion de ser atacados.

3. Modelo de dominio

En este apartado, se explicaran brevemente las clases que componen el modelo de dominio:

Clase Juego

Representa a la aplicacion. Se encarga de crear el mapa de juego con el jugador en una posicion definida.

Clase Jugador

Representa al jugador, el cual se movera por el mapa y hara uso de herramientas para recolectar materiarles y ademas, usando dichos materiales recolectados, construir herramientas.

Clase Mapa

Representa el espacio en donde se va a llevar a cabo el juego, esto es, donde estaran dispuestos tanto los materiales como el jugador. En el mismo, ademas, el jugador podra desplazarse y recolectar los materiales.

Clase Herramienta

Representa las las herramientas, las cuales el jugador podra utilizar para recolectar materiales (al comiezno de cada juego el jugador posee un pico de madera), y ademas, podra construir diferentes herramientas dependiendo de los materiales recolectados que utilice.

Clase Desgaste

Es una clase abstracta, de la cual extienden tres clases (degastePorDurabilidad, desgastePorUsos, DesgastePorMultiplo), la cual representa la forma en la que cada herramienta/material va disminuyendo su durabilidad segun corresponda.

Clase Materiales

Es una clase abstracta, de la cual extienden Diamante, Piedra, Madera y Mental. Representa los materiales que estaran dispuestos en el mapa, los cuales el jugador podra recolectar y utilizar para la construccion de herramientas.

Clase Direccion

Es una clase abstracta de la cual extienden cuatro clases hijas (direccionAbajo, direccionArriba, direccionDerecha y direccionIzquierda). Dicha clase representa las direcciones en las que se realiza cierta accion, ya sea la de moverse del jugador y la de picar un determinado material en esa direccion.

Clase Posicion

Representa las posiciones que toman los objetos en el juego. Estas posiciones constan de una coordenada x y una coordenada y.

ElementoDelJuego

Es una clase abstracta que representa, tal como lo dice su nombre, a los elementos que son parte del juego. Esto es, las clases Material y Jugador extienden de ella.

Clase MesaDeTrabajo

Representa el "tablero.[en] el cual se llevara a cabo la construccion de las herramientas, debido a que para cada herramienta, la disposicion de los materiales sobre esta mesa de trabajo varia.

4. Diagramas de clase

A continuacion, se muestran varios diagramas de clases, en los cuales se pone en evidencia las relaciones estaticas entre las clases.

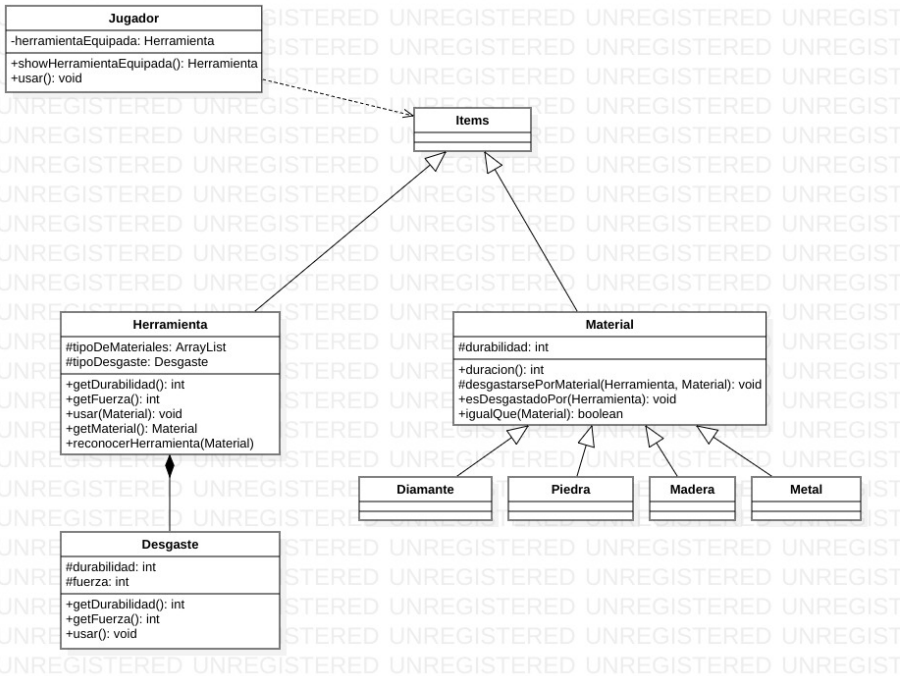

Figura 1: Diagrama Numero 1.

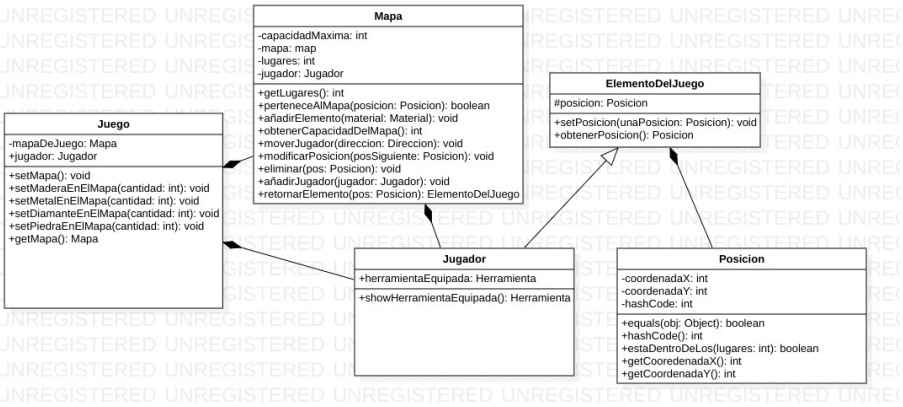

Figura 2: Diagrama Numero 2.

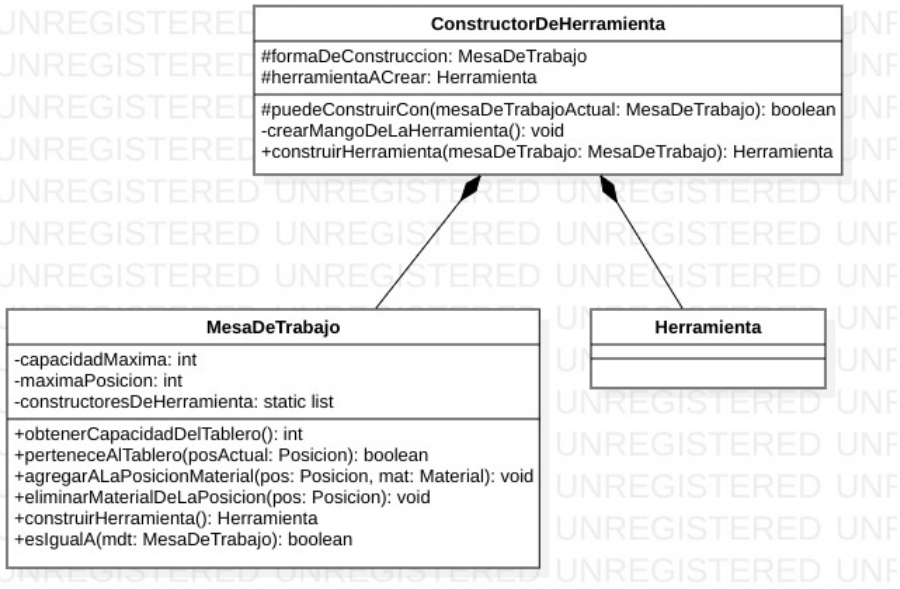

Figura 3: Diagrama numero 3.

5. Detalles de implementación

5.1. Herramienta

Para las herramienta hicimos uso del patron double dispatch, debido a que dependia de que material estaba hecha cada una cuando era usada para recolectar cierto material.

```
public void usar(Diamante unDiamante){
        this.tipoDeDesgaste.usar();
    }
    public void usar(Madera unaMadera){
        this.tipoDeDesgaste.usar();
    }
    public void usar(Metal unMetal){
        this.tipoDeDesgaste.usar();
    }
    public void usar(Piedra unaPiedra){
        this.tipoDeDesgaste.usar();
    }
```

5.2. Materiales

Al igual que las herramientas, la clase materiales tiene un double dispatch para desgastarse según el tipo de material de la herramienta. Para un ejemplo, abajo se mostrara el double dispatch escrito en el material Madera:

```
@Override
    protected void desgastarsePorMaterial(Herramienta unaHerramienta, Material unMaterial) {
        //Bloque de madera contra una herramienta de un material
        unMaterial.desgastarsePorMaterial(unaHerramienta, this);
```

```
}

@Override
protected void desgastarsePorMaterial(Herramienta unaHerramienta, Madera unaMadera) {
    //Herramienta de madera contra un bloque de madera
    unaMadera.durabilidad -= unaHerramienta.getFuerza();
}

@Override
protected void desgastarsePorMaterial(Herramienta unaHerramienta, Piedra unaPiedra) {
    //Herramienta de madera contra un bloque de piedra
    unaPiedra.durabilidad -= unaHerramienta.getFuerza();
}

@Override
protected void desgastarsePorMaterial(Herramienta unaHerramienta, Metal unMetal) {
    //Herramienta de madera contra un bloque de metal
}

@Override
protected void desgastarsePorMaterial(Herramienta unaHerramienta, Diamante unDiamante) {
    //Herramienta de madera contra un bloque de diamante
}
```

6. Excepciones

Para este trabajo práctico no nos parecio buena idea tener excepciones; tuvimos en consideración un caso en donde el usuario quiera golpear algo pero no tenga una herramienta, pero luego consideramos que, en el juego ya implementado, el usuario puede quedarse sin herramientas y, teniendo suficientes materiales en su inventario, puede fabricar otra, pero en el medio nos parece incorrecto que si se acerca a un material y le da click para golpearlo, el juego falle.

7. Diagramas de secuencia

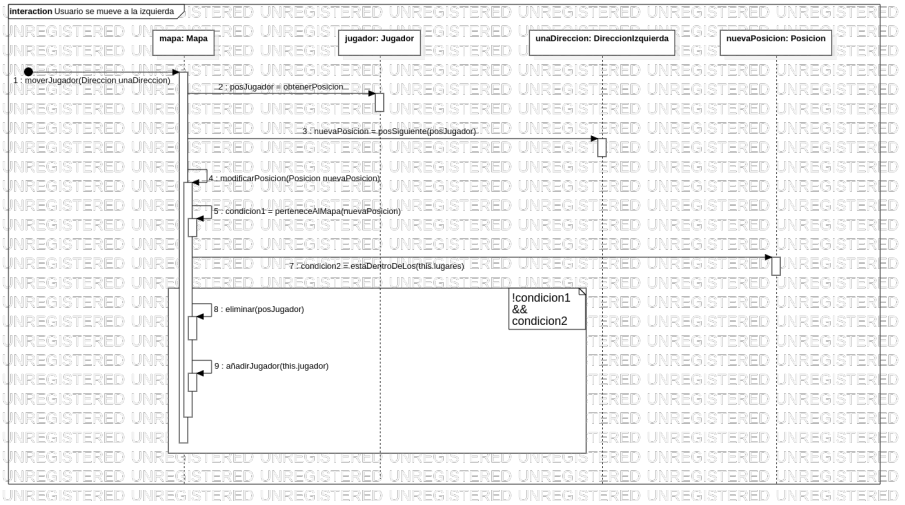

Figura 4: Usuario se mueve a la izquierda.

En el diagrama de arriba se puede ver como se procesa el movimiento del jugador (en este caso, es un desplazamiento a la izquierda) y como, si el la ubicación esta dentro de la region del mapa, y no haya ningún otro elemento en el lugar, se realiza dicha acción, caso contrario, la ubicación del jugador no se ve afectada.

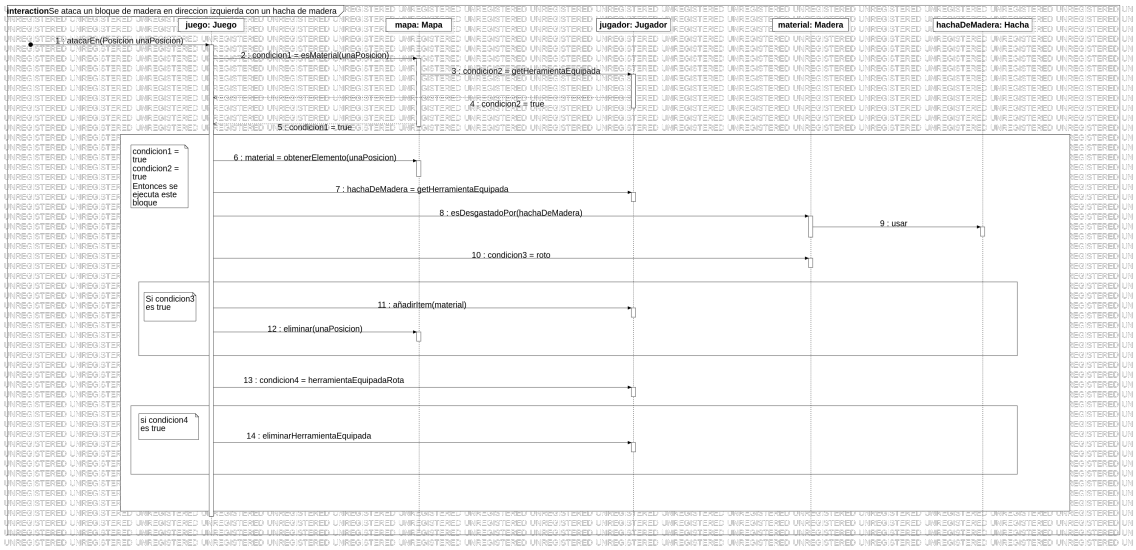

Figura 5: Usuario ataca un bloque de Madera que se encuentra a su derecha con un Hacha de Madera.

En el diagrama de arriba se puede ver como se realiza el proceso en el cual el usuario ataca un bloque de madera que se encuentra a su derecha. El usuario lo primero que hace es atacar en una direccion, en este caso a su derecha, al hacerlo se contempla si en esa direccion hay un Material, lo cual despues se realizar el proceso de desgaste del Material y la Herramienta si cumplen con las condiciones necesarias para que exista tal desgaste.

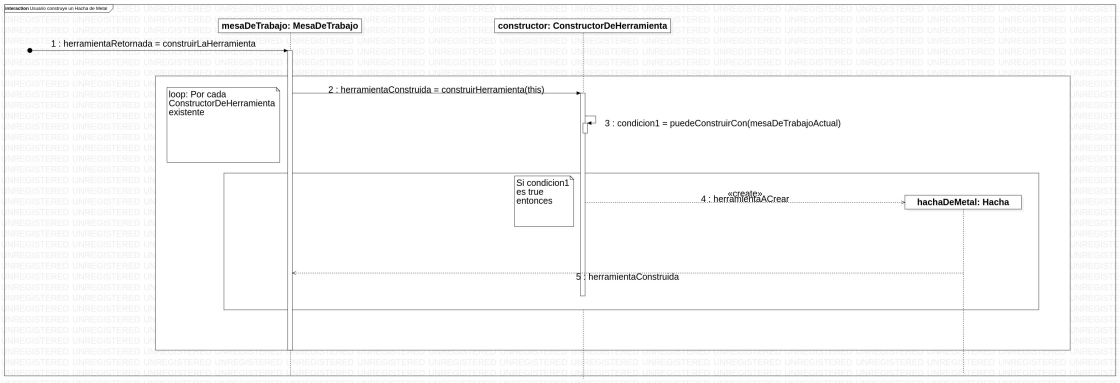

Figura 6: Usuario Construye un Hacha de Metal.

En el diagrama de arriba se puede ver como se realiza el proceso en el cual el usuario desea crear un Hacha de Metal, lo cual previamente el usuario tiene que tener cargado materiales en su Mesa de Trabajo, ahi realiza la operacion de crearLaHerramienta la cual recorre cada constructor de las herramientas(su modo o forma de ser creadas) y chequea si son iguales los ordenes de los materiales en la mesa de trabajo del usuario, en caso de que exista algun match se devuelve la herramienta, en caso contrario no devuelve ninguna herramienta.