DOCUMENTACIÓN: JUEGO DE LA OCA.

AUTORES: Enrique Sánchez Vicente y Álvaro Cañada Lorenzo.

REPOSITORIO: https://github.com/EnriqueSanVic/OcaGame

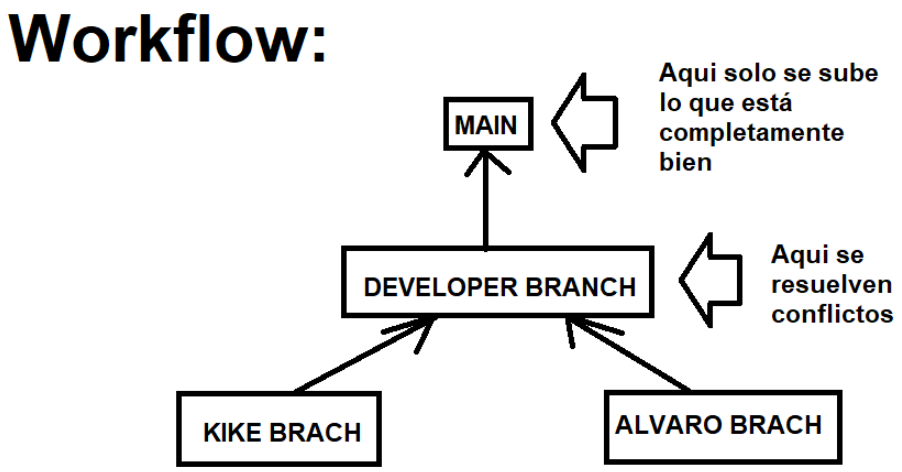
ORGANIZACIÓN DEL TRABAJO:

La forma de trabajo seleccionada será un modelo de desarrollo iterativo, ya que nos permitirá pivotar entre las diferentes implementaciones del sistema sin tener que implementar funcionalidades completas antes de continuar con las siguientes. Esto puede aportar flexibilidad a nuestro proceso de desarrollo, además de capacidad de observación en conjunto de todas las funcionalidades de nuestro proyecto. Pero tiene un contra, que requiere mucha organización entre los desarrolladores.

ORGANIZACIÓN DE LOS TRABAJADORES:

Los desarrolladores compartirán sus archivos a tiempo real, a través de una carpeta de Google Drive, donde podrán crear y modificar elementos que se necesiten durante el desarrollo. También se hará uso de git y github a la hora de codificar y modificar el software propiamente dicho, dejando constancia en los commit de cada uno de los cambios, trabajando cada desarrollador en una rama, y compartiendo el trabajo final en una rama master. Además, para ciertas tareas, se usará Skype para una mejor comunicación a tiempo real.

El flujo de las ramas de trabajo en el repositorio de git es el siguiente:



USO DEL MODELO (ITERATIVO):

Previamente a comenzar con las iteraciones, debemos de definir tanto los requisitos como los diagramas UML, del conjunto total del software, pero sin especificar cómo se implementará cada funcionalidad. Esto nos sirve para estructurar todo el sistema y tener a disposición una ruta de trabajo.

METODOLOGÍA DE LA ITERACIÓN:

Cada iteración se enfocará en implementar total o parcialmente, una funcionalidad o requisito que se dejará reflejado en la hoja de ruta de iteraciones. En esta indicaremos: el punto de partida del estado en el que se encuentra el desarrollo de la funcionalidad y el hito a conseguir con la iteración. Cada iteración consiste en un modelo en cascada con retroalimentación con las siguientes fases:

· Fase de **investigación preliminar**. Se explorarán diferentes soluciones para la funcionalidad a desarrollar.

· **Análisis**. Se elegirá la solución más conveniente. Se especifican los requisitos de la funcionalidad.

· **Diseño**. Especificación de diseño de la funcionalidad. UML. Aportación de pseudocódigo si se requiriera y una breve explicación de la solución.

· **Codificación**. (Comentar método para Javadoc).

· **Pruebas** cuando se requieran.

Al ser un método de cascada con retroalimentación, si en cualquiera de las fases del desarrollo necesitáramos volver a alguna anterior debido a algún motivo, dejándolo reflejado en la hoja de ruta.

CLASES (Divididas por paquetes):

- **Controladores:** ControladorInicio, ControladorJuego, ControladorJuegoModo1, ControladorJuegoModo2, ControladorMenu.
- **DatosEstaticos:** Constantes, TextosJuego.
- **Hilos:** Hilo, RegistradorHilos.
- **Logicas:** LogicaJuego, LogicaJuegoModo1, LogicaJuegoModo2.
- **Modelos:** CasillaLogica, DatosPartidaModo1, DirectivasEvaluacion, GestorGuardado.
- **ReproductorSonido:** ManejadorSonidos, ReproductorContinuo, ReproductorUnaVez.
- **Utilidades:** UtilidadesGraficas, VentanaConCorreccion.
- **Vistas:** DadoGrafico, PanelFondo, PanelFondoInicio, PanelNombres, PanelPenalizaciones, VistaAutores, VistaInstrucciones, VistaJuego, VistaJuegoModo1, VistaJuegoModo2.

- **Vistas.TableroSystem:** CasillaGrafica, Ficha, ManejadorFicha, NotificableTablero, PunteroGrafico, Tablero.
- **ocagame:** OcaGame(main).

Flujo de Ejecución de un ciclo de tirada:

El modelo vista controlador del juego está construido en base a este diagrama de eventos y acciones entre las entidades del programa.