[75.07 / 95.02]

Algoritmos y programación III

Trabajo práctico 2: GPS Challenge

Nombre	Padrón	Mail
Joaquin Pandolfi	108215	jpandolfi@fi.uba.ar
Facundo Aguirre Argerich	107539	faguirrea@fi.uba.ar
Eliana Harriet	107205	eharriet@fi.uba.ar
Joaquin Rivero	106032	jorivero@fi.uba.ar
Valentin Schneider	107964	vschneider@fi.uba.ar

Tutor: Joaquin Gomez

Nota Final:

Índice

Introduccion

El presente documento reúne la documentación correspondiente a la presentación de la solución del trabajo práctico número dos de la materia, el cual consiste en el desarrollo e implementación de un juego de estrategia por turnos llamado GPS challenge utilizando conceptos del paradigma orientado a objetos vistos en la materia.

El objetivo del jugador será guiar un vehículo a la meta en la menor cantidad de movimientos posibles, evitando obstáculos y recogiendo sorpresas aleatorias.

Supuestos

Para las mecánicas del juego, hubo varias cuestiones que no estaban aclaradas en la consigna. Al ser estos detalles menores de la dinámica del juego que no cambiaban según el modelo fueron decididos acorde a lo que los miembros del grupo fuimos considerando correcto de forma conjunta. A continuación, los principales supuestos que tomamos:

- Luego de pasar por una sorpresa o un obstáculo, este permanece en la calle y no se elimina, de modo que el jugador puede volver a pasar por el mismo.
- El tamaño del mapa es de 10 cuadras de ancho y de alto. Si bien nuestro modelo acepta distintos tamaños y podría recibirlo por parámetro, optamos por mantener uno fijo para mantener constancia a la hora de comparar la cantidad de movimientos que realizó cada jugador para presentarlos en el ranking.
- No hay límite para la cantidad de movimientos posibles.
- La cantidad de obstáculos y sorpresas es aleatorio con igual probabilidad de aparición.
- En el modo de juego multijugador, si un jugador llega a destino termina la partida, los otros jugadores pierden y se anota el puntaje del jugador que ganó en el ranking.
- En el modo de juego multijugador, el campo de visión se enfoca en el jugador de turno.

Diagramas de clase

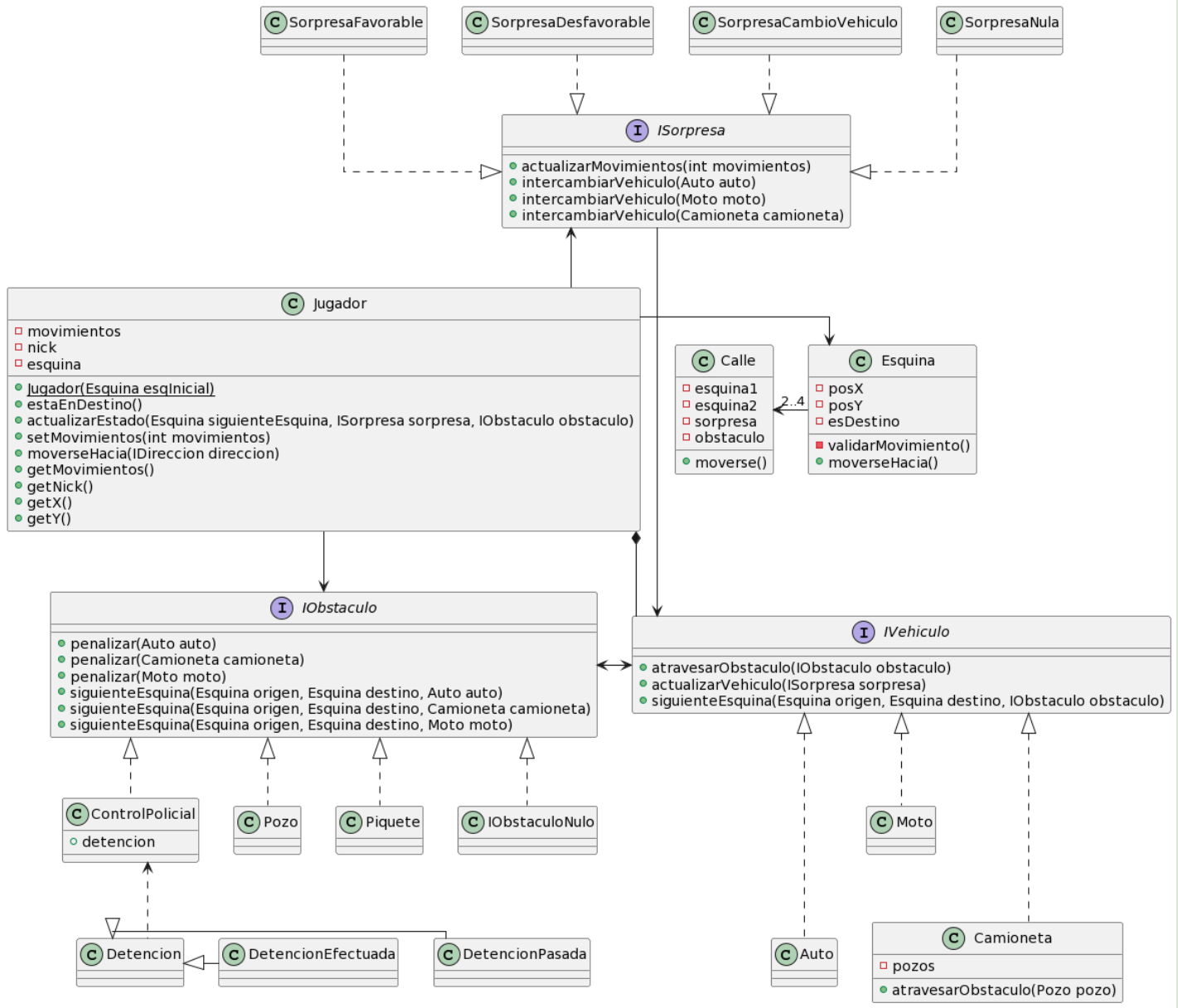


Diagrama de clases de sección Modelo (Diagrama 1.1)

Elementos que conforman la lógica de comportamiento de los objetos principales del juego. Se observan las relaciones entre los objetos, interfaces y herencias.

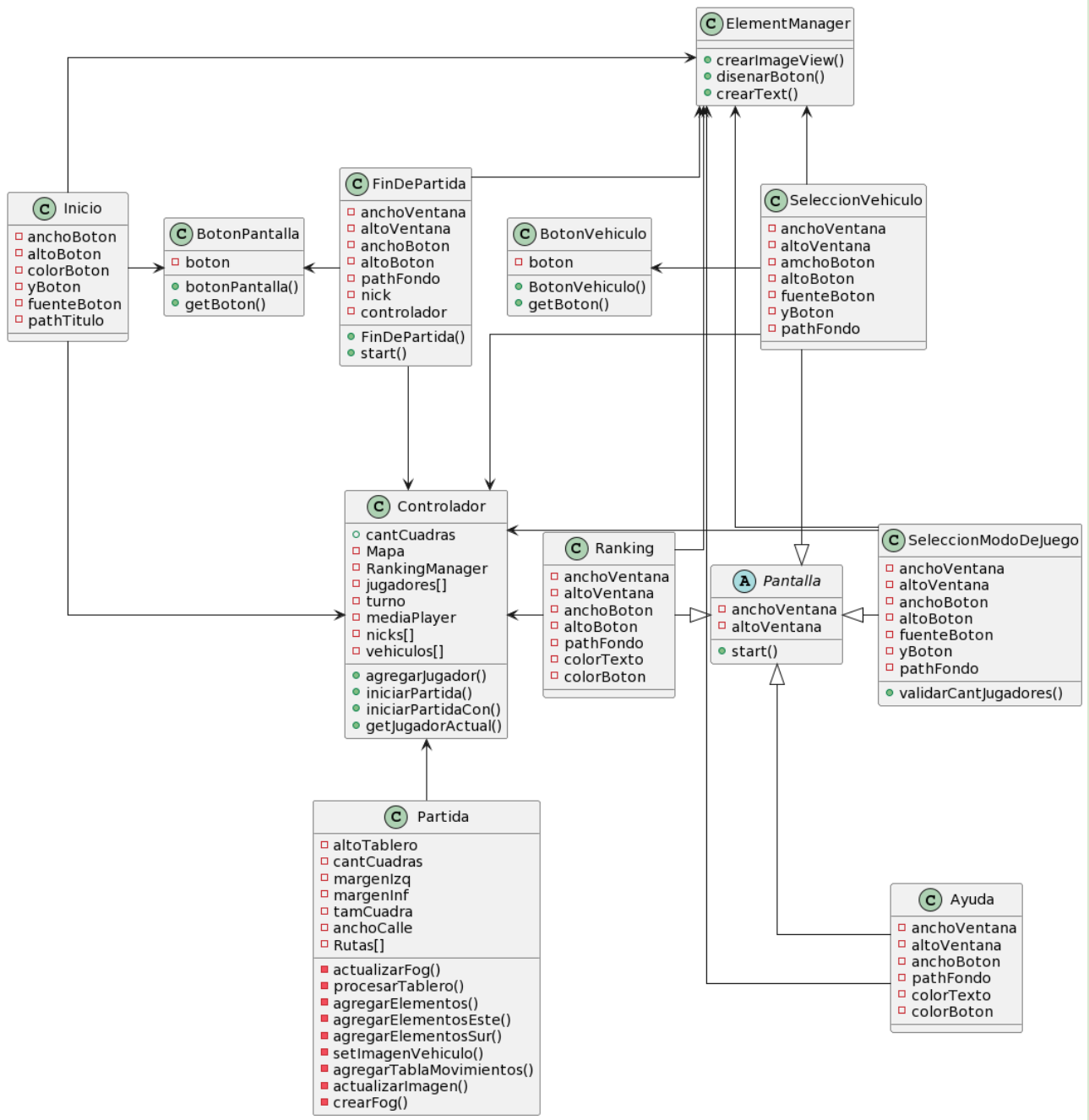
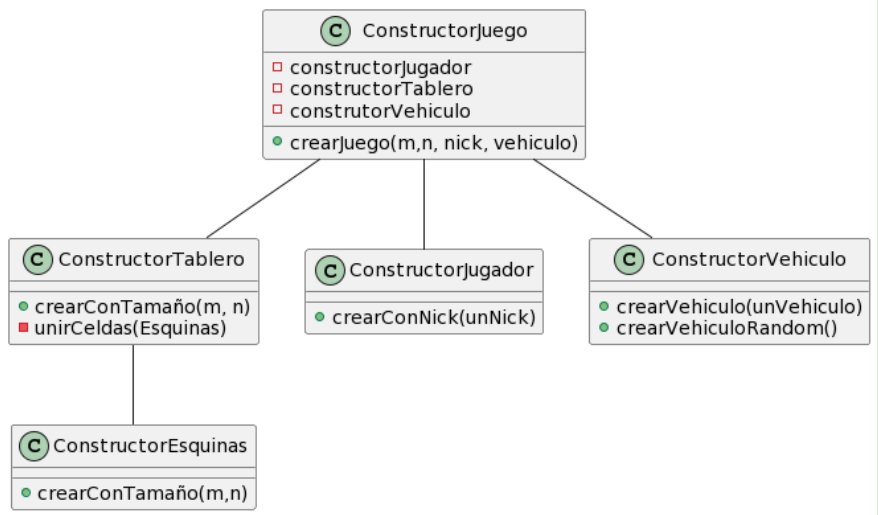


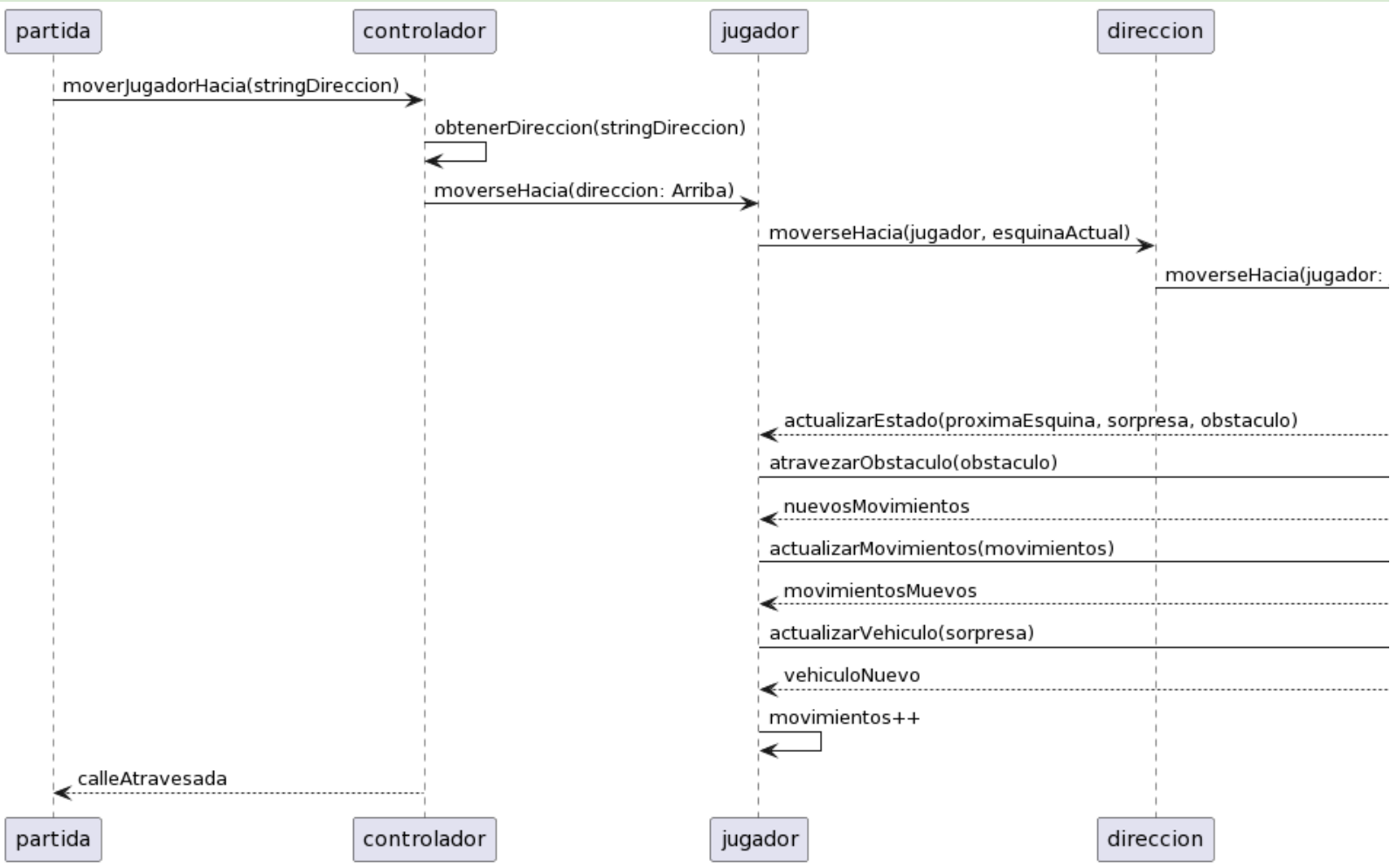
Diagrama de Clases sección Vista-Controlador (diagrama 1.2)

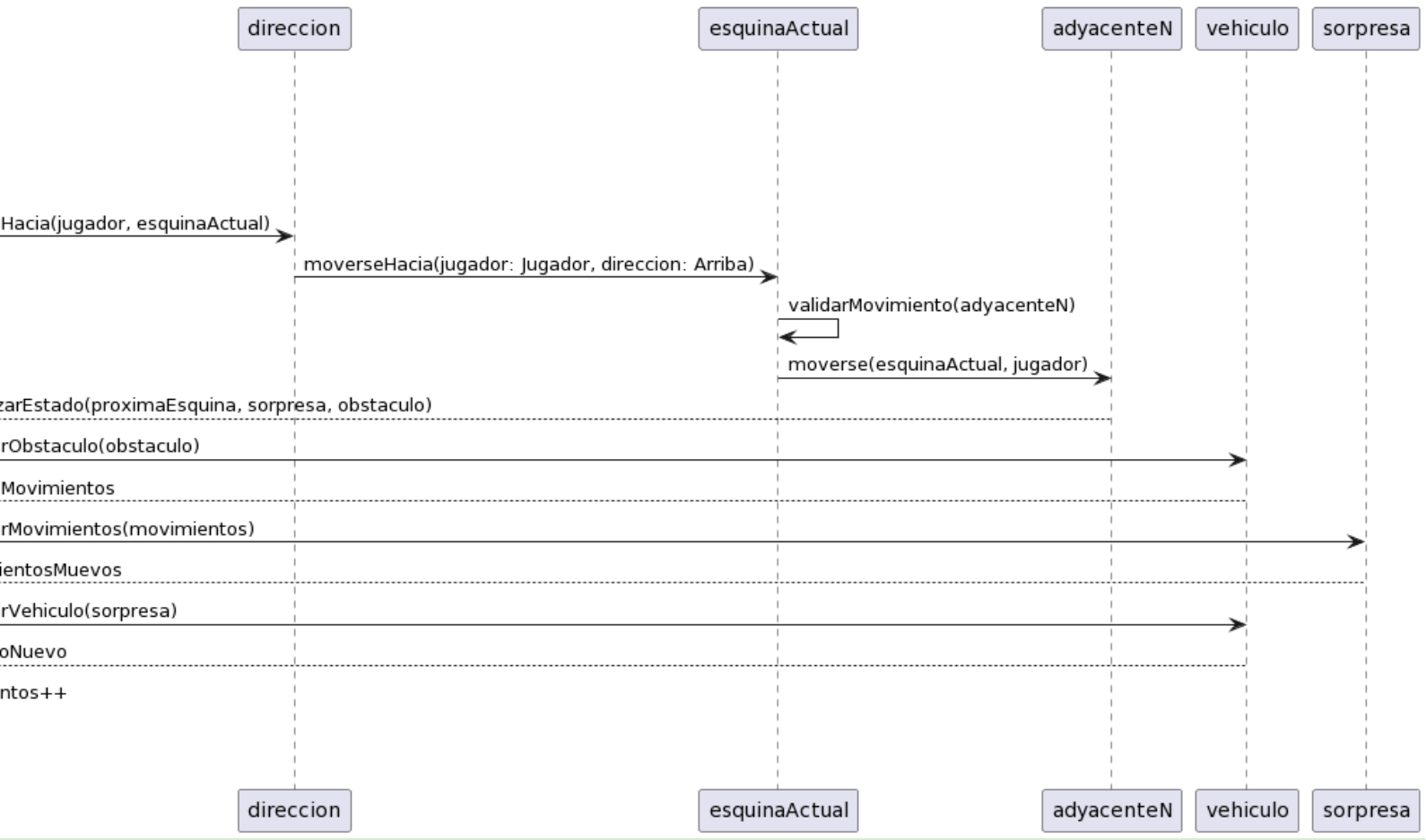
Se observan las relaciones de los objetos de la vista entre ellos y con el controlador, las distintas pantallas



Relación entre los constructores (diagrama 1.3)

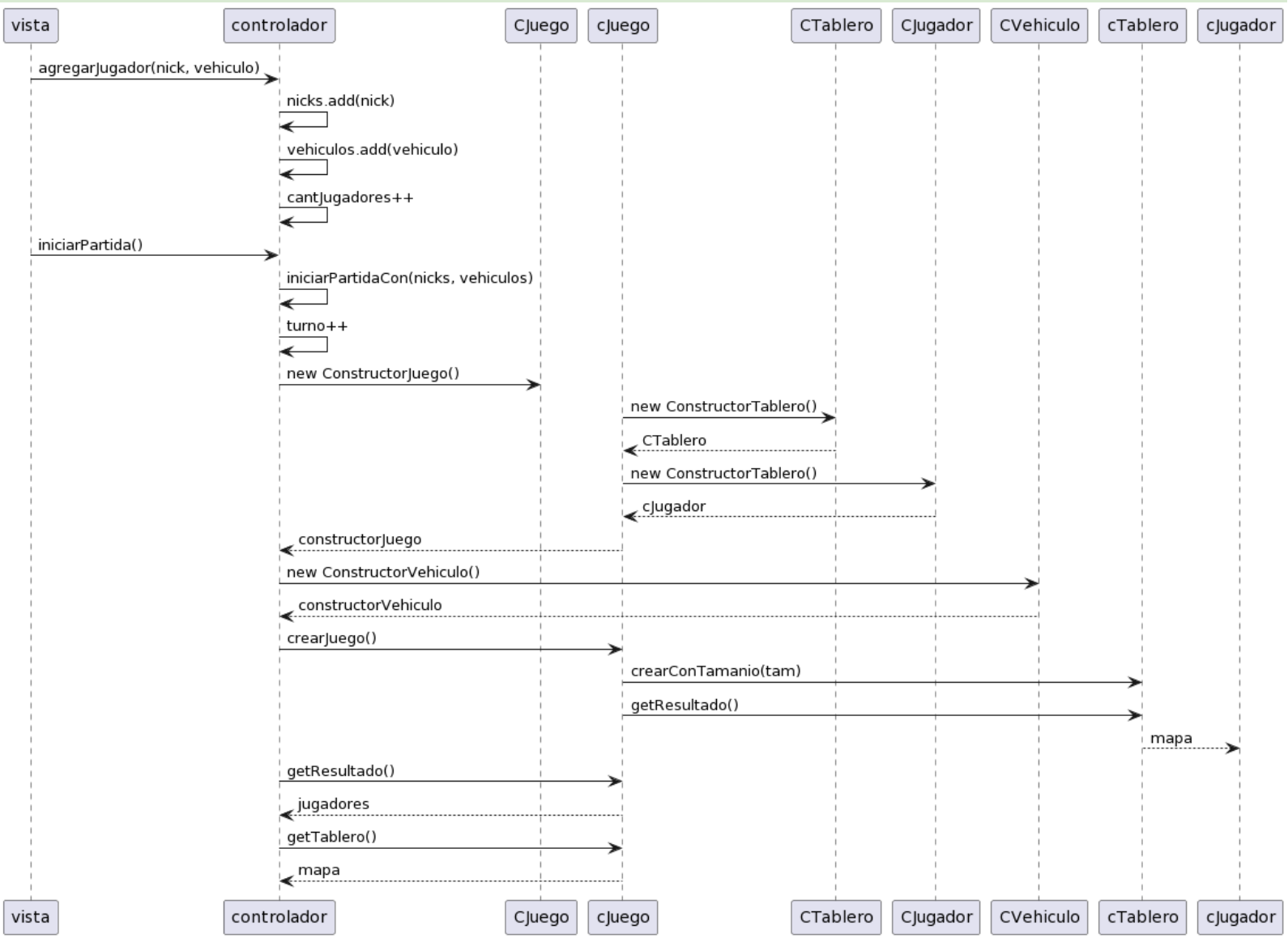
Diagramas de secuencia





[↑ página anterior] Se mueve el personaje en una dirección (Diagrama 2.1)
Se dividió el diagrama en dos partes, la izquierda arriba, y la derecha, abajo.

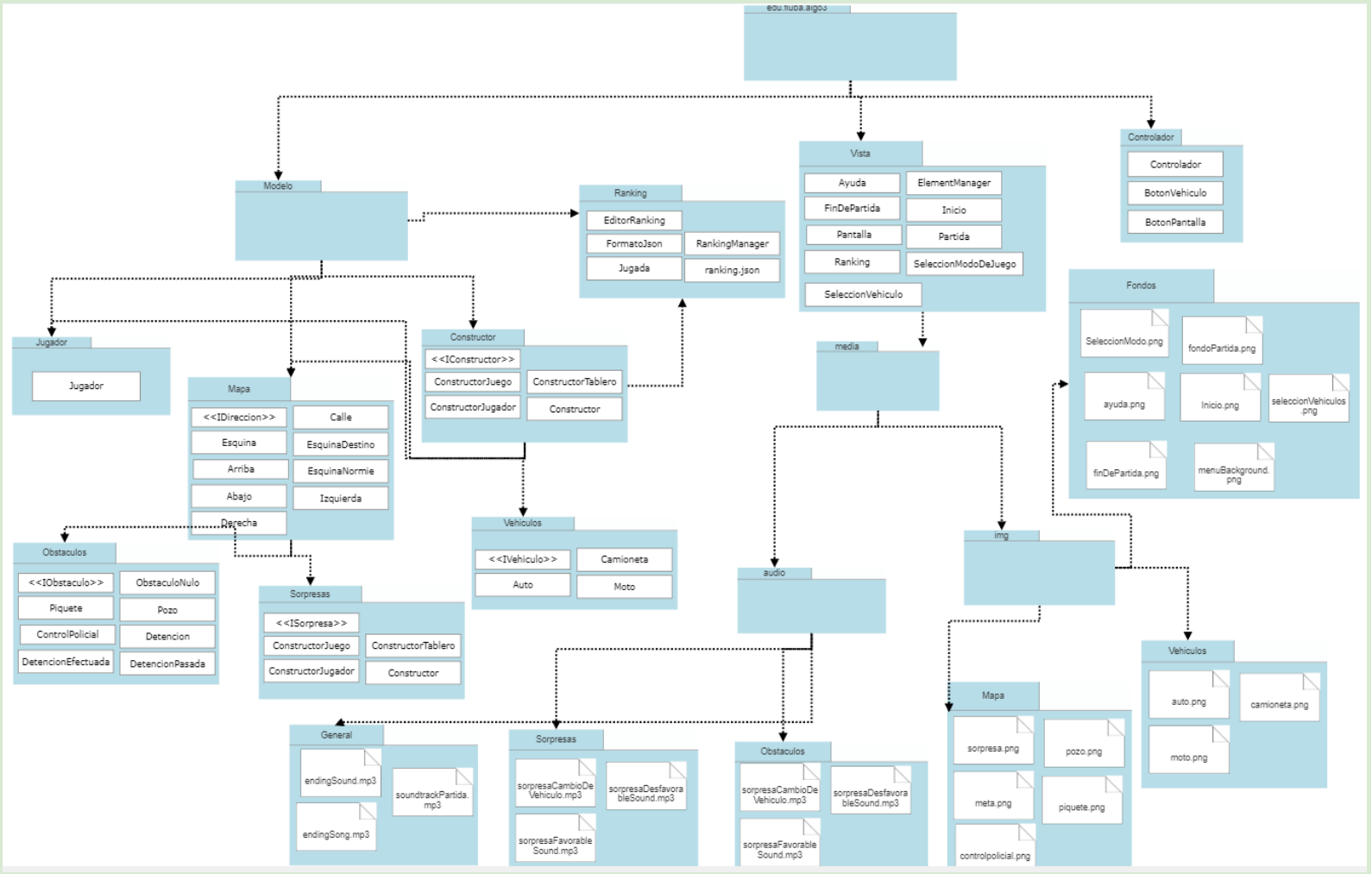
El comportamiento del jugador cuando se le envía el mensaje “moverseHacia” es el de enviarle un mensaje a la esquina correspondiente para que lo cambie de lugar, seguido de un mensaje a los obstáculos y sorpresas encontradas en el trayecto para que cada uno se ejecute.



Iniciación de partida (diagrama 2.2)

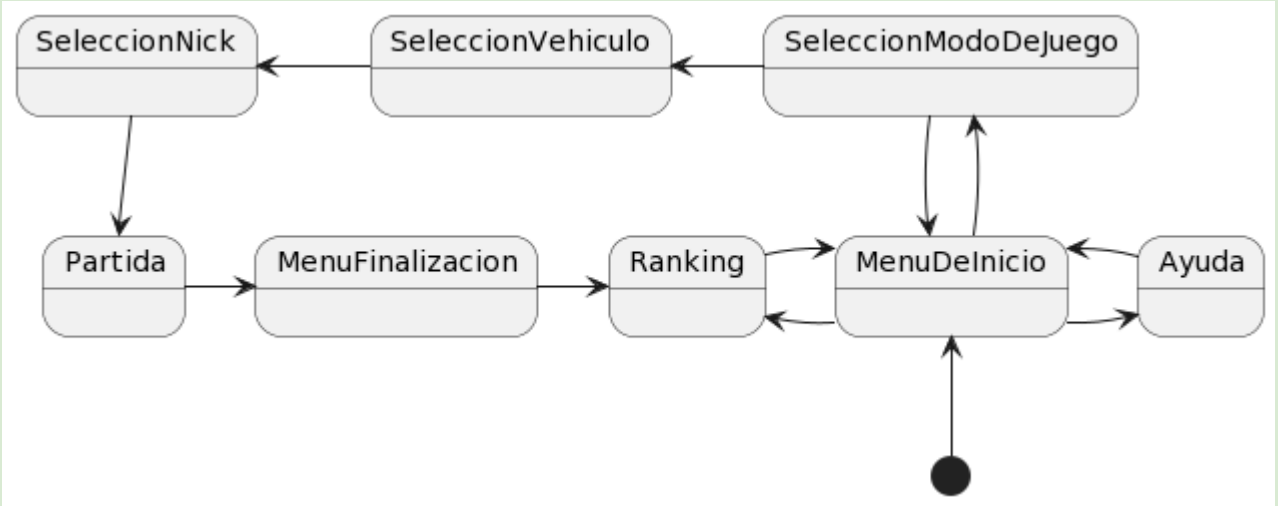
Secuencia desde una vez iniciado el programa hasta la iniciación de una partida. En los nombres de los objetos, el prefijo C significa “constructor”.
Se ve la obtención del nick y vehículos elegidos por el usuario, la creación de todos los objetos y relación de cada uno de ellos, para luego terminar con el mapa construido.

Diagramas de paquetes



Diagramas de estado

En el siguiente diagrama, se ve el flujo de estados del programa de las pantallas posibles del juego.



Pantallas del programa (diagrama 4.1)

Detalles de implementación

Modelo

En dos casos utilizamos herencia, en “Esquina” y “Detención”, ya que estas utilizan los mismos métodos y funcionan de manera similar, de esta forma evitamos repetir código, y posteriormente en Pantalla.

Utilizamos cuatro patrones de diseño: “Factory”, “Null Object”, “Double Dispatch” y “Facade”.
El primero utilizado para la construcción de los objetos, ya que necesitamos crear varios tipos de objetos distintos pertenecientes a una misma clase, por ejemplo, “Pozo”, “Control Policial”, y “Piquete”, los tres son obstáculos.

Utilizamos el patrón Null Object para evitar devolver “null” donde no haya un comportamiento asignado, y en su lugar, devolver un objeto que tenga el comportamiento necesario.

El patrón Double Dispatch fue necesario para solucionar de forma sencilla la relación entre los obstáculos y sorpresas, y los movimientos y vehículo del jugador.

Facade fue utilizado para el sistema de ranking, donde se utiliza una fachada (EditorRanking) la cual maneja los formatos de archivo de parseo en los formatos que se desee, esta fachada es utilizada para no acceder directamente en “RankingManager” las clases de formatos de archivos diversas que se podrían integrar en el proyecto, generando una interfaz cómoda para el uso de manejo de archivos y una complejidad menor al modelo.

Funcionamiento

En el diseño, nos centramos en el contraste de los colores, el fácil aprendizaje y memorización, ya que el objetivo es sencillo y utiliza pocos botones.

Para la visualización del juego, utilizamos el Patron “MVC”, por lo que separamos el modelo, de la vista y del controlador para tener tres divisiones claras e interconectadas en el código.

Dentro de este patrón, volvimos a utilizar Herencia ya que todas las pantallas comparten un comportamiento, y de esta manera, se facilitó la interacción entre vista y controlador mediante botones.

Excepciones

- TeFaltaCalleError
  - Se utiliza para controlar que cada jugador no pueda salir del mapa: cuando llega a un borde del mapa, no se podrá seguir avanzando en esa dirección.

- TeFaltaCarroError
  - Se utiliza para comprobar que no hayan errores en la creación de los personajes.