Trabajo Práctico 2 - Java

[7507] Algoritmos y Programación III
Curso 2
Primer cuatrimestre de 2019

Mauro Parafati - 102749 - mparafati@fi.uba.ar
Nicolas Aguerre - 102145 - naguerre@fi.uba.ar
Santiago Klein - 102192 - sklein@fi.uba.ar

Índice

1. Introducción

En el presente informe se desarrolla la documentación relacionada al segundo trabajo práctico de Algoritmos y Programación III, que consiste en desarrollar en su totalidad una aplicación en Java llamada *Algocraft*. Dicha aplicación es una implementación simplificada y en 2D del juego *Minecraft*.

2. Supuestos

A continuación se redactan los supuestos que se tuvieron en cuenta para la realización del trabajo práctico:

En cuanto al *Mapa*, se consideró lo siguiente:
- Se trata de una cuadrícula de tamaño fijo.
- Mapa circular. Esto significa que si el jugador se intenta mover más allá de los límites del mismo, aparecerá al otro lado.
- Distribución random de los recursos sobre el mismo.
Decidimos que cada partida sea distinta, generando bloques de cada recurso.
- El mapa es reseteable, para permitirle al jugador distribuir nuevos materiales en el mismo.

En cuanto a la *Ventana*, se consideró lo siguiente:
- Las dimensiones se setean en base a la resolución de pantalla del usuario, pero no es modificable por el mismo.
- No incluye funcionalidad de pantalla completa.

En cuanto al *Jugador*, se consideró lo siguiente:
- El mismo puede moverse para cualquier dirección solicitada, y su orientación quedará entonces definida por este último movimiento realizado.
- Cuando se le pide al jugador que golpee, el mismo golpeará en la dirección correspondiente a su orientación, es decir que para poder golpear un recurso primero hay que setear la orientación del jugador en dirección al recurso en cuestión.

En cuanto a los *Inventarios*, se consideró lo siguiente:
- El inventario de herramientas será circular, e infinito.
- El inventario de materiales será infinito.
- Cuando una herramienta se rompe, automáticamente se elimina del inventario.
- Cuando un recurso se rompe, automáticamente se agrega al inventario.

En cuanto a las *Herramientas*, se consideró lo siguiente:
- El Pico Fino es una herramienta independiente, diferente del Pico.
- El Pico Fino se romperá si su durabilidad es menor a 1, dado a que su forma de desgaste produce el problema de que nunca llegue a cero la durabilidad.

3. Modelo de dominio

En primer lugar, para el modelado de las herramientas, se definió que cada herramienta (*Pico, Hacha,* y *PicoFino*) contiene su propia fuerza, regla de desgaste, y durabilidad (dependiendo, esencialmente, del material del cual sean fabricadas). Para esta última, se implementaron diferentes tipos de durabilidad (que varían en su forma de reducción): *DurabilidadDiezPorciento*, *DurabilidadEnFuerza* y *DurabilidadTrasDiezUsos*. Para su construcción, se modeló la clase *Constructor*.

Luego, se realizó el modelado de los*Recursos*. Los recursos son aquellos bloques que pueden ser golpeados por las herramientas en pos de obtener un material (inventariable).

Así, los recursos se definieron como *BloqueDiamante, BloqueMadera, BloqueMetal* y *BloquePiedra*. Cada uno de ellos tienen una durabilidad y pueden gastarse -o no- al ser golpeados por herramientas.

Para modelar la forma de desgaste entre herramientas y materiales, se implementaron diferentes *ReglasDesgasteRecurso*.

El jugador, por su parte, es único y contiene cierta *Orientación*, *Posición*, *Inventario de herramientas*, e *Inventario de materiales*.

Asimismo, se implementó la clase *Juego*, la cual consta de un *Constructor*, un *Mapa*, un *Jugador*, y una *Consola*.

4. Detalles de implementación

4.1. Herramientas

Las distintas herramientas heredan de la clase abstracta *Herramienta*.

Para la construcción de herramientas se utilizó el patrón **Factory**, para lo cual se implementó la FábricaHerramientas. En caso de querer instanciar una herramienta, se debe llamar al método estático correspondiente de la susodicha clase (verbigracia, *crearHachaDeMadera()*). Al llamarse, se acude a métodos estáticos de las clases *Hacha*, *Pico*, y*PicoFino*, según el material deseado, el cual seteará las características instrínsecas definidas en el modelo de dominio: durabilidad, fuerza, y regla de desgaste.

Para modelar las distintas durabilidades, se implementó la interfaz *Durabilidad*, de la cual se desprenden tres clases distintas (*DurabilidadDiezPorciento*, *DurabilidadEnFuerza*, y *DurabilidadTrasDiezUsos*), a las que se les delega la tarea de gastarse.

Además, cada *Herramienta* tiene su propia regla de desgaste de recursos. Para la implementación de las mismas, se utilizó el patrón **Decorator**. Así, cada regla de desgaste en particular, y el propio objeto Decorador, implementan la interfaz *ReglaDesgasteRecurso*, a través de la cuál se va definiendo, mediante Override, las diferentes formas de desgastar a los recursos. A raíz de esto, se soluciona el modelo del golpe de una herramienta a un recurso. Así, cada herramienta delega en su regla de desgaste la responsabilidad de desgastar, o no, al recurso correspondiente.

4.2. Recursos y materiales

Los recursos se dividieron en *BloqueDiamante*, *BloqueMadera*, *BloquePiedra*, y *BloqueMetal*.

Cada recurso hereda de la clase abstracta *Recurso*, la cual, a su vez, implementa la interfaz *OcupanteDeCelda*.

4.3. Jugador

El *Jugador* implementa también la interfaz *OcupanteDeCelda*.

El *InventarioHerramientas* se implementó a partir de un *ArrayList*. Consta de una herramienta actual (la cual es utilizada por el jugador), y ofrece la posibilidad de ir moviéndose uno a uno entre las herramientas anterior y siguiente, según corresponda, de manera circular.

El *InventarioMateriales* se implementó con un *HashMap*, en el cual se mapea la cantidad de recursos disponibles.

4.3.1. Orientación

La *Orientación* del *Jugador* es responsable de golpear a un bloque -si corresponde-, acorde a la misma.

A partir de la interfaz *Orientación*, se implementan las clases *OrientacionAbajo*, *OrientacionArriba*, *OrientacionDerecha*, y *OrientacionIzquierda*.

4.4. Constructor

El *Constructor* se implementó a partir de un *HashMap*, en el cual se mapean las diferentes recetas de construcción de herramientas, a través de arreglos hasheados, con las respectiva creación, función lambda mediante, de las herramientas.

4.5. Juego

El *Juego* se implementó mediante el patrón **Singleton**. Se encarga de inicializar los componentes de AlgoCraft (mapa, jugador, constructor, y consola), y sirve de intermediario para la interacción jugador-mapa.

4.6. Mapa

El *Mapa* fue implementado a partir de una *CuadrículaCeldas*, que no es más que una colección de objetos *Celda*. Cada *Celda*, por su parte, tiene una Posición, y un ocupante. Los ocupantes posibles implementan la interfaz *OcupanteDeCelda* previamente mencionada.

4.6.1. Celda

Una *Celda*, como se dijo antes, tiene una *Posición*, y un *OcupanteDeCelda* (por defecto, dicho ocupante es *CeldaVacia*). Se encarga de mover el ocupante -si corresponde-, y actualizar su *Posición*.

4.6.2. Posición

Cada *Posición* se encarga de obtener sus posiciones adyacentes, ya sea la de arriba, abajo, izquierda o derecha. Dicho encapsulamiento otorga mucha legibilidad al código.

5. Diagramas de clase

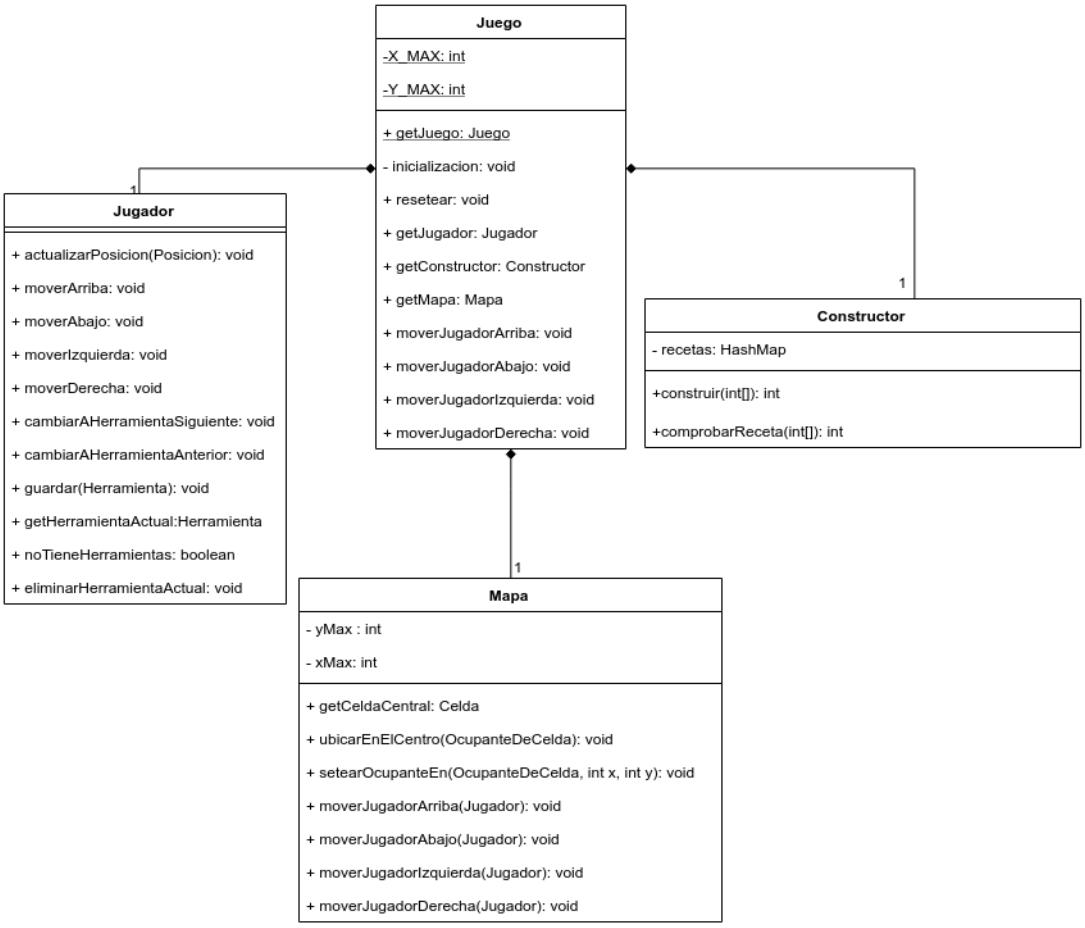

Figura 1: Diagrama general del juego.

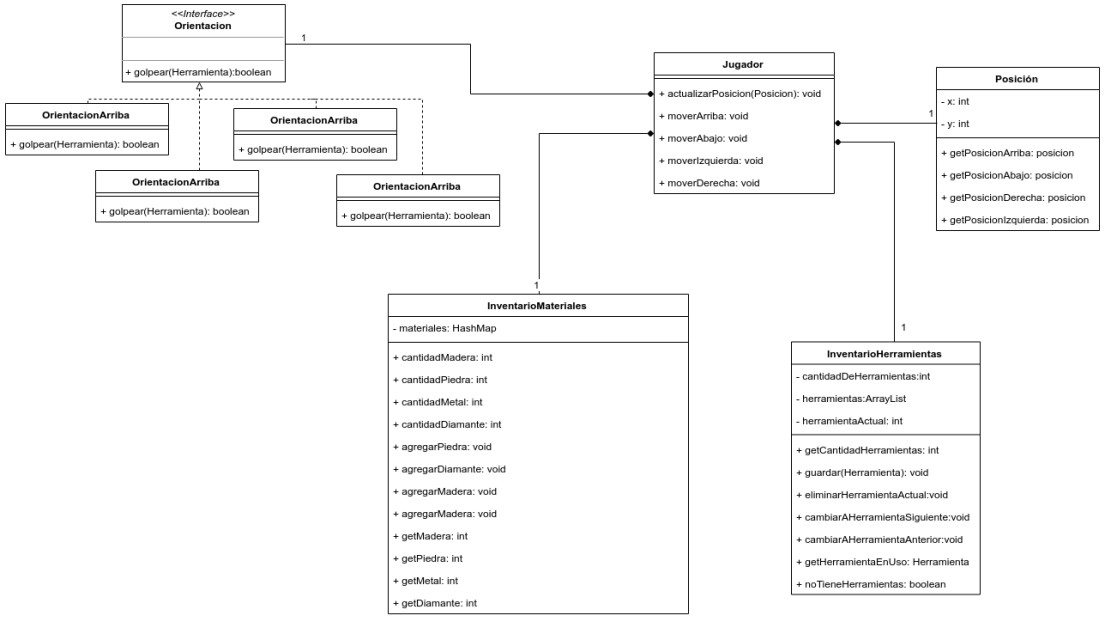

Figura 2: Diagrama del jugador.

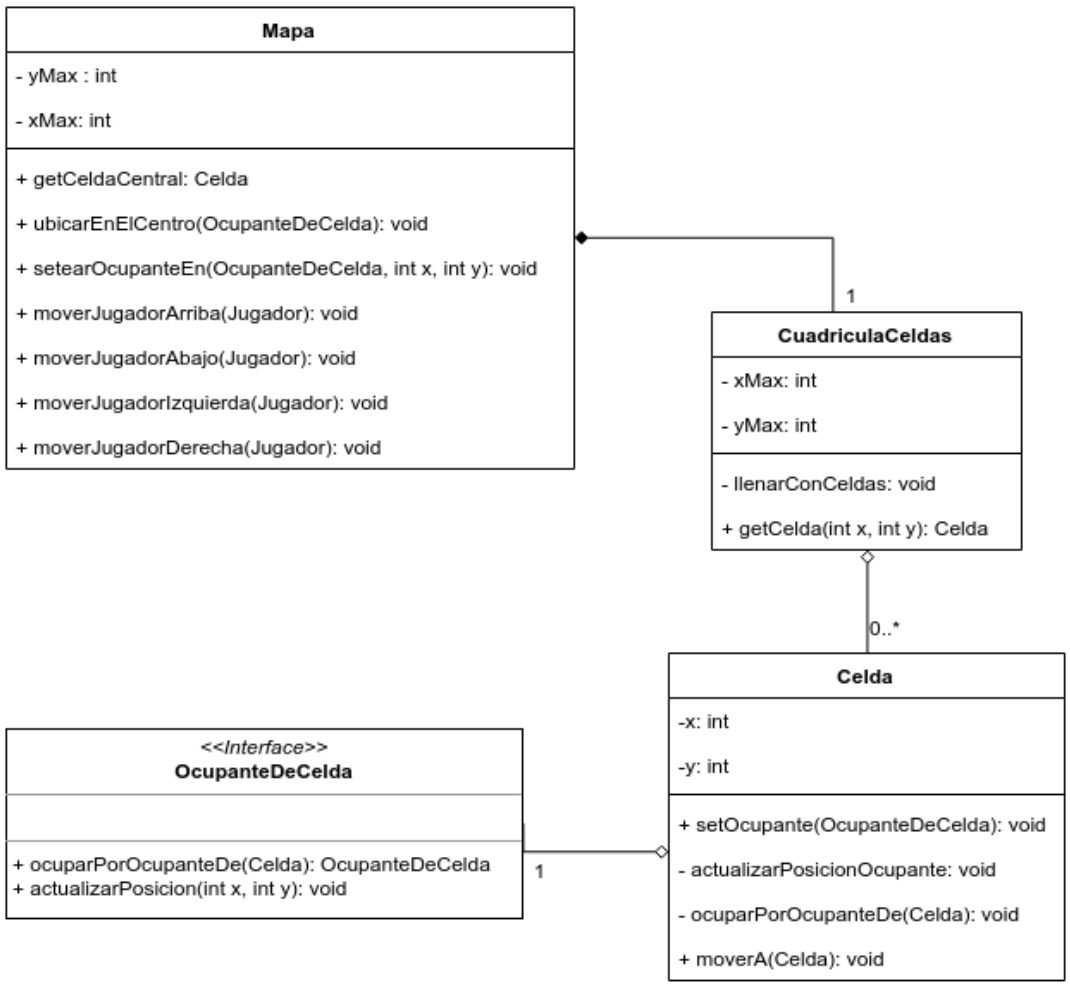

Figura 3: Diagrama del mapa.

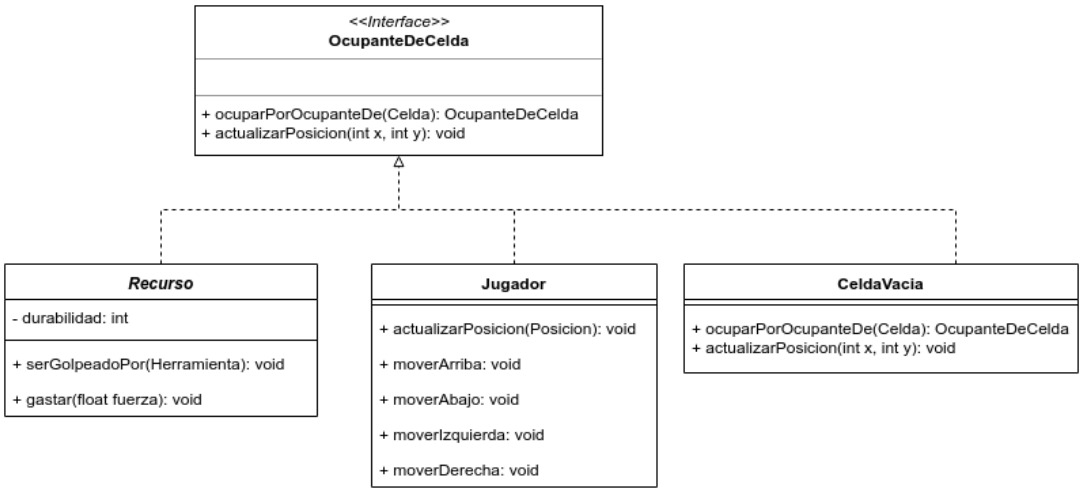

Figura 4: Diagrama de las cosas que pueden ocupar una celda.

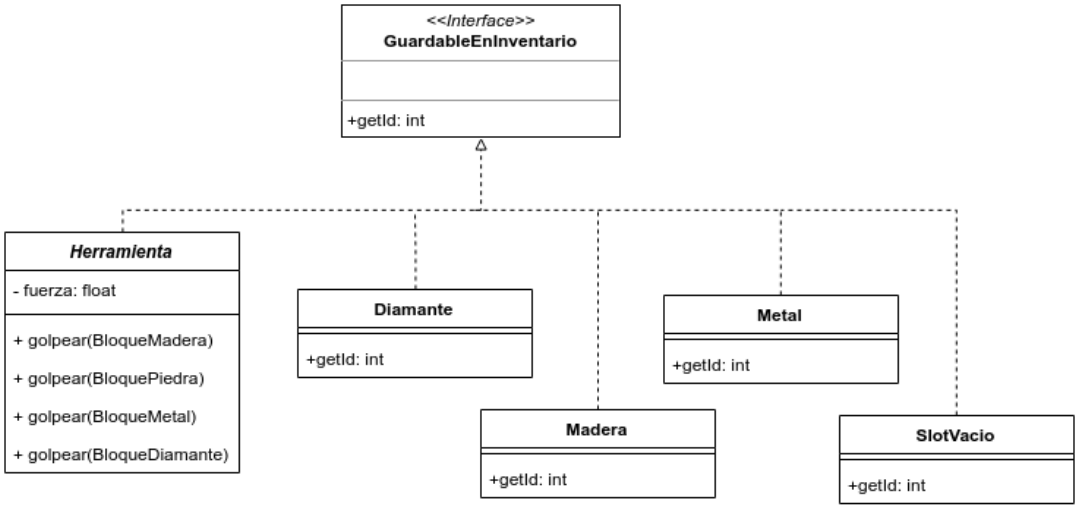

Figura 5: Diagrama de las cosas guardables en el inventario.

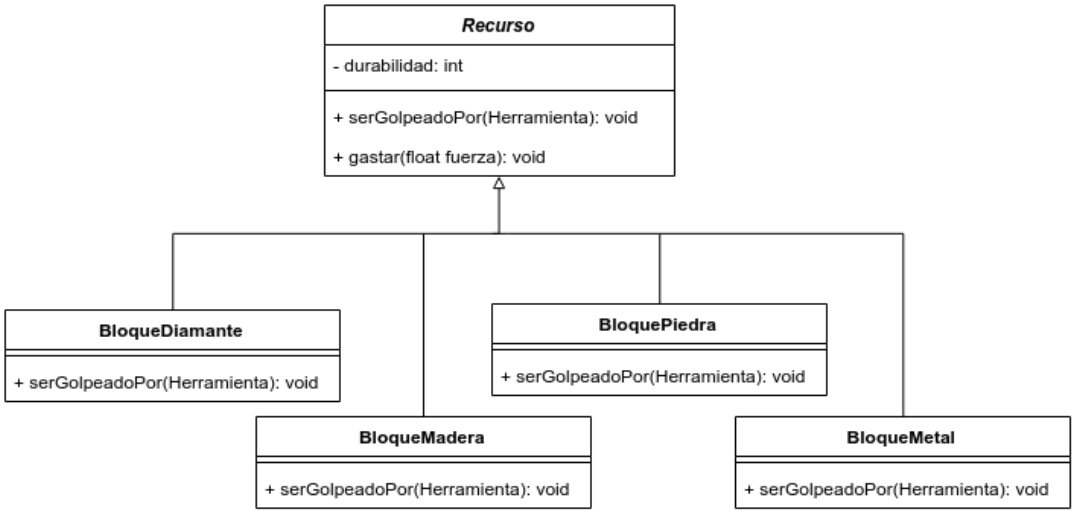

Figura 6: Diagrama de los recursos.

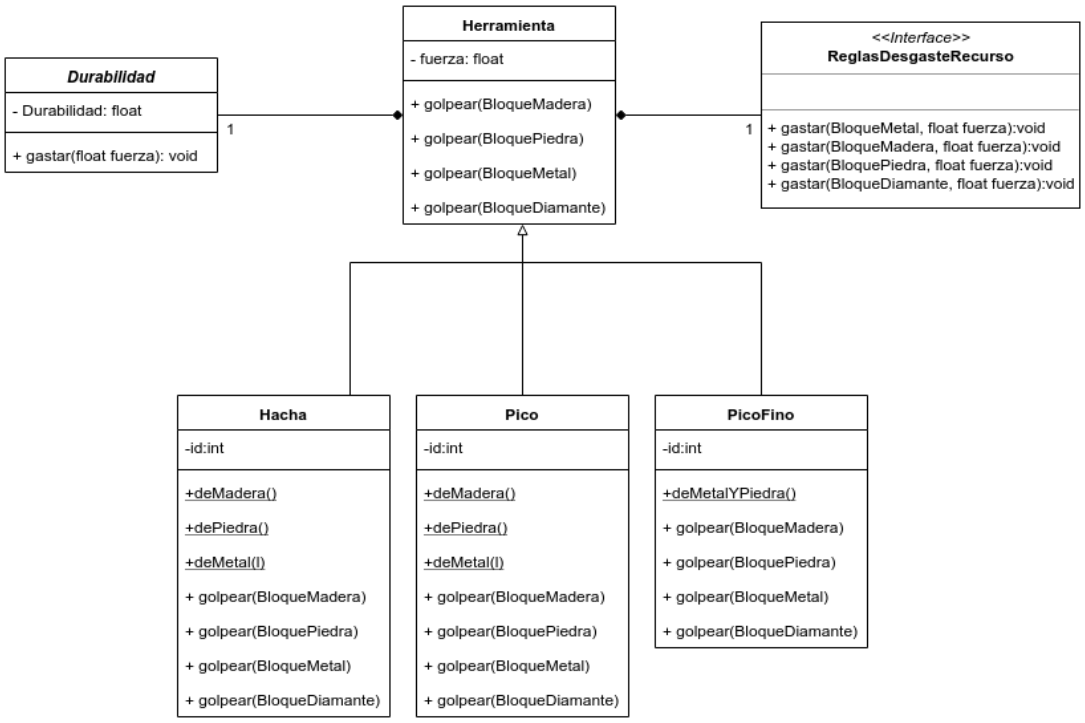

Figura 7: Diagrama de las herramientas.

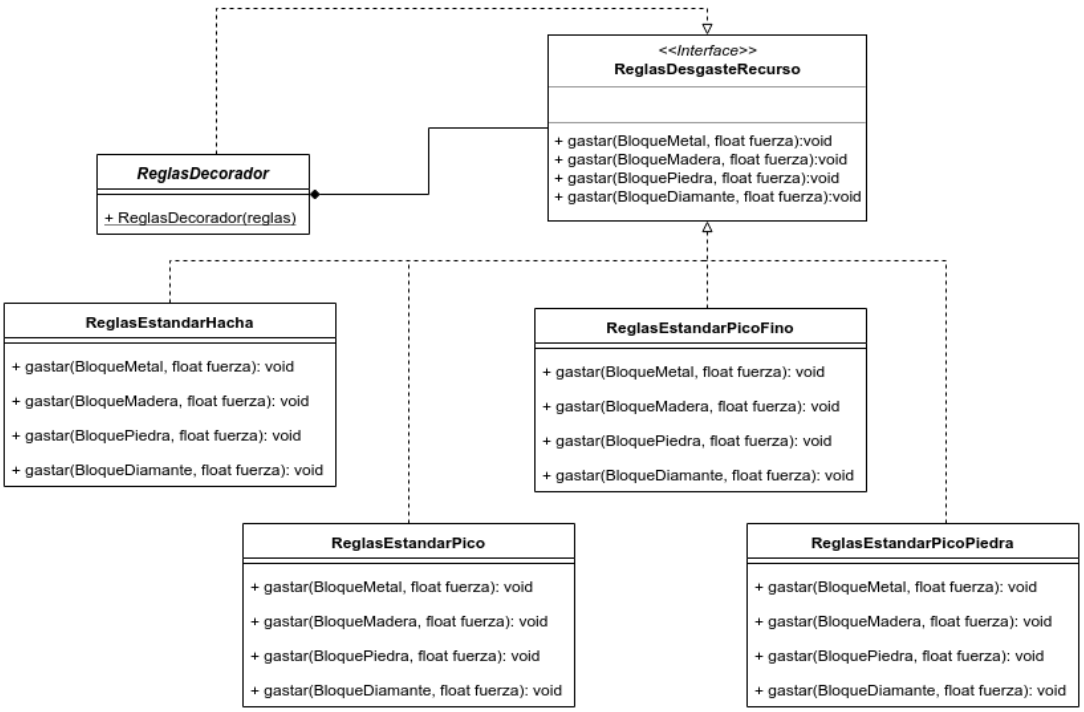

Figura 8: Diagrama de las reglas de desgaste de recursos.

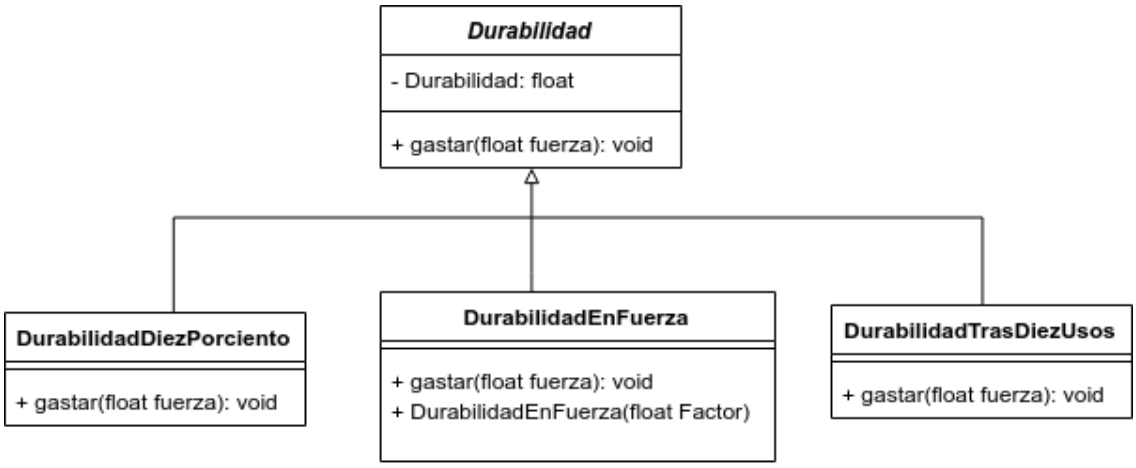

Figura 9: Diagrama de las diferentes durabilidades de herramientas.

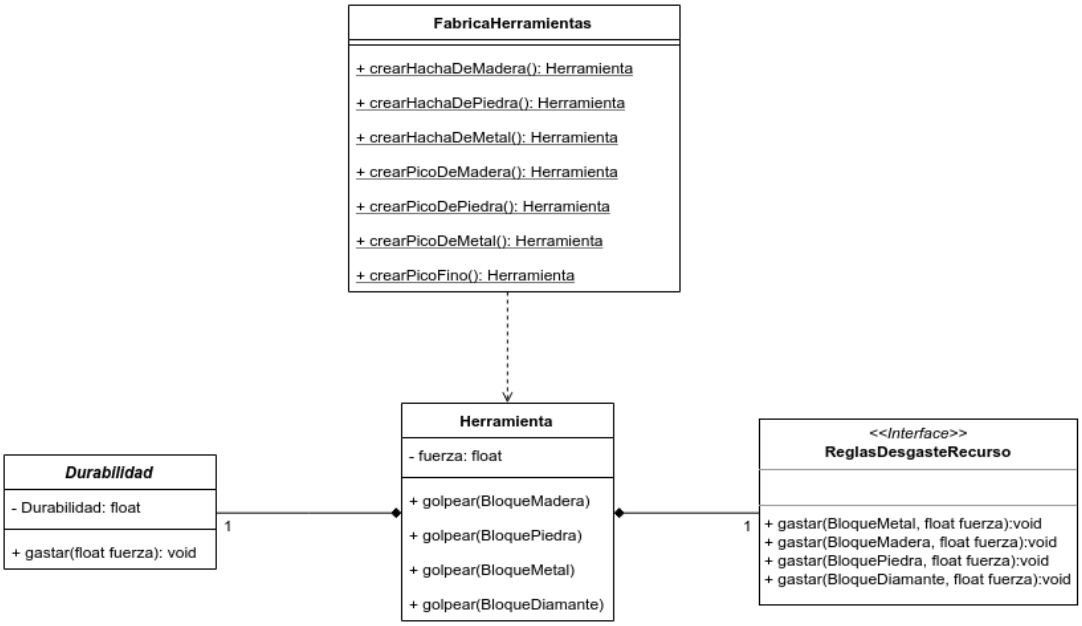

Figura 10: Diagrama del patron Factory utilizado para las herramientas.

6. Diagramas de secuencia

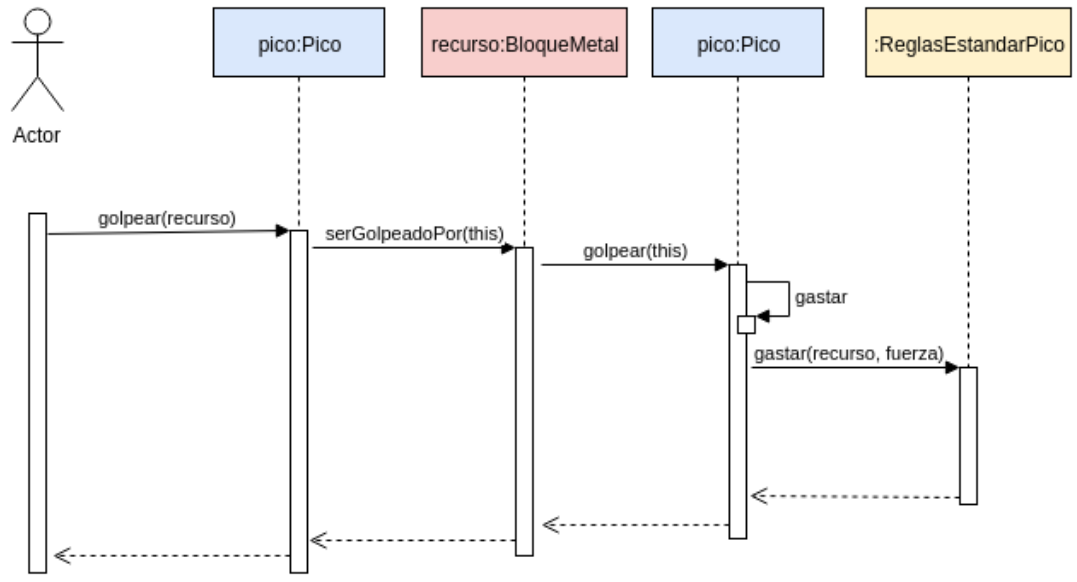

Figura 11: BloqueMetal se golpea con un Pico de Madera.

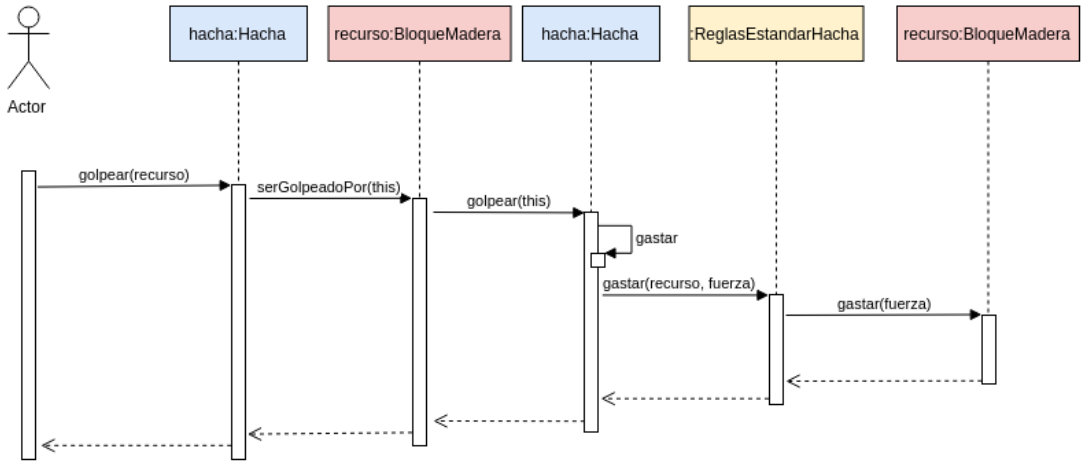

Figura 12: BloqueMadera se golpea con un Hacha de Madera

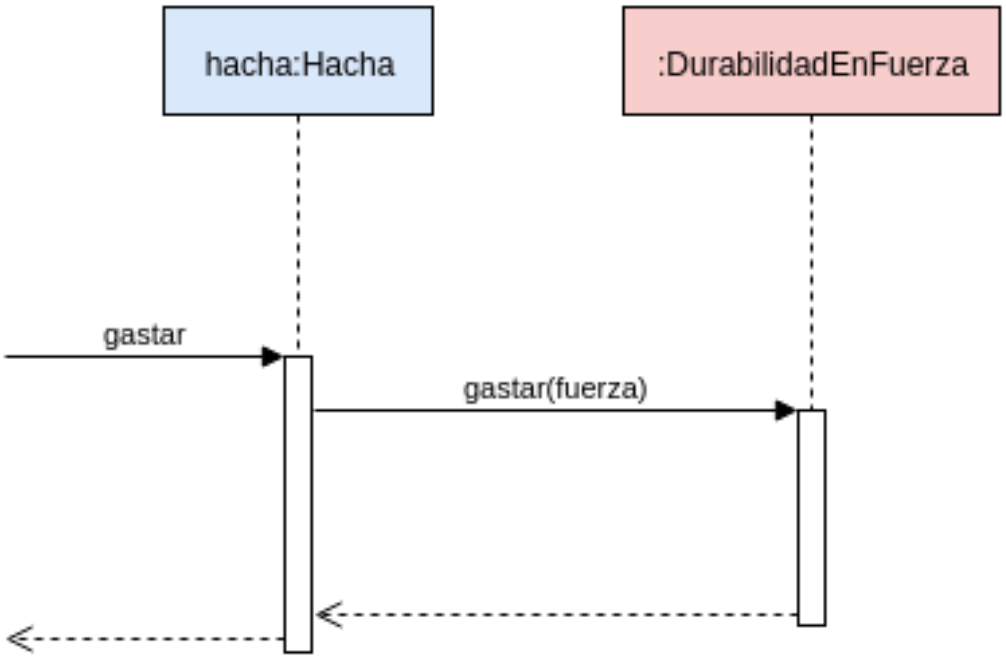

Figura 13: El hacha se gasta al golpear cualquier recurso.

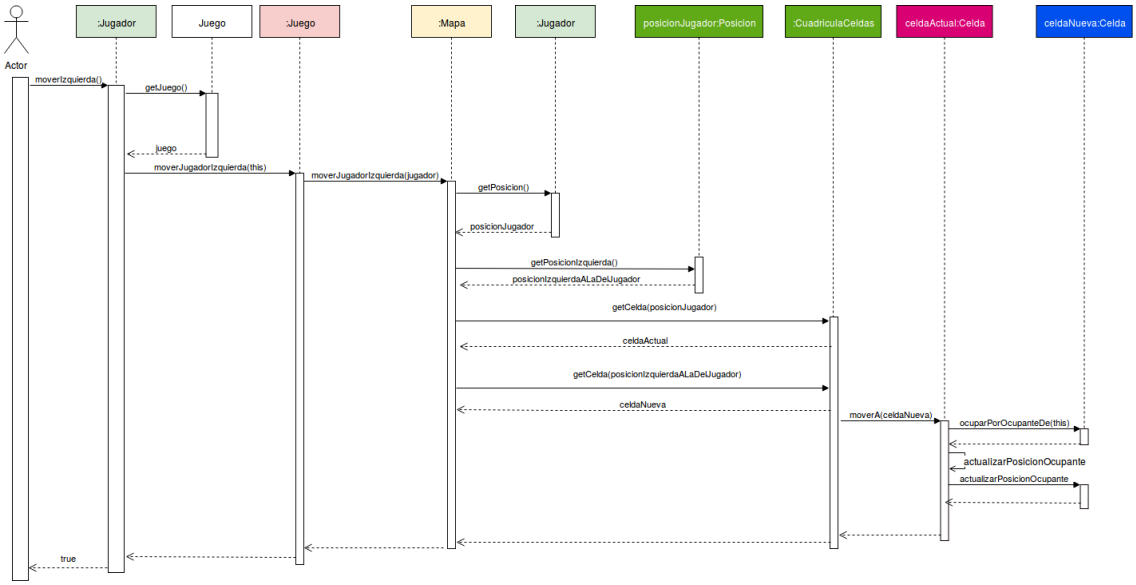

Figura 14: Jugador se mueve hacia la izquierda.

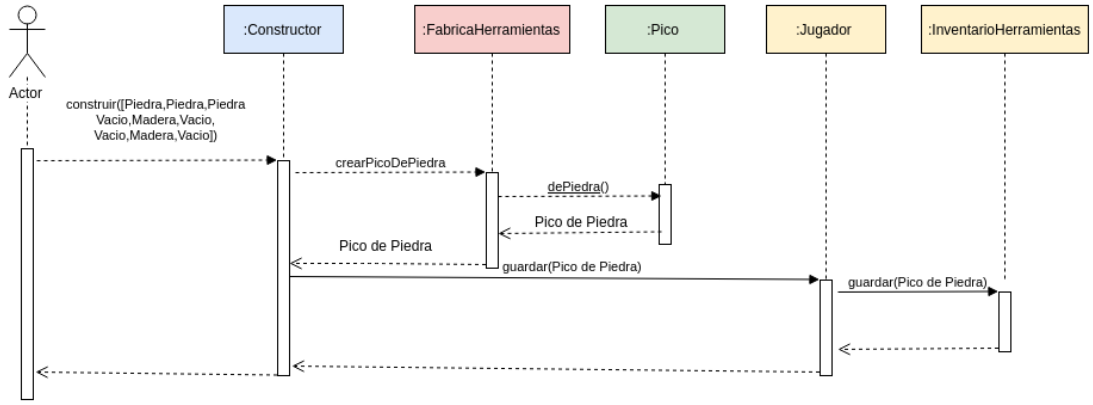

Figura 15: Construcción de un pico de piedra.

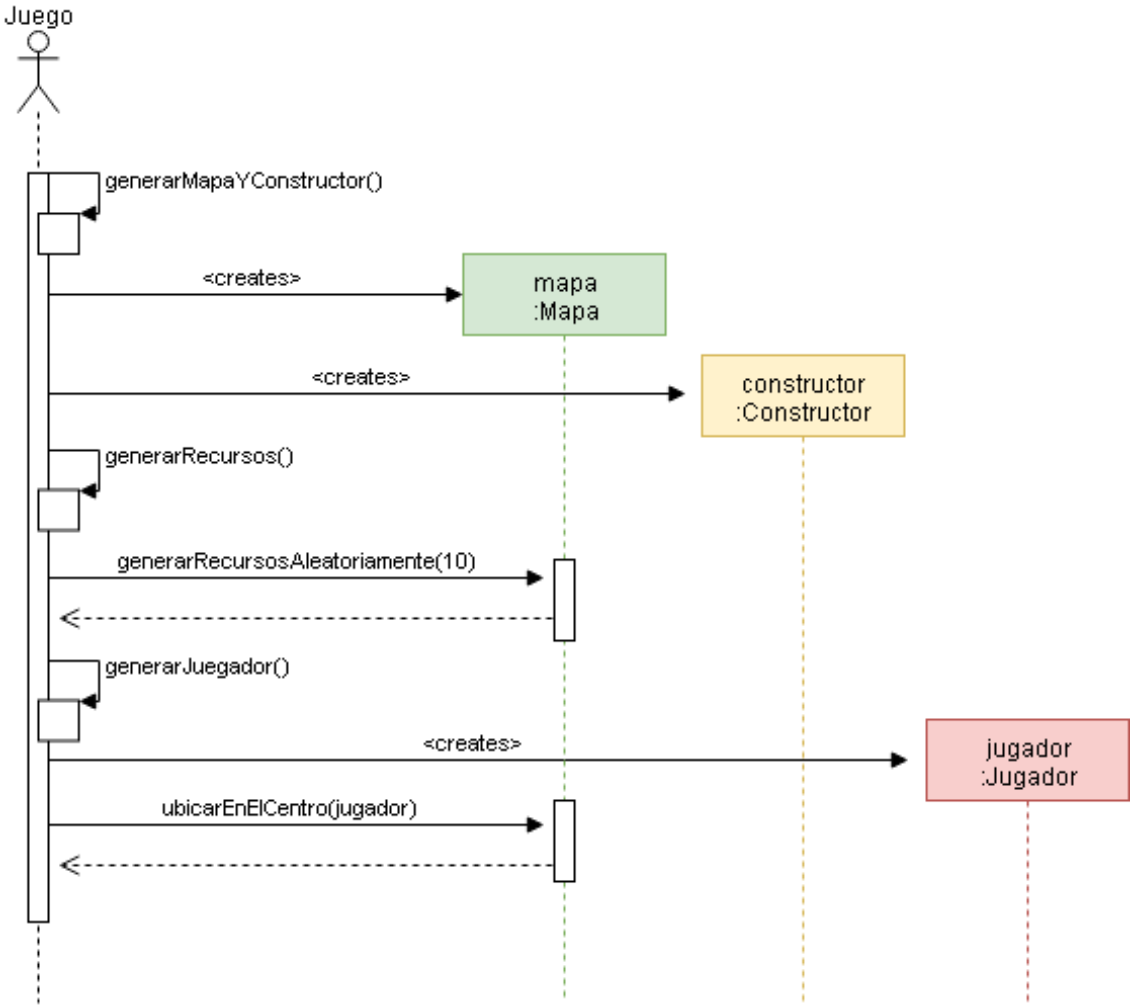

Figura 16: Inicio del juego.

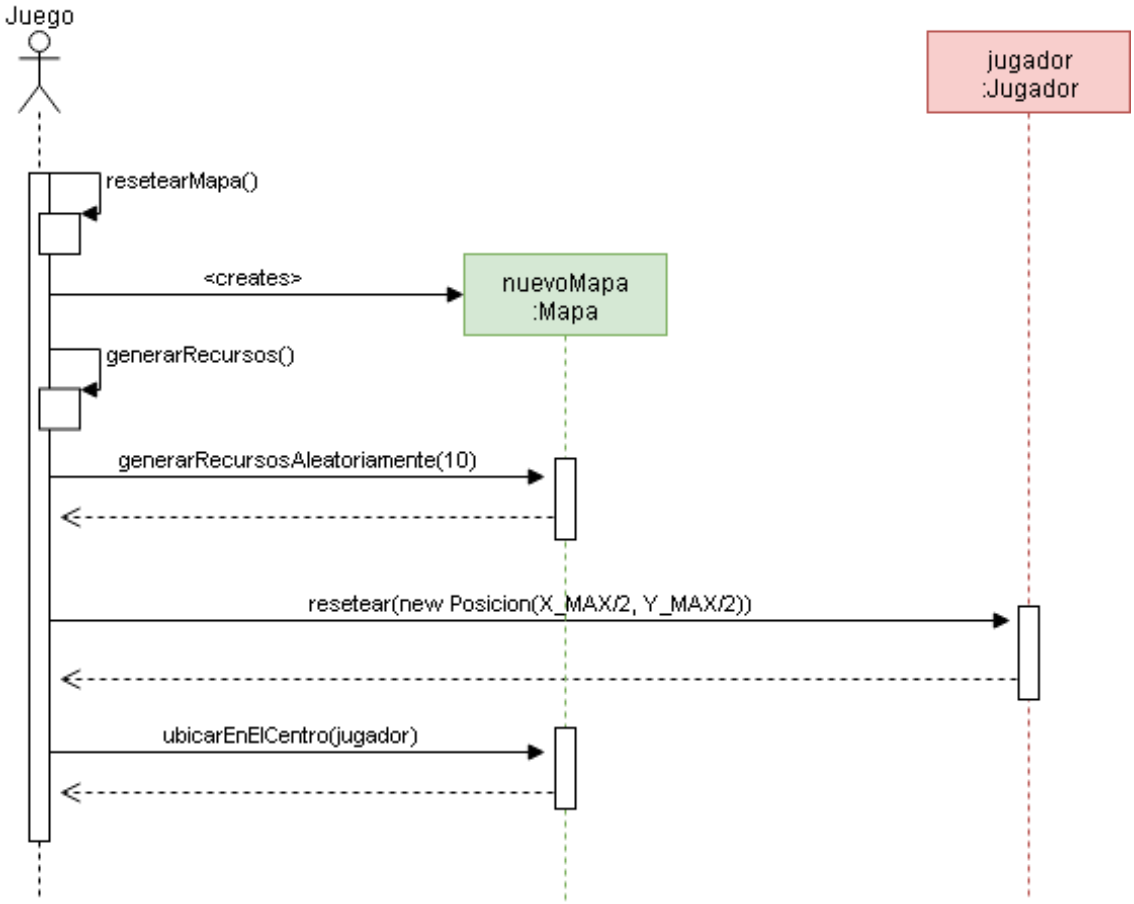

Figura 17: Reseteo del mapa.

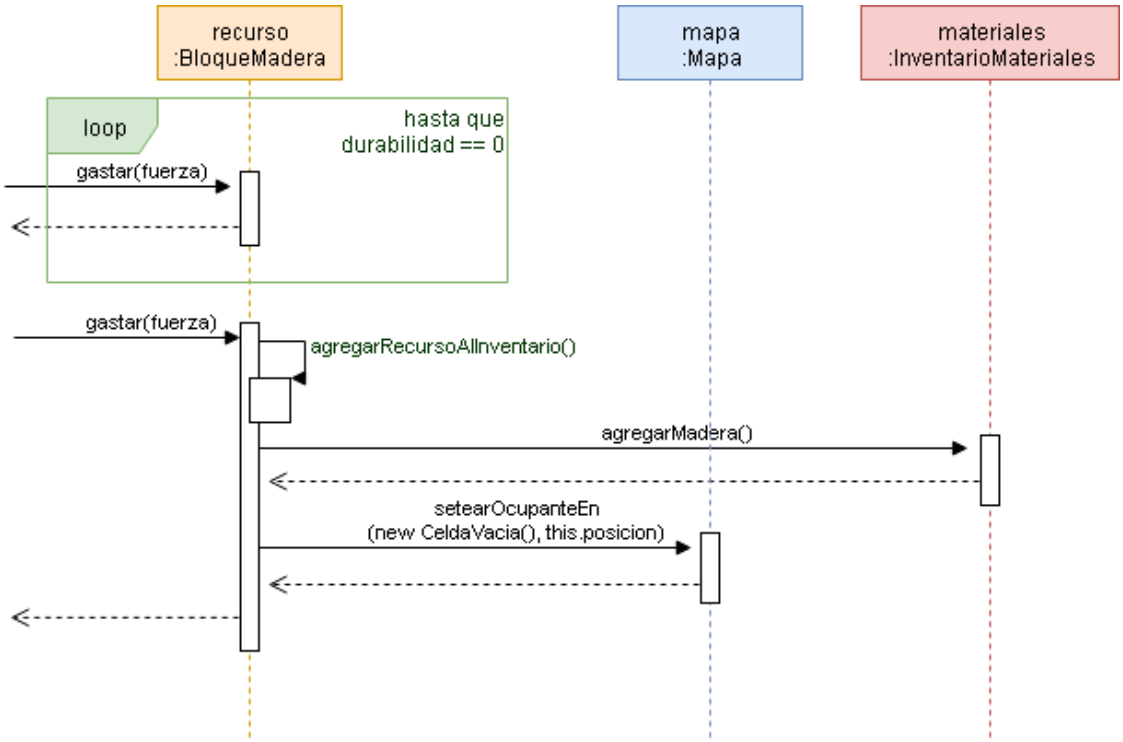

Figura 18: Agregado de un recurso al inventario.

7. Diagrama de paquetes

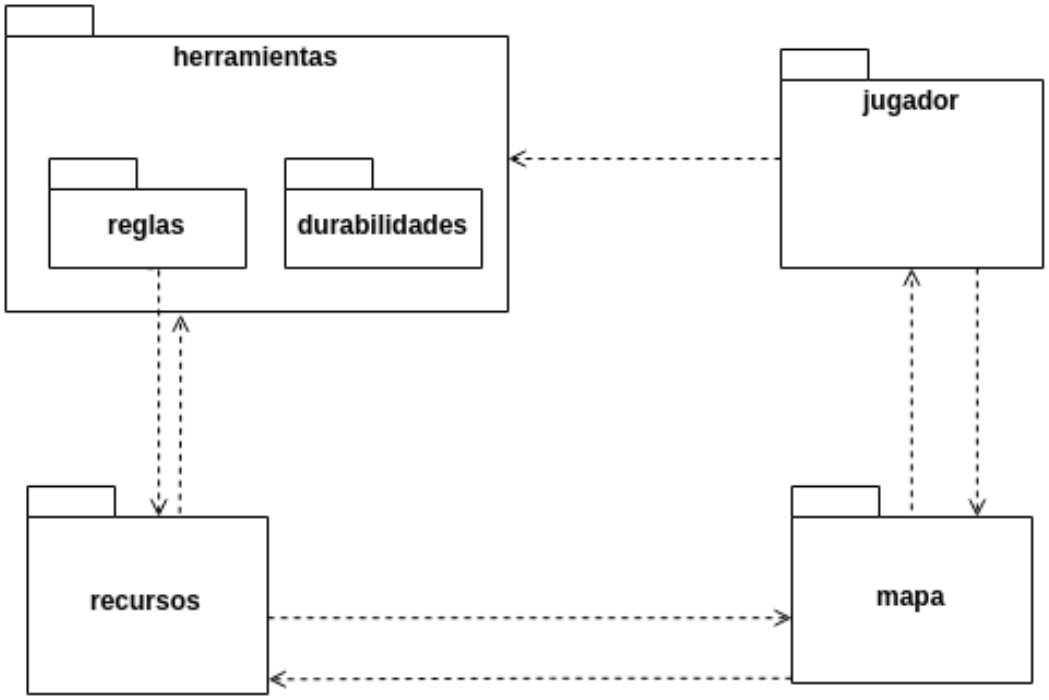

Figura 19: Diagrama de paquetes.

8. Interfaz gráfica

La interfaz gráfica esta realizada en su totalidad con el framework JavaFX. Todo el proyecto se realizó teniendo en mente el patrón de arquitectura de software MVC, que propone una separación en 3 componentes: Vista, Modelo y Controlador. A continuación se incluyen algunas capturas de pantalla de los resultados.

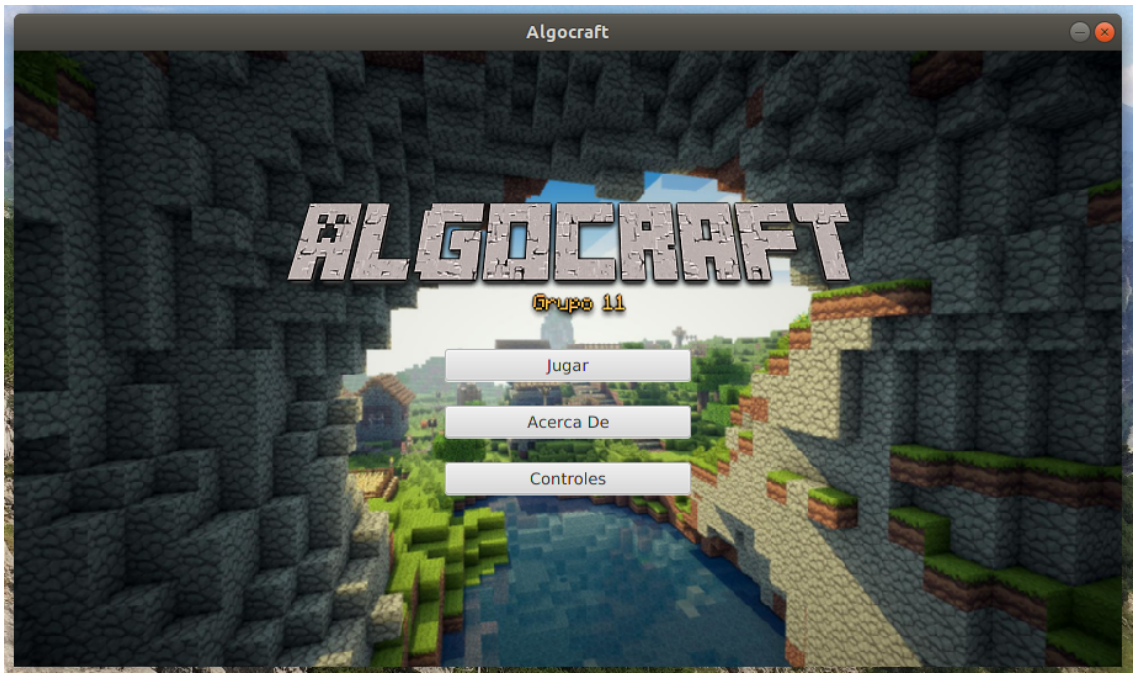

Figura 20: Menu principal.

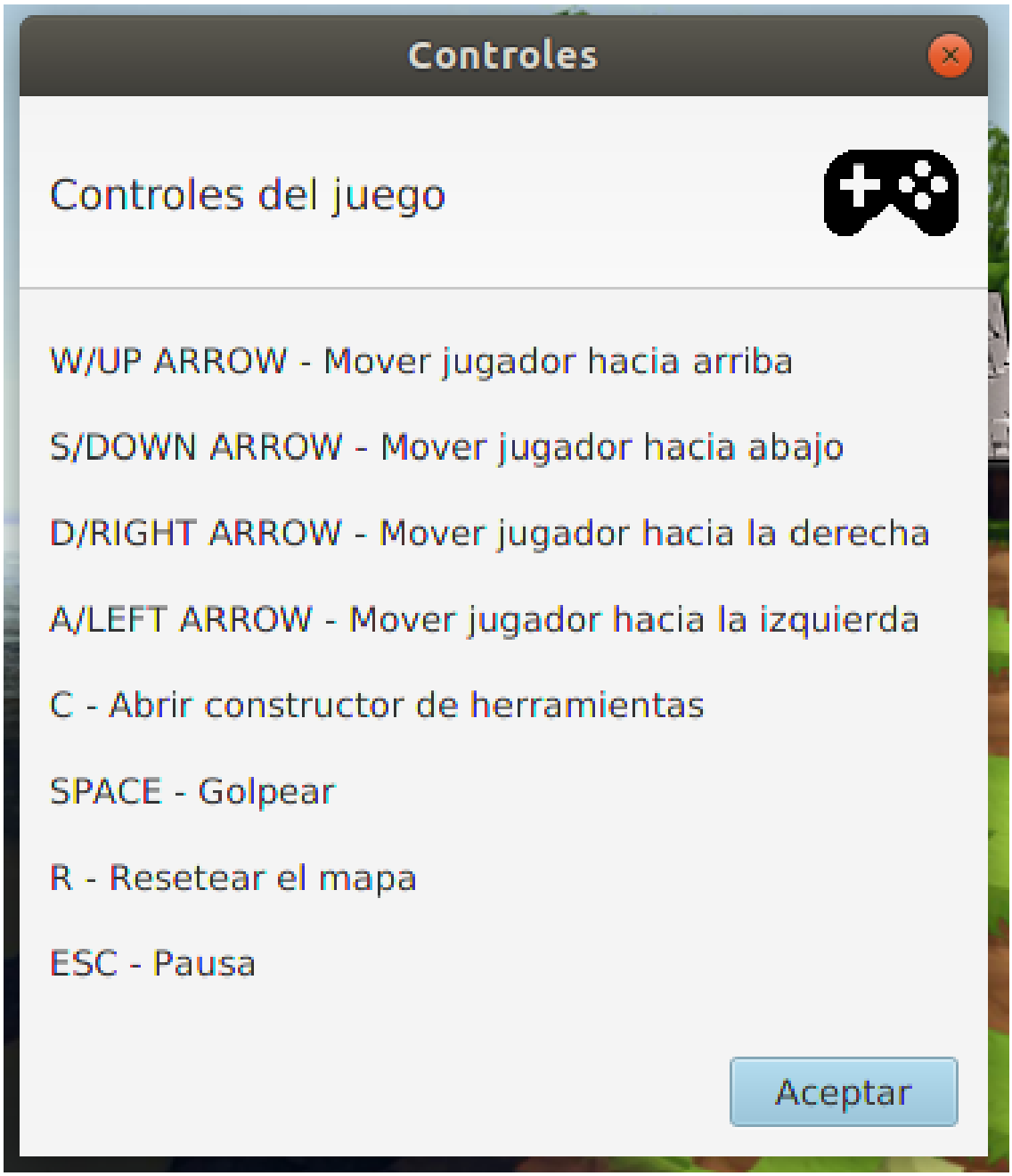

Figura 21: Ventana ilustrando los controles del juego.

Figura 22: Juego en ejecución.

Figura 23: Construccion de un hacha.

Figura 24: Construccion de un pico.