Sergio Miguel García Jiménez

Código

- Crear parte del menú principal en modo consola.
- Colaborar en el diseño de la clase Player.
- Desarrollar gran parte de la lógica de cargar, guardar y borrar partida (SaveLoadManager).
- Contribuir a que modelo trabaje exclusivamente con JSON. Esto incluía la concepción del diseño de reports así como su implementación en las distintas clases, tarea inicialmente desarrollada por Leo, Virginia y yo, y continuada por el resto de compañeros.
- Desarrollé junto con Leo toda la funcionalidad de red, incluyendo elementos de la GUI que precisa, su extensión a todos los modos de juego y la lógica detrás de las conexiones.
- Me encargué junto a otros del desarrollo de la estructura de la GUI; concretamente en lo referente al tablero, diálogos de cargar, crear, borrar partida y la interacción de la GUI con el modelo, así como una participación activa en su pulido y debug por las posteriores características introducidas por el modelo.
- Realicé diversas labores de optimización para solventar un pobre rendimiento de la GUI a la hora de mostrar el tablero, actualizándose ahora en tiempo casi instantáneo y constante.
- El debug de las características que estaba desarrollando conllevó a debugear otras clases referentes al modelo para perfeccionarlo.
- Encargado de realizar algunos merge y etiquetado.
- Proponer y solventar algunas de las issues.

Documentación

- He realizado los primeros diagramas de clases y UML para ofrecer unos primeros prototipos para poder determinar por qué estilo de diseño queríamos adoptar.
- Concebí el diseño de la refactorización de las historias de usuario para adaptarlas al formato de la asignatura, tarea después completada por compañeros.
- Creación de los primeros artículos y la estructura preliminar de la Wiki en los primeros sprints.
- Trabajo UML preliminar con Modelio.
- Pugnar por pasar de trabajar en el software Modelio a PlantUML para trabajar con mayor comodidad.

Leonardo Macías Sánchez

Código

- Concepción, diseño y desarrollo de la clase SaveLoadManager, que en sprints futuros sería refactorizada por Sergio para adaptarla a las nuevas necesidades.
- Desarrollo de las clases Player, Cube y Color, sobre todo durante la etapa inicial del proyecto.
- Refactorización de la clase Board para que contuviera la lógica del juego.
- Desarrollo y adaptación del juego a la red, incluyendo una extensión de la interfaz gráfica que facilitase el funcionamiento al usuario. Esta tarea fue desarrollada conjuntamente con Sergio.
- Diseño y desarrollo del paquete replay, que abarca la interfaz Replayable y las clases Replay y GameState. Además de implementar toda la lógica, también me he dedicado a su correcta visualización, tanto en consola como en GUI.
- Principal impulsor y defensor del uso de JSONObjects en el proyecto. Ideé la interfaz Reportable y me encargué del diseño de los JSON y de su posterior implementación junto con Sergio y Virginia.
- Diseño e implementación de los métodos toString() del modelo.
- Colaboración en el diseño de los GameBuilders junto con Juan Diego y Daniel.
- Refactorización, junto a Juan Diego, de la clase Game para poder introducir las modalidades de GameTeams y GameClassic.
- Ayudar a Virginia y Mar con problemas que surgieron al crear JUnits.
- Creación del ranking junto con Mar.
- Refactorización completa de todas las clases de la GUI, rediseñandola al completo (a excepción del tablero) y creando nuestros propios componentes visuales, los RolitComponents.
- Propuse la creación de la clase TurnManager.
- Labores de debug.

Documentación

- Diseño de los reports.
- Redactar el funcionamiento de los Builders.
- Desarrollo de diagramas de secuencia.
- Redactar algunos de los sprint reviews y retrospectives.

- Colaboración en el desarrollo de la wiki.
- Búsqueda de una alternativa a Modelio para el desarrollo de diagramas de clases.

Daniel González Arbelo

Código

- Desarrollo en las etapas más tempranas del proyecto de las clases Game y Board, implementando los métodos encargados de la ejecución y actualización del juego en los turnos.
- Retoques y debug en las primeras etapas de la clase SaveLoadManager para el funcionamiento de cargado y guardado de partida.
- Creación de las clases herederas de Command, junto con Mar, aplicándose el patrón comando.
- Desarrollo de la lectura de datos de entrada para la creación del juego por consola junto con Leo y Mar.
- Creación de los ficheros para las formas de los tableros y las imágenes de las celdas en GUI.
- Creación de las primeras versiones funcionales de la GUI (ventanas de creación de juego, tablero, ranking, etc.), desarrolladas todas conjuntamente con Sergio.
- Creación, desarrollo y debug de las inteligencias artificiales y sus estrategias (y desarrollo de algunas alternativas para su funcionamiento con Juan Diego, que fueron finalmente deprecadas).
- Resolución de algún fallo en el funcionamiento de la red.
- Refactorización, junto con Juan Diego, del modelo para que trabajase en su propia hebra, y de la vista por consola.
- Refactorización del manejo de turnos (con la creación de TurnManager, idea elaborada con Leo) y de la clase Player para adaptarlos al funcionamiento con la hebra del modelo y las inteligencias artificiales.
- Creación de la lógica encargada del funcionamiento del tutorial.
- Tareas de debug, manejo de excepciones y resolución de issues.

Documentación

- Creación de los primeros diagramas de secuencia para fijar el formato.
- Establecimiento de un procedimiento para el desarrollo de los diagramas de secuencia usando PlantUML y IntelliJ.
- Documentación acerca del funcionamiento de la GUI.
- Participación en los sprint reviews y retrospectives.

María del Mar Ramiro Ortega

Código

- Desarrollo de un primer modelo de ranking, mostrado al final de la partida y ordenado por puntos.
- Creación de las clases herederas de Command, junto con Daniel, aplicándose el patrón comando.
- Diseño, junto con Daniel, y desarrollo de los distintos tamaños y formas del tablero, tanto a nivel de lógica del juego como a nivel de su carga y guardado por medio de la clase SaveLoadManager.
- Realización de una versión incial de los JUnits con mi compañera Virginia.
- Actualización de los JUnits en todas las refactorizaciones siguientes junto con Sergio.
- Añadido de nuevos tests a los JUnits de las distintas historias de usuario que íbamos implementando.
- Creación del ranking de GUI junto con Leo y mejora del ranking de console.

Documentación

- Creación de diagramas de secuencia.
- Desarrollo de una refactorización de las Historias de Usuario junto con Virginia, que fueron actualizadas y subidas a la wiki.
- Documento acerca de los diversos cambios que hicimos durante los distintos sprints junto con Leo, así como su actualización en la wiki.
- Redactar algún sprint review y retrospective, así como la actualización de la wiki respecto a varios sprints reviews y retrospectives que realizamos en PDF.
- Participación en los sprint reviews y retrospectives, junto con el resto de mis compañeros.
- Javadoc del paquete Builder y de la vista de consola.
- Documentación y UML sobre la HU *Como usuario quiero poder jugar a Rolit siguiendo un conjunto mínimo de normas* .

Juan Diego Barrado Daganzo

Código

- Desarrollo de la lógica interna de Game en las fases iniciales del proyecto, creación de la clase Player e integración de la misma sobre el modelo.
- Desarrollo del mecanismo de cambio de puntuaciones a través de los Cube, que permitía encapsular mucho mejor la puntuación de cada jugador.
- Diseño de las responsabilidades conceptuales de los integrantes del modelo, es decir, asignar las capacidades y dependencias del Board, Game, Player... tanto a nivel de código como a nivel de concepto, lo que unificó lo que se esperaba de cada clase e hizo más fácil la integración de funcionalidades posteriores.
- Creación y diseño de la clase builder y de la refactorización del juego a través de factorías.
- Encapsulación del modelo en la clase abstracta de Game, definición de las responsabilidades de cada una de las clases hijas (GameClassic y GameTeams) e implementación de las mismas.
- Adaptación del Modelo-Vista-Controlador a través de la generación de un hilo propio para el modelo que permitía una mejor integración de los jugadores automáticos. Creación de la cola de prioridad de Cube para poder utilizar el modo red y los jugadores automáticos de forma simultánea.
- Creación de la vista de consola y generación de la interfaz *ConsoleWindow* para mantener un criterio uniforme a modo de "componentes" tal y como se hace en Swing, lo que simplificó la independencia del modelo y la vista por su similitud con las ventanas de Swing.
- Depurado de la interfaz de RolitObserver y de los métodos de notificación a observadores.
- Depurado de la red para integrar los jugadores automáticos.

Documentación

- Rediseñado de los reports iniciales.
- Creación del formato de los documentos de desarrollo (los utilizados durante los sprint por el equipo de desarrollo, no los que se incluyen en la entrega) para informar a todos los participantes de la implementación final que se dió al diseño planeado de forma general.

Virginia Chacón Pérez

Código

-

Documentación

-