**FIUBA - 75.07**

**Algoritmos y programación III**

*Trabajo práctico 2: AlgoCraft*

1er cuatrimestre, 2015

(trabajo grupal)

Alumnos:

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Padrón** | **PuertoEstelarTerranMail** |
| Pablo Murck |  |  |
| Emiliano Sorbello |  |  |
| Lucas Soro | 95665 | lugusor@gmail.com |
|  |  |  |

***Fecha de entrega final***: Miércoles 24/06/2015 - Jueves 25/06/2015

***Tutor***: Devoto

***Nota Final***: 3\*pi ?

Informe

Supuestos

* Mientras un edificio se construye, no puede ejecutar sus acciones.
* Los recursos se sustraen del jugador apenas comienza la construcción o entrenamiento.
* Los edificios recolectores de recursos se construyen solo sobre el recurso.
* Si un edificio o unidad es destruido, los recursos no se devuelven.
* Los recursos nunca se acaban.
* En cada base debe haber solo un volcán y 2 o más cristales.
* Un jugador gana si destruye todos los edificios incrementadores de población del otro jugador

Modelo de dominio

**Clases superiores**: Son las de mayor interés para comunicar al modelo con el exterior.

* Juego: Contiene ambos jugadores, se encarga de manejar el sistema de turnos.
* Jugador: Modela a un jugador con nombre, color y raza. Tiene conocimiento parcial del mapa y conoce sus recursos, unidades y edificios.
* MapaReal: Modela el mapa real de juego, Se genera al instanciarlo. Almacena e informa sobre los terrenos y ocupantes de cada una de sus celdas.

**Clases genéricas**: Reúnen características comunes de las entidades concretas, que quedarán definidas de acuerdo a los componentes y atributos que se le asignen.

* Recurso: Ocupante que define que puede ser reemplazado por otros ocupantes que puedan ocupar al recurso.
* ObjetoVivo: Un ocupante que tiene vida, estados y puede pertenecer a algún jugador
* Unidad: ObjetoVivo con posibilidad de movimiento. Puede atacar, hacer magia o transportar según los componentes.
* Edificio: ObjetoVivo que modela un edificio. No puede moverse.

**Componentes:** Se asignan a las clases genéricas, y cada uno define parte de su comportamiento. Pueden tener atributos asociados.

* IVida: Se encarga de los aspectos de la vida (Máximo, actual, escudo, regenerar, recibir daño). Usa AtributosVida
* IAtaque: Para operaciones de ataque (Y validación). Usa AtributosAtaque para datos de rango y daño.
* Costo: Usa AtributosCosto e informa sobre costos y tiempos de construcción.
* IMagia: Se encarga de la energía y las magias de las unidades. Usa AtributosMagia.
* Movimiento: Valida y efectúa el movimiento de una unidad. Usa AtributosMovimiento.
* Transporte: Se ocupa del almacenamiento de unidades (Solo para unidades que puedan almacenar). Usa AtributosTransporte).
* Estado: Una acción que se ejecute a lo largo de varios turnos. Cada edificio tiene alguno, de acuerdo a su función.

**Atributos:** Almacenan características particulares de alguna entidad distinguiéndola de otras. Cada entidad concreta tiene sus atributos (Ejemplo: AtributosMarine define al Marine). Además los componentes tienen los suyos, varios enumerados en el ítem anterior.

Cada jugador tiene su instancia de los atributos que le corresponden según la raza. Si se aplica alguna tecnología, la actualización debe realizarse en estos atributos y el cambio se reflejará inmediatamente en todas las unidades o edificios de ese jugador.

Diagramas de clases

*[Varios diagramas de clases, mostrando la relación estática entre las clases, pueden agregar todo el texto necesario para aclarar y explicar su diseño, recuerden que la idea de todo el documento es que quede documentado y entendible como está hecho el TP]*

Diagramas de secuencia

*[Varios diagramas de secuencia, mostrando la relación dinámica entre las clases planteando una gran cantidad de escenarios que contemplen las situaciones del trabajo práctico]*

Diagrama de paquetes

*[incluir un diagrama de paquetes para mostrar el acoplamiento de su trabajo ]*

Diagramas de estado

*[Incluir diagramas de estados, mostrando tanto los estados como las distintas transiciones de los mismos para varias entidades del trabajo práctico ]*

Detalles de implementación

*[Deben* ***detallar/explicar*** *qué estrategias utilizaron para resolver los puntos más conflictivos del trabajo práctico. (Incluida la persistencia) ]*

Excepciones

*[Explicar las excepciones creadas, con qué fin fueron creadas y cómo y dónde se las atrapa explicando qué acciones se toman al respecto una vez capturadas.]*

**AccionesPorTurnoInsuficientes**: Se lanza cuando se desea ejecutar acciones que requieran de mas “acciones por turno”, como por ejemplo movimientos y ataques. Se atrapa en la interfaz para dar mas información.

**CapacidadAlmacenamientoInsuficiente:** Se lanza cuando se quiere almacenar a una unidad en un transporte pero el costo de transportar excede la capacidad del transporte. Se atrapa en la interfaz para dar mas información.

**CeldaNoVisible: ?**

**EnergiaInsuficiente:** Se lanza cuando no hay suficiente energía para ejecutar una magia. Se atrapa en la interfaz para dar mas información.

**EstadoFinalizado:** Se lanza cuando un Estado informa a su portador que ha finalizado. Se atrapa en el ObjetoVivo donde se remueve de su lista de Estados y se le llama el método “desactivar”.

**FueraDelRangoPermitido:** Se lanza cuando se desea ejecutar acciones que estén limitadas por un rango y no se cumpla con el mismo. Se atrapa en la interfaz para dar mas información.

**GasVespenoInsuficiente / MineralInsuficiente / SuministroInsuficiente:** Se lanza cuando el Jugador no tiene suficiente recurso para llevar a cabo una “compra” de edificio o unidad. Se atrapa en la interfaz para dar mas información.

**MovimientoInvalido:** Se lanza cuando una Unidad trata de moverse de forma errónea. Se atrapa en la interfaz para dar mas información.

**NoEsUnAliado:** Se lanza cuando se desea ejecutar una acción que solamente debe influir sobre unidades aliadas, como por ejemplo la magia “Alucinación”. Se atrapa en la interfaz para dar mas información.

**NoEsUnEnemigo:** Se lanza cuando se desea ejecutar una acción que solamente debe influir sobre unidades enemigas, como por ejemplo un ataque común. Se atrapa en la interfaz para dar mas información.

**NombreInvalido:** ?

**OrdenConstruccionViolado:** Se lanza cuando se trata de construir un edificio que necesita tener otros edificios construidos con anterioridad. Se atrapa en la interfaz para dar mas información.

**PosicionOcupada:** Se lanza cuando un Ocupante quiere posicionarse sobre otro Ocupante que no permite ser reemplazado, como por ejemplo ObjetoVivo. Se atrapa en la interfaz para dar mas información.

**RecursoAusente:** Se lanza cuando un Ocupante, que necesita posicionarse sobre un recurso, trata de posicionarse en una posición sin recurso, como por ejemplo un Edificio recolector. Se atrapa en la interfaz para dar mas información.

**RecursoPresente:** Se lanza cuando un Ocupante quiere posicionarse sobre un Recurso y el mismo no lo tiene permitido, como por ejemplo cualquier Unidad. Se atrapa en la interfaz para dar mas información.

**TerrenoInadecuado:** Se lanza cuando un Ocupante desea posicionarse sobre un Terreno que no tiene permitido, como por ejemplo Edificios en Terreno Espacio. Se atrapa en la interfaz para dar mas información.

**UnidadNoEsAlmacenable:** Se lanza cuando un transporte trata de almacenar a una Unidad que no puede ser almacenada. Se atrapa en la interfaz para dar mas información.

**UnidadYaAlmacenada:** Se lanza cuando un transporte trata de almacenar a una Unidad que ya esta almacenada en ese mismo transporte. Se atrapa en la interfaz para dar mas información.

**VidaEnCero:** Se lanza cuando la vida de un ObjetoVivo llega a cero. Lanzado por el componente de Vida, el ObjetoVivo se encarga de atraparlo y cuando sucede debe proceder a “auto-destruirse” llamando al método “destruir”.

**Checklist de corrección**

Esta sección es para uso exclusivo de los docentes, por favor no modificar.

Carpeta

**Generalidades**

* ¿Son correctos los supuestos y extensiones?
* ¿Es prolija la presentación? (hojas del mismo tamaño, numeradas y con tipografía uniforme)

**Modelo**

* ¿Está completo?¿Contempla la totalidad del problema?
* ¿Respeta encapsulamiento?
* ¿Hace un buen uso de excepciones?
* ¿Utiliza polimorfismo en las situaciones esperadas?

Diagramas

**Diagrama de clases**

* ¿Está completo?
* ¿Está bien utilizada la notación?

**Diagramas de secuencia**

* ¿Está completo?
* ¿Es consistente con el diagrama de clases?
* ¿Está bien utilizada la notación?

**Diagrama de estados**

* ¿Está completo?
* ¿Está bien utilizada la notación?

**Código**

**Generalidades**

* ¿Respeta estándares de codificación?
* ¿Está correctamente documentado?