Implementación de un sistema de "test multi-jugador de cruce perceptual"

PROYECTO DE FIN DE CARRERA

Autor: David Gracia Larrodé

Director: Manuel González Bedia

Codirector: Francisco Serón Arbeloa

Ingeniería en Informática
Curso 2012-2013

Departamento de Informática e Ingeniería de Sistemas

Escuela de Ingeniería y Arquitectura

Universidad de Zaragoza

Junio de 2013

Implementación de un sistema de "test multi-jugador de cruce perceptual"

RESUMEN

El objetivo de este proyecto de fin de carrera es conseguir un marco de ejecución virtual para experimentos de cruce perceptual (paradigma para la realización de experimentos de cognición social, basado en los trabajos realizados en la universidad francesa de Compiègne en 2009 de Auvray, Lenay y Stewart[5]). Se desarrollará una aplicación virtual a la que se accederá desde computadoras situadas en habitaciones separadas. Cada participante modificará la posición de un cursor en la pantalla a lo largo de una línea mediante el movimiento transversal de su ratón, y recibirá un estímulo en el momento en que el cursor se cruce con un objeto durante su movimiento. Los sujetos podrán cruzarse con dos tipos de cosas: el cursor real del otro participante (persona) o con el cursor de un objeto artificial (móvil o fijo). Se les pide a los sujetos que sean capaces de identificar cuándo la interacción establecida es con el humano y cuando no.

Para dar solución al objetivo planteado se implementará una aplicación web, que trabajará sobre un servidor que soporte la tecnología WebSockets, que aporta un canal de intercambio de información bidireccional real entre el servidor y el cliente en tiempo real. El experimento se plantea sobre una red local para minimizar la latencia en la comunicación. Para la construcción del "front-end" se utilizará una combinación de HTML5, CSS y JavaScript.

Se construirá un *framewok* que permita realizar experimentos formados, por un lado, por dos contrincantes (persona vs persona o persona vs bot) y, por otro, por múltiples participantes, donde cada persona se enfrentará a otro participante (persona), a un objeto móvil y a un objeto fijo. En el primer tipo de experimento, al terminar cada prueba, el sujeto debe responder si ha interactuado con otra persona o con un objeto artificial. En el segundo tipo, durante la prueba, el sujeto podrá hacer click cuando crea que esta interactuando con un igual. El tipo de objeto móvil, en el segundo caso, podrá ser la sombra del oponente humano con cierto retardo en tiempo o a una distancia marcada.

Con esta plataforma se pretenden analizar todos los aspectos presentes en un intercambio comunicativo humano puesto que existen elementos que se refieren a las contingencias de la interacción que nos permiten identificar que una comunicación real se está produciendo. Pretendemos analizar las propiedades que permiten a los interlocutores construir patrones de interacción comunes que les hagan regular mutuamente sus acciones.

Para mi abuelo Agustin.
Por hacerlo posible.

Agradecimientos

A mi familia, mi novia y mis amigos por su apoyo y cariño. A los dos Carlos, Daniel y Gonzalo por ayudarme. A Manuel y Paco por confiar en mi. Gracias.

Índice general

Índice de figuras

Índice de tablas

Parte I

Memoria

Capítulo 1

Introducción

1.1. Objetivo y alcance del proyecto

En los últimos años, la investigación en Cognición Social se ha interesado por fenómenos de interacción humana que no pueden entenderse en función de capacidades individuales propias de los agentes que se encuentran en la interacción. Aparentemente, existen procesos autónomos de coordinación dinámica que sólo emergen cuando dos o más sujetos participan en una interacción en tiempo real. A partir de trabajos realizados en la universidad francesa de Compiègne en 2009 (Auvray, Lenay y Stewart, 2009[5]) se desarrolló un modelo mínimo para la realización de experimentos de cognición social donde se analizaba la interacción que dos sujetos mantenían en una comunicación ciega a través de un dispositivo electromecánico, estableciéndose la definición del paradigma de cruce perceptual. Estudios posteriores profundizaron más en detalle en el paradigma (Lenay et al., 2012[26]; Auvray y Rhode, 2012[4]) y otros analizaron modelos de simulación de esta tarea (Iizuka y Di Paolo de 2007[25], Di Paolo et al, 2008[11]) con el fin de explicar los diferentes patrones generados.

Con este proyecto de fin de carrera se pretende conseguir un marco de ejecución virtual para experimentos de cruce perceptual con el objetivo de poder realizar medidas que no son sencillas de implementar en la configuración original. Se desarrollará una aplicación virtual a la que se accederá desde computadoras situadas en habitaciones separadas. Cada participante modificará la posición de un cursor en la pantalla a lo largo de una línea mediante el movimiento transversal de su ratón, y recibirá un estímulo en el momento en que el cursor se cruce con un objeto durante su movimiento. Los sujetos podrán cruzarse con dos tipos de cosas: el cursor real del otro participante (persona) o con el cursor de un objeto artificial (móvil o fijo). En caso de lograr el escenario original, que no tenía ningún acceso visual al monitor e interactuaba con dos señales idénticas, los sujetos deben ser capaces de identificar cuándo la interacción establecida es con un humano y cuando con un objeto artificial.

Para dar solución al objetivo planteado se implementará una aplicación web, que trabajará sobre un servidor que soporte comunicación de datos en tiempo real. Dicho servidor debe dar soporte a la tecnología WebSockets, que aporta un canal de intercambio de información bidireccional y asíncrono entre el servidor y el cliente. El experimento se plantea sobre una red local para que la inevitable latencia por el paso de mensajes entre los terminales sea la mínima posible. Para la parte del "front-end" se utilizará una combinación de HTML5, CSS y JavaScript.

En primer lugar se construirá un *framewok* simple para realizar experimentos de cruce perceptual, los experimentos estarán formados únicamente por dos personas, una contra otra. En segundo lugar se implementará un *framewok* partiendo del anterior que permita experimentos persona contra persona y persona contra bot. En este marco seguimos teniendo un integrante por lado del experimento, pero se añade un nuevo tipo de participante, el bot. Se crearán diferentes tipos de bots, unos enviarán posiciones aleatorias, otros representarán la traza de movimientos de un experimento anterior, otros seguirán una política de movimiento según la posición del participante humano y finalmente otros repetirán los movimientos que este haciendo su oponente humano con un cierto retardo en tiempo. Al final de cada experimento cada persona deberá afirmar si ha interactuado con otra persona o con una máquina. En tercer lugar se añadirá funcionalidad al *framewok* anterior permitiendo realizar experimentos con más de un integrante por lado de los mismos. Para ser más exactos cada participante humano se enfrentará a otro participante humano, a un objeto móvil y a un objeto fijo. Durante el experimento la persona podrá hacer click cuando crea que esta interactuando con otra persona. El objeto móvil, un bot, en primera instancia será un bot sombra del oponente humano con un cierto retardo en tiempo, un tipo de bot ya creado en la anterior etapa de implementación. Posteriormente se creará otro tipo de experimento con múltiples componentes, en el cuál se cambiará el tipo de bot sombra, por otro que realice la misma traza de movimientos que el oponente humano pero con una diferencia de distancia impuesta.

Con esta plataforma se pretenden analizar todos los aspectos presentes en un intercambio comunicativo humano puesto que existen elementos que se refieren a las contingencias de la interacción que nos permiten identificar que una comunicación real se está produciendo. Pretendemos analizar las propiedades que permiten a los interlocutores construir patrones de interacción comunes que les hagan regular mutuamente sus acciones.

1.2. Contexto en el que se realiza el proyecto

La cognición social constituye en la actualidad un área de gran interés en diversos campos de la psicología, siendo múltiples los trabajos que se han preocupado por delimitar su concepto y desarrollo. La cognición social incluye aspectos tanto "sociales" como "cognitivos" de la representación del mundo en las mentes de las personas, y por tanto, es un concepto más amplio que el de Teoría de la Mente o ToM[1]. No obstante, ambos conceptos hacen referencia a la destreza cognitiva que capacita a los humanos para informar de los estados mentales propios y de las otras personas (creencias, deseos, emociones, intenciones), y entender que estas representaciones basadas en sensaciones y percepciones, no siempre se corresponden con la realidad (Astington, Harris y Olson, 1988[3]). Por consiguiente, la cognición social es esencial para predecir y explicar el comportamiento de las personas (Astington, 1993[2]; Flavell y Miller, 1998[14]), tanto en el plano de la acción (Mitchell, 1997[31]) como en el comunicativo (Bishop, 1997[7]; Sperber y Wilson, 2002[39]).

Sin embargo, en interacciones específicas como parecen existir en los fenómenos de atribución de "emociones" entre interlocutores en una conversación, se puede ver que no partimos de la observación sensorial de los movimientos corporales y faciales (la posición, orientación, tono de voz, gestos, sonrojo,...) para deducir lo que está sintiendo el otro, sino que reconocemos las emociones ajenas de manera directa, espontánea y sin ningún tipo de análisis, al mismo tiempo que se genera en nosotros un sentimiento similar por empatía. Por tanto, en nuestras interacciones sociales se expresan capacidades interactivas básicas, que recogen la perspectiva de quien forma parte de esa situación comunicativa, y capacidades inferenciales abstractas, que analizan lo dicho y pueden desplegarse, sin necesidad de participar efectivamente de la interacción.

[1] de la abreviatura del inglés *Theory of Mind*, propuesto originalmente por Premack y Woodruff (1978)[34]

Los investigadores de la cognición social son cada vez más conscientes de que hay muchos fenómenos que no se pueden entender mediante la investigación de las situaciones "*off-line*". El paradigma de cruce perceptual de Auvray et al (2009)[5] ofrece el paradigma más sencillo para el estudio de estas interacciones en línea: dos personas, un espacio unidimensional, un bit de información, y una respuesta tipo sí / no.

1.3. Metodología: Desarrollo ágil de software

Este proyecto nació a partir de una propuesta de prácticas que luego se convirtió en proyecto de fin de carrera. Por ello, alejándome del modelo clásico de desarrollo de software (requerimientos, análisis, diseño, implementación, ...), decidí optar por un tipo de metodología más ágil y práctico, que tan de moda se está poniendo en la actualidad.

He ido desarrollando mi trabajo marcándome una serie de hitos que, en si mismos, tuvieran visibilidad funcional. Es decir, una vez conseguido el objetivo marcado por el hito, tendría algo que poder mostrar y poner en funcionamiento. Esto permite introducir cambios clave que en una metodología clásica no se podrían añadir hasta el fin del proceso y vuelta a empezar. Un ejemplo sería que si quisiera implementar una web para vender un producto, me marcaría como primer objetivo fabricar una web con un botón que permita a un usuario comprar de un único tipo de producto una cantidad fijada del mismo, en vez de plantearme si debo construir la base de datos, montar un servidor, diseñar la página web, etc.

Durante las distintas etapas de consecución de un objetivo marcado, he intentado seguir la filosofía "*Less talking More doing*" ("Menos hablar y más hacer"), por lo que me documentaba y aprendía lo suficiente como para ponerme en el menor tiempo posible a implementar una solución. No realizaba siempre una misma estructura de pasos a seguir para alcanzar mi meta (metodología flexible), ya que cada tarea me exigía un grado diferente de esfuerzo. Si tuviera que dar un esquema general de pasos a seguir, sería el siguiente:

1. Una vez marcado el hito a conseguir, dividir en tareas el trabajo a realizar de forma lógica.
2. Documentarse lo suficiente para poder empezar implementar una tarea.
3. Realizar la tarea. Si se encuentra algún problema volver a documentarse para resolverlo.
4. Si aún faltan tareas por realizar, volver al paso 2 con la tarea siguiente, si no continuar al siguiente paso.
5. Comprobar el buen funcionamiento de la tarea en su conjunto antes de enseñársela al cliente (en mi caso el director del proyecto).

1.4. Trabajo a realizar

Para alcanzar el objetivo de este proyecto, que es conseguir un marco de ejecución virtual para experimentos de cruce perceptual, se realizarán las siguientes tareas:

1. En primer lugar se construirá un *framewok* simple para realizar experimentos de cruce perceptual, los experimentos estarán formados únicamente por dos personas, una contra otra. Esta primera construcción servirá de base a las siguientes implementaciones.

2. En segundo lugar se implementará un *framewok* partiendo del anterior que permita experimentos "persona contra persona" y "persona contra bot". En este marco seguimos teniendo un integrante por lado del experimento, pero se añade un nuevo tipo de participante, el bot. Se crearán diferentes tipos de bots, unos enviarán posiciones aleatorias, otros representarán la traza de movimientos de un experimento anterior, otros seguirán una política de movimiento según la posición del participante humano y finalmente otros repetirán los movimientos que este haciendo su oponente humano con un cierto retardo en tiempo. Al final de cada experimento cada persona deberá afirmar si ha interactuado con otra persona o con una máquina.

3. En tercer lugar se añadirá funcionalidad al *framewok* anterior permitiendo realizar experimentos con más de un integrante por lado de los mismos. Para ser más exactos cada participante humano se enfrentará a otro participante humano, a un objeto móvil y a un objeto fijo. Durante el experimento la persona podrá hacer click cuando crea que esta interactuando con otra persona. El objeto móvil, un bot, en primera instancia será un bot sombra del oponente humano con un cierto retardo en tiempo, un tipo de bot ya creado en la anterior etapa de implementación. Posteriormente se creará otro tipo de experimento con múltiples componentes, en el cuál se cambiará el tipo de bot sombra, por otro que realice la misma traza de movimientos que el oponente humano pero con una diferencia de distancia impuesta.

4. Por último, implementaré un visor de gráficas, que permita el estudio de la información acumulada durante los experimentos.

1.5. Herramientas utilizadas

La herramientas que se han utilizado a lo largo del desarrollo de la aplicación, atendiendo a su funcionalidades, han sido las siguientes:

- Entorno de programación para el desarrollo de la aplicación web.
  - Eclipse Java EE IDE for Web Developers. Versión: Juno Service Release 1, con un plug-in para Maven
- Trazado de bocetos y construcción de esquemas y figuras.
  - Evolus Pencil versión 2.0.2
- Editores de HTML, usados sobretodo al principio, antes de añadir el servidor, para construir las paginas web y posteriormente como bancos de pruebas para funcionalidades de HTML5, JavaScript y CSS.
  - Notepad++ (más usado), última versión 5.9.6.2
  - HTML-Kit 292
  - BlueGriffon versión 1.5.2
  - Komodo Edit versión 7.1.3
- Biblioteca para facilitar el uso de JavaScript.
  - jQuery versión compacta "jquery-1.8.3.min.js"
- Construcción visores de gráficas.
  - Easy Java Simulations versión 4.3.7

1.6. Estructura del documento

La estructura de esta memoria esta dividida en seis capítulos, incluyendo este capítulo introductorio. En el capítulo 2 se define el contexto en el que se mueve la aplicación y descripción de experimentos relacionados con el paradigma de cruce perceptual. En el capítulo 3 se describe la tecnología utilizada para la implementación de la aplicación distinguiendo entre las distintas partes de la misma. En el capítulo 4 se describe la aplicación desarrollada. En el capítulo 5 se describe la información almacenada por la aplicación para su posterior tratamiento y resultados obtenidos de la medición de prestaciones temporales de la misma. En el capítulo 6 se recogen las conclusiones extraídas durante el desarrollo del proyecto y las diferentes líneas de trabajo futuro.

1.7. Planificación

Durante los 7 meses de duración del proyecto, se han realizado las tareas que se muestran en el diagrama de Gantt contenido en la figura 1.1. Es necesario señalar que, por la metodología seguida a la hora de llevar a cabo el desarrollo, dentro de cada etapa de implementación existe un tiempo variable paralelo de documentación.

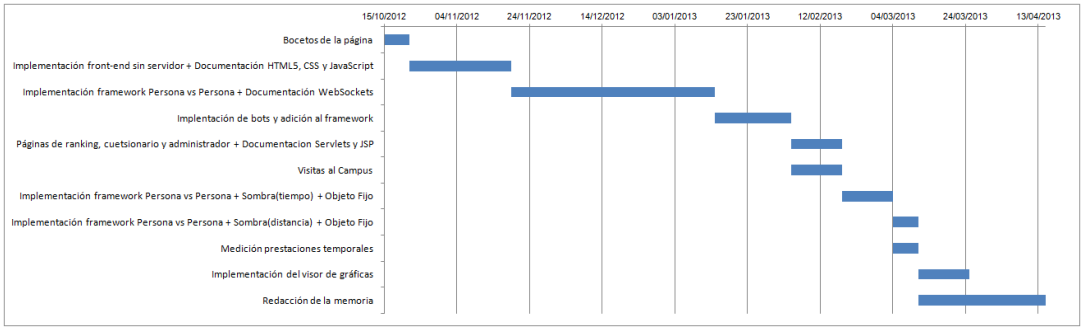

Figura 1.1: Diagrama de Gantt de las actividades realizadas.

Capítulo 2

Estado del arte

En este capítulo se describe en profundidad el ámbito donde se enmarcan los experimentos para los que se quiere construir un marco de ejecución virtual donde poder representarlos, además de una descripción de los mismos. En la primera sección se presenta la definición de cognición social, su visión más clásica y algunos cambios en el enfoque que en los últimos años se han ido introduciendo. En la segunda sección se introduce el concepto de cruce perceptual y se describen más en detalle los trabajos realizados en la universidad francesa de Compiègne en 2009 (Auvray, Lenay y Stewart, 2009[5]). En la tercera sección se describe, el trabajo de modelización y un análisis de las dinámicas que se generan (Iizuka y Di Paolo et al., 2007[25]). En la cuarta sección se da una visión general de trabajos relacionados con el paradigma de cruce perceptual.

2.1. Cognición social

La cognición social estudia la manera en que la gente procesa la información social, en particular su codificación, almacenamiento, recuperación y aplicación en situaciones sociales. Se trata de un proceso neurobiológico, psicológico y social, por medio del cual se perciben y reconocen a otros seres en interacción social. El enfoque de la cognición social en el procesamiento de la información tiene muchas afinidades con su disciplina hermana, la psicología cognitiva. En general, se necesitan y despliegan mecanismos para evaluar las intenciones de los demás a través de su conducta, la lectura de su expresión facial y sus movimientos corporales, para de esa manera interpretar las intenciones subyacentes.

La cognición social constituye en la actualidad un área de gran interés en diversos campos de la psicología, siendo múltiples los trabajos que se han preocupado por delimitar su concepto y desarrollo (ej. Bartsch y Wellman, 1995[6]; Flavell, 2000[13], 2004[12]; Flavell y Miller, 1998[14]; Garfield, Peterson y Perry, 2001[20]; Repacholi y Slaughter, 2003[35]; Wellman, Cross y Watson, 2001[43]). La cognición social incluye aspectos tanto "sociales" como "cognitivos" de la representación del mundo en las mentes de las personas, y por tanto, es un concepto más amplio que el de Teoría de la Mente o ToM[1]. No obstante, ambos conceptos hacen referencia a la destreza cognitiva que capacita a los humanos para informar de los estados mentales propios y de las otras personas (creencias, deseos, emociones, intenciones), y entender que estas representaciones basadas en sensaciones y percepciones, no siempre se corresponden con la realidad (Astington, Harris y Olson, 1988[3]). Por consiguiente, la cognición social es esencial para predecir y explicar el comportamiento de las personas (Astington, 1993[2]; Flavell y Miller, 1998[14]), tanto en el plano de la acción (Mitchell, 1997[31]) como en el comunicativo (Bishop, 1997[7]; Sperber y Wilson, 2002[39]).

[1] de la abreviatura del inglés *Theory of Mind*, propuesto originalmente por Premack y Woodruff (1978)[34]

Para abordar qué capacidades cognitivas se necesitan en los procesos de cognición social, teniendo en cuenta que la ejecución de la cognición general o ejecutiva tiene que ver con procesos cómo la rapidez en el procesamiento de información, memoria verbal, memoria de trabajo, atención sostenida, funcionamiento sensomotor, funciones ejecutivas y procesamiento verbal, etc, se han propuesto diferentes teorías, entre las que destaca dentro del ámbito más clásico la Teoría de la Mente (ToM). La ToM se define como la capacidad para atribuir un estado mental (pensamientos, emociones, deseos, creencias, intenciones) a las otras personas. Se constituye a partir de otros procesos como el análisis de la dirección de la mirada, el procesamiento de información no verbal, la asignación de metas e intenciones. Se desarrolla de manera paulatina desde la infancia, aunque es evidente hacia los cuatro años de edad con la atribución de creencias falsas (Wimmer y Perner, 1983[45]).

2.1.1. Teoría de la Mente

El término teoría de la mente (ToM) fue utilizado por primera vez por Premack y Woodruff en 1978[34] cuando de forma experimental observaron que los chimpancés poseían capacidad de inferencia. Este término se verá posteriormente sustentado por lo que Brothers definirá como «hipótesis del cerebro social»[8] entendido como una respuesta evolucionista a los cambios que presenta el cerebro frente a un entorno social cada vez más complejo.

Respecto a la ToM han surgido numerosas construcciones teóricas que han tratado de conceptualizar las diferentes hipótesis existentes. Los modelos más conocidos se enumeran a continuación:

- **La perspectiva modular**. Desarrollada por Fodor[15], consistente en proponer la existencia de una ToM independiente. Al igual que determinadas capacidades cognitivas se encuentran asociadas a funciones específicas del cerebro se presupone que la ToM se limita al procesamiento únicamente de la información con contenido social.

- **La perspectiva teoría-teoría o «metarrepresentacional»**. Perner[33] representa un modelo no modular que sugiere que durante la infancia se adquieren distintos niveles de habilidades representacionales. El hecho de tener auténticas metarrepresentaciones permite teorizar acerca de las representaciones de los otros.

- **El concepto de Frith**[16]. Sobre un déficit de la ToM propone que los síntomas de la esquizofrenia se explican por la presencia de una representación cognitiva errónea de las intenciones de uno mismo y de los demás. Frith señala que los pacientes con esquizofrenia presentan dificultades para distinguir la objetividad de la subjetividad y de este modo se mantienen falsas creencias en forma de ideas delirantes. Así mismo existiría una incapacidad para distinguir los estímulos externos de las intenciones internas que podrían estar en la base de las experiencias de control y pasividad.

- **El modelo de Hardy-Baylé**[21]. Este modelo hipotetiza sobre una ToM deficitaria que estaría en relación con un déficit ejecutivo. Según esto, los pacientes con desorganización del pensamiento, el lenguaje y las habilidades sociales son los que peor realizarían las tareas ToM porque son incapaces de monitorizar sus propias acciones. La ausencia de representación mental de la propia acción deseada comprometería la capacidad del paciente para asignar estados mentales a las acciones de otras personas. Siguiendo este modelo, el déficit en ToM se presentaría únicamente en los pacientes con predominio de la desorganización de pensamiento y de lenguaje, sin embargo, existen estudios controvertidos al respecto.

- **Teoría de la simulación**. Este modelo se fundamenta en la observación mediante RMN (Resonancia Magnética Nuclear) de la existencia de neuronas espejos responsables de la empatía localizadas predominantemente en el área de Broca (sección del cerebro humano involucrada con la producción del habla, el procesamiento del lenguaje y la comprensión), que se activan cuando se observan en los demás determinados movimientos de las manos o la boca (Gallese y Goldman[19]).

- **Otras propuestas alternativas** se encuentran en Bailey y Abu-Akel[1] que proponen un modelo de continuidad en el déficit de la ToM y hacen una clasificación que comprende desde la ToM verdaderamente deteriorada a la hiper-ToM asociada con la sobregeneración cuantitativa de hipótesis o la sobreatribución de estados mentales que se plantea como posibilidad de desarrollo de sintomatología paranoide.

2.1.2. Nueva corriente de pensamiento. Superando la Teoría de la Mente clásica

Si pensamos cuál sería la forma en la que la ToM nos describe, frente a una conversación, se adoptaría una actitud distanciada, objetiva, puramente interesada en la deducción a partir de la exploración del contenido de los mensajes.

Sin embargo, existen numerosos estudios que destacan como muchos aspectos de la comunicación humana no son el resultado de ningún análisis. Por ejemplo, existen capacidades expresivas que los bebés desarrollan y que les permiten participar en interacciones con sus progenitores, antes de poder hablar ni de explicar su conducta. Esta capacidad que aparece a edades muy tempranas, no desaparece para ser sustituida por una nueva aptitud cognitiva, más madura, que emerge con el inicio del lenguaje y que permite interpretar las interacciones comunicativas en términos predictivos. Al contrario, pervive como soporte natural de nuestras interacciones, generando una experiencia psicológica esencial para los procesos de socialización.

Al estudiar lo que ocurre en los fenómenos de atribución de emociones entre interlocutores en una conversación, se puede ver que no partimos de la observación sensorial de los movimientos corporales y faciales (la posición, orientación, tono de voz, gestos, sonrojo,...) para deducir lo que está sintiendo el otro, sino que reconocemos las emociones ajenas de manera directa, espontánea y sin ningún tipo de análisis, al mismo tiempo que se genera en nosotros un sentimiento similar por empatía.

"Vemos" directamente que alguien está enfadado, molesto, triste o eufórico, sin necesidad de deducirlo de sus movimientos musculares o faciales. En relación con estas ideas, es importante señalar que, al igual que estos procesos de atribución emocional, existen fenómenos similares en los intercambios lingüísticos entre personas: nuestra percepción social está constituida por el significado de lo que oímos/leemos y no por un contenido sonoro/escrito que tengamos que analizar.

Por tanto, en nuestras interacciones sociales se expresan capacidades interactivas básicas, que recogen la perspectiva de quien forma parte de esa situación comunicativa, y capacidades inferenciales abstractas, que analizan lo dicho y pueden desplegarse, sin necesidad de participar efectivamente de la interacción. Estas últimas son las que exclusivamente destaca la ToM clásica y, precisamente estas, son las únicas que manifiestan los niños autistas. Desde sus primeros meses de vida, los autistas no parecen sintonizar ni mostrar especial preferencia por la información visual de los rostros ni por la que procede de otras voces. La interacción con personas les merece similar atención que con otros objetos físicos. Su forma de interpretar a los demás se reduce, como propone la ToM clásica, a su análisis.

2.2. Cruce perceptual: paradigma de interacción más simple

Los investigadores de la cognición social son cada vez más conscientes de que hay muchos fenómenos que no se pueden entender mediante la investigación de las situaciones *"off-line"* únicamente, centrándose en los mecanismos individuales y en una perspectiva de mero observador. El paradigma de cruce perceptual de Auvray et al (2009)[5] ofrece posiblemente el paradigma más sencillo para el estudio de estas interacciones en línea: dos personas, un espacio unidimensional, un bit de información, y una respuesta tipo sí / no.

En este experimento, realizado por un equipo de la Universidad de Compiègne, los participantes interactúan por medio de un dispositivo de retroalimentación táctil moviendo un ratón a izquierda y derecha. Este movimiento controla la posición de un sensor en un espacio unidimensional virtual donde se pueden encontrar objetos fijos o móviles. Al cruzarse con un objeto en el espacio virtual se produce una activación eléctrica sobre el dedo: esta es la única información sensorial a disposición de los participantes. Se les dice que un objeto móvil está controlado por otra persona y se les pide que hagan click con el ratón cada vez que piensen que están en contacto con ese objeto. La tarea no es trivial debido a la presencia de objetos estáticos y un objeto sombra que se mueve exactamente del mismo modo que la otra persona. A pesar de esta ambigüedad, los participantes son capaces de hacer click con más frecuencia cuando está en contacto directo con el sensor de la otra persona y no con la sombra.

Este estudio ha provocado mucho interés en diferentes áreas de investigación, incluyendo la psicología experimental, modelado computadora/robot, la filosofía, la psicopatología, e incluso en el campo del diseño. Una explicación más detallada del experimento se describe a continuación.

Estudio del cruce perceptual

Para aclarar la noción de "cruce perceptual", y hacer posible un análisis preciso de las dinámicas mutuas, se ha reducido el espacio de acción de los participantes a una sola dimensión, y se ha reducido también el repertorio sensorial de entrada a una sola estimulación de todo o nada (sólo 1 bit de información en cada punto de tiempo). Dos participantes con los ojos vendados se colocan en habitaciones diferentes, y sólo pueden interactuar a través del dispositivo. Cada uno de ellos explora una pantalla de ordenador con un ratón, y recibe la estimulación táctil en el dedo índice de su mano libre. Los movimientos del ratón controlan los movimientos de un campo receptor de 4 píxeles en un espacio unidimensional. Sólo los movimientos horizontales del ratón se toman en cuenta. El espacio de acción consiste en una línea recta de 600 píxeles de largo, sus extremos se suponen unidos para formar un círculo continuo a fin de evitar efectos de borde. Diversos objetos, consistentes en píxeles negros, se colocan en esta línea. Cada vez que el campo receptor encuentra un píxel negro, el participante recibe una estimulación táctil todo o nada en el estimulador táctil (Ver figura 2.1).

Dos sistemas de este tipo están conectados en una red, por lo que los dos participantes comparten el mismo espacio unidimensional. Hay tres tipos de objetos que cada participante puede encontrar:

1. El objeto-cuerpo de otro participante (su cuerpo percibido), que coincide exactamente con su campo receptor (4 píxeles de ancho). Cuando los dos participantes se encuentran en la misma posición, cada uno recibe una estimulación táctil todo o nada. Esta situación se llama "cruce perceptual."

2. Un objeto fijo llamado "señuelo fijo": se trata de un segmento de 4 píxeles de ancho. El señuelo fijo del participante 1 es invisible para el participante 2, y se coloca en una posición diferente que el señuelo fijo del participante 2 (Ver figura 2.2).

3. Un objeto en movimiento (4 píxeles de ancho) llamado "señuelo móvil." Con el fin de garantizar que el señuelo móvil tuviera la misma riqueza de movimiento que el objeto-cuerpo del otro participante, pero sin ser sensible a los cruces perceptuales, se une por un enlace virtual rígido al campo receptor del objeto-cuerpo de la pareja. Así pues, el señuelo móvil sigue exactamente, pero a una distancia constante, todos los movimientos realizados por el compañero (Ver figura 2.2). En todo lo que sigue, las distancias entre los dos objetos se miden en píxeles.

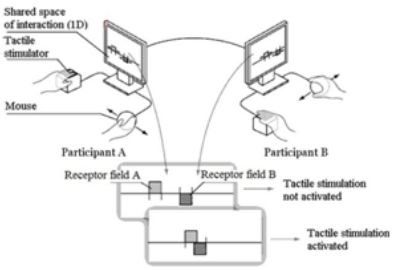

Figura 2.1: Espacio unidimensional de interacción perceptual. Con el ratón de su ordenador, cada sujeto mueve un campo receptor en línea recta a lo largo de un espacio digital compartido. Cuando los dos campos receptores se encuentran, cada usuario recibe un estímulo táctil en la mano libre. [Extraído de Lenay y Stewart, 2012[26]]

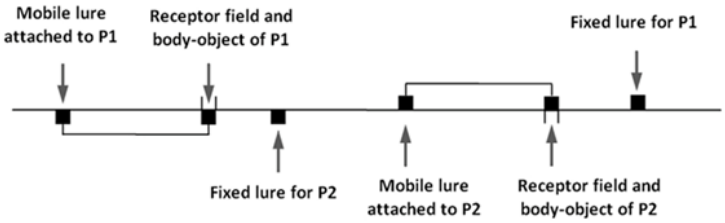

Figura 2.2: Ilustración esquemática del espacio unidimensional explorado por los sujetos. El sujeto P1 recibe un estímulo táctil siempre que se encuentre bien con su objeto fijo, bien con el campo receptor de P2 o bien con el objeto móvil asociado al campo receptor de P2. [Extraído de Lenay y Stewart, 2012[26]]

Esta configuración experimental hace que sea posible poner a prueba una hipótesis teórica: ¿incluso aunque el señuelo móvil y el objeto-cuerpo del contrincante (que corresponde con su campo receptor) tengan exactamente los mismos movimientos, los participantes serán capaces de distinguirlos únicamente sobre la base de la interacción?

Diez parejas de participantes tomaron parte en este experimento. A los participantes se les vendaron los ojos y se les colocó en diferentes habitaciones. Se les explicó que los movimientos de izquierda / derecha del ratón les permiten moverse en un espacio unidimensional compartido. En este espacio se pueden encontrar tres tipos de objetos: un objeto fijo, un objeto móvil, y el objeto-cuerpo del otro sujeto. La relación entre el objeto móvil y el objeto-cuerpo de la pareja no se les explicó. Las instrucciones les pedía hacer click en el botón izquierdo del ratón cuando pensaran que se habían encontrado con un humano. Esta configuración experimental tiene una serie de ventajas:

1. La situación perceptual es radicalmente nueva para los sujetos. Así se evita la importación directa de conocimientos ya elaborados.

2. La reducción de la entrada sensorial obliga a un despliegue espacial y temporal de las actividades de percepción, y esto hace que sea posible grabar y analizar en detalle.

3. La simplicidad de la configuración hace posible dilucidar las condiciones suficientes para un esquema explicativo detallado de la dinámica colectiva, que se espera tenga cierta generalidad. Los resultados para todos los participantes y todas las sesiones mostraron que la mayoría de los clicks (62 %) se produjo cuando los dos socios estaban uno frente al otro, es decir, en una situación de cruce perceptual. (Ver figura 2.3).

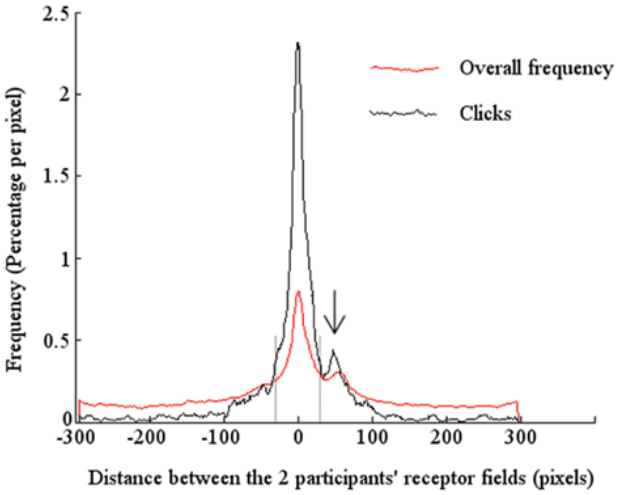

Figura 2.3: Distribución de frecuencias en función de la distancia entre los campos receptores de los dos participantes. La línea negra representa la frecuencia total de clicks realizados: el 62 % de la distribución se encuentra entre ± 30 píxeles. La línea roja representa la frecuencia total de estímulos recibidos por los sujetos: el 28 % de la distribución se encuentra entre ± 30 píxeles. En ambos casos, hay un pico claro en torno a la distancia de 0 píxeles, es decir, la situación de cruce perceptual, lo que demuestra que hay un atractor en este punto, al menos en el débil sentido descriptivo una vez que los sujetos han alcanzado la situación de cruce perceptual tienden a permanecer en esta configuración dinámica estable. Un pico menor a la distancia de 50 píxeles (marcado con una flecha) se corresponde con el señuelo móvil. [Extraído de Lenay y Stewart, 2012[26]]

A continuación, se analizó la distribución de clicks como una función de la causa de los estímulos recibidos por el participante durante los precedentes 2 segundos. Los resultados de todos los participantes muestran que el 66 % (± 4) de los clicks siguen los estímulos de cruce perceptual; el 23 % (± 10) de los clicks siguen los estímulos debidos al señuelo móvil, y sólo el 11 % (± 9) siguen estímulos provocados por el señuelo fijo. Estos resultados muestran que los participantes son capaces de distinguir entre las tres categorías de objetos que se encuentran en el espacio unidimensional. Distinguen entre el campo receptor de la pareja y un objeto, ya sea fijo o móvil. Este éxito global puede parecer sorprendente, ya que, por construcción, el señuelo móvil tiene exactamente los mismos movimientos que el campo receptor de su pareja. Parece que lo que se reconoce es, de hecho, la actividad perceptual del sujeto, y no sólo la estructura objetiva de los movimientos (Wilkerson, 1999[44]). Sin embargo, un análisis más detallado muestra que este aparente éxito a nivel general oculta lo que era en realidad un fracaso a nivel individual.

En primer lugar, se llevó a cabo una comparación entre la distribución de los clicks y la distribución de los estímulos táctiles recibidos. Los resultados globales para todos los participantes muestran que el 52 % (± 12) de los estímulos provienen de un cruce perceptual, el 33 % (± 12) provienen del señuelo fijo, y sólo el 15 % (± 6) del señuelo móvil (Ver tabla 2.1; figura2.3).

	Receptor field	Mobile lure	Fixed lure
Percentage of clicks	66 ± 4	23 ± 10	11 ± 9
Percentage of stimulations	52 ± 12	15 ± 6	33 ± 12
Ratio clicks/stimulations	1.26	1.51	0.33

Tabla 2.1: Distribución de los clicks y de las simulaciones táctiles. [Extraído de Lenay y Stewart, 2012[26]]

Cuando se calcula la ratio de clicks/estímulos, nos encontramos con 0,33 para el señuelo fijo, 1,26 para el cruce perceptual, y 1.51 para el señuelo móvil. Estos resultados muestran una diferencia importante entre el señuelo fijo por un lado (0.33) y las entidades móviles por el otro (1,26 y 1,51). Los participantes tienen una probabilidad de hacer click que es cuatro veces mayor si la estimulación proviene de una entidad móvil que si es debida al señuelo fijo. Por lo tanto, la ratio entre clicks y estímulos muestra que, en general cada participante no parece distinguir entre estímulos debidos a cruce perceptual y estímulos debidos al señuelo móvil (1,26 vs 1,51). La diferencia en los clicks en el señuelo móvil y en el campo receptor de la pareja (23 vs 66 %) parece deberse sólo a las estrategias de los movimientos, tales que, los encuentros con el señuelo móvil son mucho menos frecuentes que los encuentros debidos al cruce perceptual (15 vs 52 %). Si los participantes tienen éxito en la tarea, es esencialmente porque logran situarse cara a cara con su contrincante, y no porque reconozcan en el patrón de estimulación alguna pista con la que discriminar el campo receptor de la pareja del señuelo móvil. La única diferencia reside en la interacción en sí misma. Con el fin de dar cuenta de estos resultados, hay dos cosas que deben ser explicadas. Por un lado, la capacidad de los participantes a encontrar situaciones de estar "cara a cara" y, por otro lado, las razones que les lleva a hacer click.

a) Atractor en dinámicas colectivas

Cabe señalar que todas las observaciones realizadas con estas configuraciones muestran que la percepción de un objeto en una posición es determinada mediante su exploración activa y reversible: los sujetos van y vienen alrededor de la singularidad que provoca un retorno sensorial (Sribunruangrit et al. , 2004[40]). Por lo tanto, hay una estrategia general que consiste en invertir el movimiento del campo receptor después de un evento sensorial. En la medida en que la estrategia de percepción de cada participante consiste en invertir su movimiento después de una alteración en la entrada sensorial, si un participante se encuentra con su contrincante invertirá su movimiento mientras que el segundo hará lo mismo. Los dos campos receptores entrarán, por lo tanto, en una "especie de danza". Esto puede describirse como un atractor en la dinámica colectiva; un atractor que no es un punto fijado espacialmente, sino una región que puede ser desplazada. A pesar de que los participantes no tienen un objetivo específico de colaboración, los esfuerzos simultáneos para discriminar la presencia de su pareja producen un atractor en las dinámicas colectivas de sus actividades perceptivas (Froese y Di Paolo, 2010[18], 2011a[17]).

b) Las razones que llevan a los sujetos a hacer click

Si se estudia los acontecimientos que preceden a cada click, se observa que si en los últimos 2 segundos de su actividad perceptiva un sujeto se encuentra:

1. pocos estímulos, no se constituye la percepción y la probabilidad de hacer click es baja;
2. muchos estímulos, pero para un objeto que se reconoce como fijo (estabilidad sensomotora), la probabilidad de hacer click es de nuevo bajo;
3. pero si hay muchos estímulos, para un objeto que permanece indeterminado espacialmente, la probabilidad de hacer click es alta (ver figura 2.4).

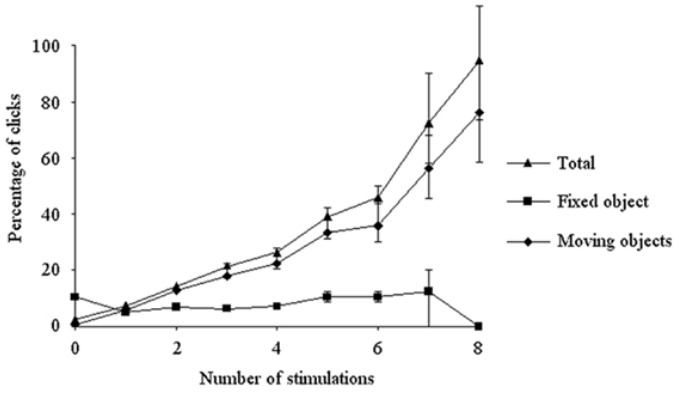

Figura 2.4: Probabilidad de hacer click. La probabilidad de que los participantes hagan click, se representa gráficamente en función del número de estimulaciones distintas recibidas durante los últimos 2 segundos. Triángulos: estímulos totales (cuerpo-objeto, objeto fijo y atractivo móvil); Cuadrados: estímulos debido a encuentros con el objeto fijo; Rombos: estímulos debido a los encuentros con un objeto en movimiento (avatar o señuelo móvil). Las barras de error representan los errores estándar de las medias. [Extraído de Lenay y Stewart, 2012[26]]

En este último caso, es probable que el objeto sea el del otro participante, pero también es posible que sea el del señuelo móvil. Así, los clicks de los participantes se pueden explicar en gran medida por la conjunción de dos criterios, uno negativo y uno positivo:

1. "Otro sujeto" es algo que se resiste a la determinación espacial precisa: no es ni un objeto fijo, ni un objeto con movimientos determinados por una regla simple.
2. Sin embargo, al mismo tiempo, "otro sujeto" es algo que mantiene su presencia. Esta es de hecho una característica del objeto-cuerpo del otro participante, pero no del señuelo móvil, debido a que es sólo este objeto-cuerpo el que tiene un campo receptor sensible a su vez a la presencia de objetos, es decir, es probable que cambie su comportamiento de acuerdo a la entrada sensorial que recibe. La (única) diferencia entre el campo receptor del otro participante y el señuelo móvil unido a él, es que sólo el primero es sensible a mi presencia, y como se ha visto, esta sensibilidad está vinculada a una intencionalidad perceptiva que pretende continuamente permanecer cerca de una singularidad. Esta es precisamente una condición suficiente para la formación de un atractor en la dinámica conjunta lo cual tiende a aumentar la probabilidad de que el compañero este presente. Por lo tanto, el criterio que parece estar al servicio de los participantes para hacer click no es arbitrario, sino que se produce lógicamente de la reunión de dos intencionalidades. Este criterio es coherente con el contenido mismo de lo que ha de ser reconocido. El otro sujeto es reconocido sólo como algo que se resiste a su determinación precisa y, sin embargo, que persiste en estar presente.

Sin embargo, incluso si este criterio parece razonable, no hay suficiente garantía de distinguir entre el campo receptor de la pareja y el señuelo móvil. Si por un golpe de mala suerte es el señuelo móvil el que permanece presente, los participantes tienen la misma probabilidad de hacer click que en el campo receptor. Por ejemplo, si el campo receptor de mi contrincante se dedica a oscilar alrededor de un objeto situado a 50 píxeles desde mi posición, y que provoca un movimiento del señuelo alrededor de mi propia posición, voy a ser inducido a hacer click en el señuelo.

Conclusión del experimento

En este experimento en el que el objetivo es discriminar la presencia de otro sujeto, los individuos fallan mientras que la acción colectiva tiene éxito. Así, el éxito colectivo no puede explicarse por la capacidad individual de reconocer otro sujeto por medio de una sensación particular (Michael y Overgaard, 2012[30]). El éxito colectivo se explica principalmente por una dinámica colectiva que resulta de la participación de cada individuo en su actividad perceptiva en busca de un compañero. Los clicks provienen de una regla de decisión que parece ser prudente, pero que es insuficiente a nivel individual para distinguir entre las sensaciones específicas.

2.3. Modelo de detección de agentes sociales

Iizuka y Di Paolo (2007)[25] elaboraron un modelo mínimo de cruce perceptual inspirado por los experimentos de Murray y Trevarthen (1985)[32] y en los trabajos de Auvray et al (2009)[5]. Dos agentes moviéndose a derecha e izquierda deben interactuar cruzando sus sensores individuales en varias ocasiones. Sin embargo, si se le presenta una grabación de una interacción previa, el agente vivo debe alejarse de ella. Una descripción más detallada del estudio se presenta a continuación.

a) Modelo acoplado

Los agentes están obligados a discriminar si otro agente es un compañero de interacción en vivo o una grabación de la conducta del agente mediante el uso de sensores y motores mínimamente restringido. Los dos agentes, superior e inferior, se enfrentan entre sí en un espacio ilimitado unidimensional. Cada agente sólo puede moverse a izquierda y derecha horizontalmente. Un sensor de encendido/apagado está unido en el centro del agente y esta activado, por ejemplo, estableciéndose en 1 cuando se cruza con la pareja, mientras que se establece en 0 en caso contrario. Con una cierta probabilidad la información sensorial salta a un estado diferente (la probabilidad se establece en 0,05) en cada paso temporal del método de Euler como ruido sensorial. La presencia de ruido jugará un papel importante para los agentes para lograr la tarea, tal como se explicará más adelante.

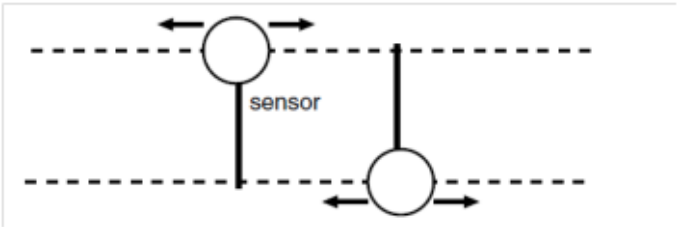

Figura 2.5: Esquema agentes superior e inferior. [Extraído de Iizuka y Di Paolo, 2007[25]]

b) Agentes móviles

Los agentes están controlados por una red neuronal recurrente de tiempo continuo (CTRNN), la cual consiste en 8 nodos totalmente conectados en el modelo. El tiempo de evolución de los estados de las neuronas se expresa mediante:

$$\tau_i \dot{y}_i = -y_i + \sum_{j=1}^{N} w_{ji} * z(y_j) + I_i, \qquad z(x) = 1/(1 + e^{-x-b_i})$$

Donde y_i representa el potencial de la célula de la neurona i, z_i es la tasa de disparo,τ_i (rango [1,100]) es su constante temporal, b_i (rango [-3,3]) es un término de sesgo, y w_{ij} (rango [-8 , 8]) es el peso de la conexión de la neurona, j, a la neurona i. I_i representa la entrada sensorial, que da sólo a una neurona sensorial. El número de neuronas, N, se establece en 8. La información sensorial se calcula multiplicando 1/0 (on / off) de señal por un parámetro de ganancia (rango [1, 100]), que está codificado genéticamente. Hay dos neuronas efectoras para controlar la actividad motora. Del mismo modo, la salida del motor se calcula a partir de la diferencia de las tasas de disparo de las neuronas efectoras, que corresponde a una rango de [-1,1] y es entonces multiplicada por un parámetro de ganancia (rango [1, 100]).

c) Configuración evolutiva

Los agentes se desarrollaron utilizando un algoritmo genético con elitismo. Se diferencian agentes en dos grupos: los representados en la parte superior y la parte inferior de la figura 2.5. Con el fin de permitir que cada grupo de agentes tenga diferentes estrategias de comportamiento, se usan dos poblaciones. Cada población tiene 20 agentes, que se evalúan mediante la interacción con los mejores 5 agentes de otra población. En cada población, todos los parámetros de la red neuronal y las ganancias están representadas por un vector de valores reales ([0,1]) que se decodifica linealmente al rango correspondiente a los parámetros. Se utilizan operadores de intercambio genético y de vectores de mutación, que añaden un pequeño vector aleatorio con el genotipo de valor real. Los mejores 5 agentes de la población se conservan y 10 agentes se sustituyen por agentes con mutación que son seleccionados de las poblaciones originales basadas en una función de adecuación (*fitness*) y los 10 agentes restantes se sustituyen mediante cruce genético entre dos agentes seleccionados.

La función de adecuación o *fitness,* que puede observarse en la figura 2.6, se calcula sobre la base de dos factores. Uno de ellos es la cantidad de veces que el agente puede cruzar su posición central con la de un agente de interacción en vivo. La otra es cuanto puede el agente mantenerse alejado de un agente ficticio que sólo reproduce los movimientos de su pareja, registrados en la primera etapa (interacción unidireccional). Ambas condiciones comienzan a partir de las mismas configuraciones iniciales, tales como posiciones, velocidades, y estados neuronales y, entonces, en cierto punto, el agente de la parte superior se sustituye por los movimientos registrados. Por lo tanto, los agentes necesitan discriminar las dos condiciones a través de la interacción en curso, mientras que de alguna manera se explota la presencia de ruido. Esta forma de modelar el movimiento reproduce las condiciones análogas de la doble pantalla de televisión (Murray y Trevarthen., 1985[32]) y los experimentos de cruce perceptual (Auvray et al., 2009[5]).

En este trabajo, sólo los agentes inferiores fueron desarrollados para discriminar las interacciones en vivo de las grabadas. Esto significa que cada población tiene una función de adecuación diferente (Ver figura 2.6). La población de los agentes de la parte inferior es evaluada por ambos factores clave explicados anteriormente, mientras que solo se aplica el primer factor a los agentes de la parte superior. Con el fin de medir cuántas veces un agente no se cruza con la grabación durante una prueba, los tiempos máximos de cruce se establecieron arbitrariamente a 20 (un número estimado en ejecuciones piloto) y de no-cruce se cuentan restando los tiempos de cruce del mismo.

$$F_i^1 = \frac{1}{20}\sum_{j=1}^{20} \#Cross(a_i^1, a_j^2) \times (20 - \#Cross(a_i^1, Record(a_j^2)))$$

$$F_i^2 = \frac{1}{20}\sum_{j=1}^{20} \#Cross(a_j^1, a_i^2)$$

Figura 2.6: Funciones *fitness* o de adecuación para llegar a obtener la CTRNN óptima para cada población. [Extraído de Lenay y Stewart, 2012[26]]

d) Resultados

Los agentes evolucionados adquieren con éxito la capacidad de discriminar entre las dos condiciones clave. Aquí se muestra uno de dichos pares de agentes en detalle en un intento de sacar algunas conclusiones generales. La figura 2.7 muestra las trayectorias de ambos agentes bajo la interacción en vivo y la trayectoria del agente de la parte inferior cuando interactúa con el movimiento grabado del agente de la parte superior (interacción unilateral); las diferencias de posición entre agentes superior e inferior se muestran a la derecha. La coordinación de cruce se establece mientras que los dos agentes se mueven en una dirección y controlan sus aumentos y disminuciones de velocidad para cruzarse entre sí, sin perderse de vista el uno al otro. Bajo la condición de la interacción unilateral, se observa la caída de la coordinación. Aunque no existe una gran diferencia en el comienzo de la interacción, con el tiempo el agente de la parte inferior se aleja de los movimientos grabados. Como la interacción unilateral comienza de la misma manera que la interacción en vivo, es decir, ambas condiciones tienen los mismos estados internos y físicos en el momento de la conmutación, una distorsión de la interacción en curso por la acumulación de ruido sensorial debe causar la diferencia de comportamientos.

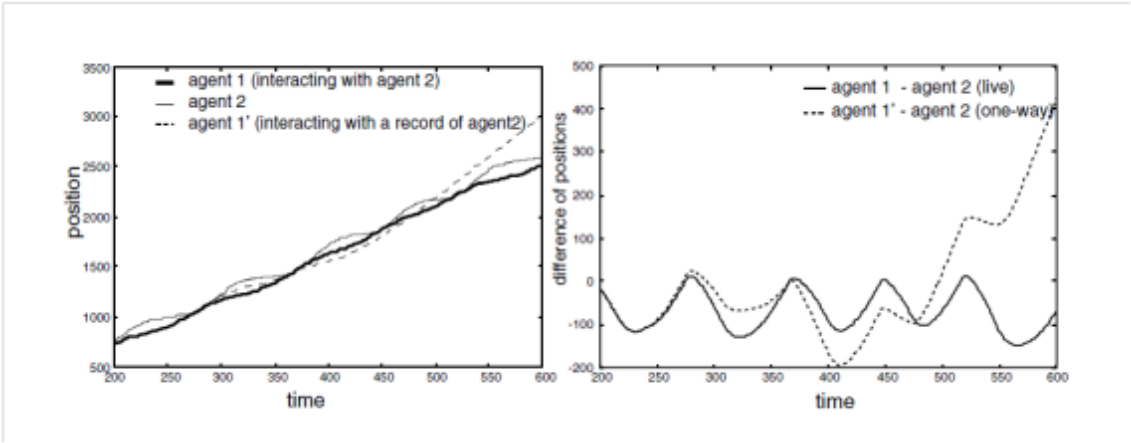

Figura 2.7: Trayectorias de los agentes debidas a las interacciones en vivo y unilateral. La líneas continuas delgada (agente de la parte superior) y gruesa (agente de la parte inferior) muestran los resultados de los comportamientos coordinados bajo la interacción en vivo. La línea discontinua muestra la trayectoria del agente inferior al interactuar con la repetición de la línea continua delgada (agente de la parte superior).(**izquierda**) La diferencia entre las dos trayectorias en cada condición. Cuando la línea cruza 0 significa que dos agentes se cruzan entre sí.(**derecha**) [Extraído de Iizuka y Di Paolo, 2007[25]]

Estabilidad de ruido

Sin ruido en la configuración actual, no hay forma de saber si la pareja es reactiva o no reactiva. En otras palabras, el agente de la parte inferior necesita explotar el efecto del ruido para lograr la tarea. La figura 2.8 muestra el efecto del ruido en comportamientos coordinados. La coordinación establecida entre dos agentes adaptativos se puede mantener sin ser destruida por el aumento de ruido. Aunque el ruido más fuerte rompe más la coordinación, la coordinación muestra una robustez frente a perturbaciones de ruido. Sin embargo, si el agente de la parte superior se sustituye por la grabación no reactiva, la coordinación no se establece más. Esto significa que el agente de la parte inferior explota la presencia de ruido para detectar que el compañero no es reactivo y evitarlo. A pesar del hecho de que los comportamientos del agente de la parte superior son idénticos en movimiento bajo las interacciones en vivo y unilaterales, el agente de la parte inferior no se coordina con la grabación. Esto significa que sólo las dinámicas mutuas de acoplado pueden mantener la coordinación, mientras se suprimen las perturbaciones inducidas por el ruido sensorial.

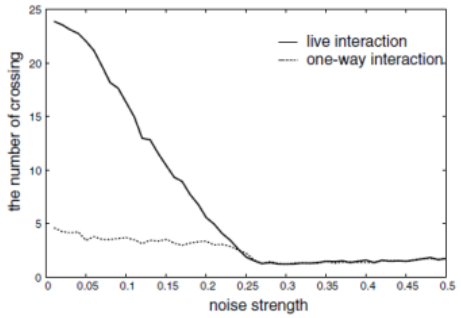

Figura 2.8: El número promedio de cruces que los agentes evolucionados contra la intensidad de ruido que se investiga a intervalos de 0,01. Para calcular las medias, se utilizaron 100 pruebas para cada intensidad de ruido. La intensidad de ruido durante la evolución es 0,05. [Extraído de Iizuka y Di Paolo, 2007[25]]

2.4. Extensiones y Modelos del Paradigma de cruce perceptual

El estudio de Auvray et al. (2009)[5] ha tenido una gran influencia en diferentes áreas de investigación. Debido a su simplicidad, el paradigma de cruce perceptual sirve como ejemplo ilustrativo de la importancia de la dinámicas de interacción en línea. En la tabla 2.2 se enumera y resume los estudios más importantes relacionados con el cruce perceptual.

Studies	Methods	Questions	Results
Auvray et al., 2009	Human behavior (VR)	Self-organization of coordination	Coordination arises despite individual's incapacity to explicitly discriminate live versus non-live interaction
Di Paolo et al., 2008; Rohde, 2010; ch. 6	Simulation modeling	Mechanisms underlying PC	Behavior observed in PC can emerge from simple behavioral control circuits in online interaction
Iizuka and DiPaolo, 2007; Di Paolo et al., 2008; Iizuka et al., 2009, 2012a	Human behavior (VR), simulation modeling	Individual modulation of interaction to discriminate online interaction from a recording	Simulated agents as well as individuals use active perceptual strategies and turn-taking to discriminate live interaction from one-sided coordination
Martius et al., 2008	Simulation modeling	Emergence of coordination in homeokinetic agents	If rewarded for seeking stimulation, PC, and agency detection emerge from homeokinetic control rule
Rohde and Di Paolo, 2008; Rohde, 2010; ch. 7; (Lenay et al., 2011)	Human behavior (VR) and simulation modeling	PC in 2D; embodiment and spatial properties	The results from PC in 1D transfer to 2D; oscillatory interaction persists and may serve exact localization; humans perceive entities whose location cannot be fully predicted as other agents
Froese and Di Paolo, 2008, 2010, 2011	Simulation modeling	Dynamic stability and mechanisms underlying PC	Coordination in PC paradigm is stable across many scenarios. Evolved solutions rely on subtleties in brain-body-environment interaction dynamics
Timmermans et al., 2011	Human behavior (VR) with autistic patients	Differences and similarities between autistic and non-autistic people	Similar motor behavior and coordination patterns. Small differences in failed clicks
Iizuka et al., 2012b	Human behavior (VR)	Emergence of symbolic communication through interaction	With training, humans develop turn-taking strategies and characteristic movements to represent different kinds of visual stimuli to the interaction partner in a PC experiment
Lenay and Stewart, 2012	Human behavior (VR)	Conscious recognition of the source of stimulation	Human classify the sources of stimulation if these are characterized by different sounds
Lenay and Stewart, 2012	Human behavior (VR)	Slow modulation of fast PC dynamics	Humans can negotiate a common distance between sensor and avatar in PC using the interaction as feedback to improve interaction

Tabla 2.2: Resumen de los estudios experimentales y de modelado más importantes de cruce perceptual. [Extraído de Auvray y Rohde, 2012[4]]

- **Versión bidimensional del experimento original por Lenay et al. (2011)[27]**. Diez parejas de participantes fueron colocadas en un espacio virtual de dos dimensiones. Se obtuvieron resultados similares al original: identificación exitosa del contrario en términos de porcentaje de clicks y distinción correcta debida a la frecuencia de estimulación de cada fuente. Lenay et al. (2011)[27] señalan que, mientras que los participantes exploran todo el espacio en busca unos de otros, una vez establecido el contacto, se vuelve a interacciones oscilatorias unidimensionales. Los patrones de comportamiento observado en los agentes robóticos, obtenidos por simulaciones de la tarea (Rohde y Di Paolo, 2008[37]; Rohde, 2010[36]; Lenay et al, 2011[27]), se ensayaron frente a los datos humanos. Hay evidencia del papel del escaneo oscilatorio para relocalizar al otro después de perder contacto. Se sugiere la imposibilidad de predecir con precisión la ubicación del contrario a pesar de la interacción coordinada, lo que puede explicar el comportamiento al hacer click(Auvray et al., 2009)[5]. No queda claro si la reducción de movimiento en una dimensión está relacionada con la anatomía del brazo humano.

- **Variación del paradigma por Iizuka et al. (2009[24],2012a[22]).** Probó empíricamente lo estudiado por simulaciones en anteriores estudios (Iizuka y Di Paolo de 2007[25] y Di Paolo et al, 2008[11]), la detección de agentes en un entorno, donde la coordinación unilateral (De Jaegher, 2009[10]) es una posibilidad teórica. Introdujo un importante cambio en el paradigma original, en lugar de colocar simultáneamente a la persona y su sombra en el mundo virtual, las pruebas fueron aleatorizadas para exponer a los participantes a una interacción en vivo (persona contra persona) o a una grabación de otro participante en un ensayo anterior. Los participantes tenían que decidir si habían percibido que la interacción era en vivo o no. Todos los sujetos empezaban a explorar y oscilar alrededor de las entidades encontradas de ambos tipos. Después de sólo unas pocas decenas de ensayos, los participantes desarrollaron un comportamiento de cambio de turno como una estrategia de sondeo activo. La diferencia importante con el paradigma de cruce perceptual de Auvray et al. (2009)[5] es que el procedimiento utilizado en este experimento aporta una solución de coordinación unilateral teóricamente posible, explica el paradigma del doble monitor de TV de Murray y Trevarthen (1985)[32] más fielmente en un entorno virtual mínimo. Auvray et al. (2009)[5] centraba su atención en el tipo de entornos y comportamientos que llevan a la coordinación social, mientras que Iizuka et al. (2009[24], 2012a[22]) se centró en cómo un individuo puede modular la dinámica de interacción para averiguar si una interacción es en vivo o no.

- **Variante del paradigma de cruce perceptual por Iizuka et al. (2012b)[23].** Parejas de participantes se enfrentaban a diferentes estímulos visuales (formas) y tenían que decidir después de 30 segundos de cruce perceptual si veían la misma forma o una diferente. Con la información proporcionada, los participantes aprendieron no sólo a esperar su turno en la interacción, sino también a acordar patrones característicos de movimiento para representar la forma que podían ver. Lo particularmente interesante es que el movimiento oscilatorio se utiliza para localizar al contrario, comunicarse y negociar un vocabulario común. Todas estas actividades son procesos funcionales diferenciados, sin embargo, todos ellos se producen simultáneamente

- **Variante del paradigma original de Lenay y Stewart (2012)[26]**. Evaluaron el grado en el que los participantes son capaces de reconocer explícitamente a su pareja. Los participantes recibían información sonora en lugar de táctil, cada sonido se asoció con uno de los tres tipos de objetos (contrario, sombra y fijo) del experimento original. Diez parejas de participantes fueron probados en 4 sesiones y la correspondencia sonido-entidad cambiaba en cada sesión. En las sesiones 1 y 2, el objeto móvil se encuentra a una distancia fija del contrario y en las sesiones 3 y 4, corresponde a las grabaciones de la trayectoria del contrario durante la sesión 2. Al final de cada sesión, se pedía a los participantes asignar los tonos a los distintos tipos de entidades. Los resultados mostraban que el objeto fijo se reconocía fácilmente. En cuanto a la diferenciación entre entes móviles: en las sesiones 3 y 4, los participantes los distinguían perfectamente; en las sesiones 1 y 2 solo acertaban en una de las sesiones. Se pudo observar una estrategia común para resolver la tarea. Los participantes identificaban primero el sonido correspondiente al objeto fijo, luego buscaban a su oponente, tratando de mantenerse en contacto con él, y finalmente, verificaban su clasificación buscando el objeto móvil. Este cambio en la estrategia respecto al original indica que el reconocimiento consciente requiere de la modulación de las dinámicas de cruce perceptual emergentes de la búsqueda mutua (Iizuka et al., 2009[24] y 2012a[22]). También se observó una mayor dificultad en discriminar la coordinación unilateral de una interacción previa, en lugar de a una interacción en vivo, de acuerdo con los resultados de Iizuka et al. (2009[24], 2012a[22]).

- **Otra variación del paradigma de cruce perceptual de Lenay y Stewart (2012)[26].** Se investigó si los participantes son capaces de ajustar sus acoplamientos sensomotores para facilitar el cruce perceptual. En esta variación del paradigma, los participantes pudieron ajustar la distancia entre su campo receptivo y su representación (perceptible por contrincante). La distancia podía ser ajustada usando clicks de ratón durante el cruce perceptual. Si la discrepancia es muy grande, se espera que la coordinación no se estabilice. La tarea de los participantes fue ajustar dicha distancia para disminuir cualquier desviación experimentada y estabilizar el cruce perceptual. Se encontró que, utilizando la dinámica de interacción como una señal de realimentación, los participantes fueron capaces de disminuir la discrepancia entre sus parámetros de distancia y lograr una interacción más suave.

- **Investigación del paradigma de cruce perceptual en autistas de alto funcionamiento (HFAs) por Timmermans et al. (2011)[41]**. Su objetivo fue determinar el nivel en el cual los HFAs tienen disminuidas las habilidades sociales. Algunos científicos afirman que los autistas tienen problemas con los aspectos automáticos de la cognición social (McIntosh et al., 2006[29]), otros no encuentran alteración en sus facultades cognitivas que involucran la cognición social implícita, como en la representación de acciones (Sebanz et al., 2005[38]) o el aprendizaje implícito (Brown et al., 2010[9]). Mediante el estudio de los HFAs en el paradigma de cruce perceptual, es posible comprobar si las personas autistas tienen dificultades en la coordinación de las interacciones en línea, o su percepción consciente de tales interacciones. Quince parejas de participantes fueron probadas en el paradigma de cruce perceptual; en ocho de estas parejas, un componente de la pareja de interacción era un HFA, en las otras siete no. No hubo diferencias significativas en el comportamiento motor o los patrones de coordinación entre los dos grupos. Por tanto, los HFAs no parecen tener disminuidos los niveles de interacción social que se requieren para la coordinación en el paradigma de cruce perceptual.

- **Investigación en el campo del diseño de Martí (2010)[28]**. Se inspiró en el paradigma de cruce perceptual para construir el robot compañero *Iromec* desarrollado con el propósito de colaborar con niños con diferentes discapacidades. El robot sigue a un niño a una distancia fija tomando la misma trayectoria, ritmo y velocidad que el niño, si otra persona se acerca más al robot, este comienza a seguirle hasta que el niño se acerque de nuevo a él. El autor probó un escenario con un niño de 9 años de edad con una discapacidad cognitiva leve implicando dificultades de atención y retrasos en el aprendizaje. Analizando las grabaciones de vídeo, los maestros del niño estuvieron de acuerdo en que el niño era muy capaz de sostener la actividad a través de una serie de tareas, lo que sugiere que su capacidad de enfocar la atención era mejor de lo habitual. El autor muestra cómo las lecciones aprendidas de la experiencia de cruce perceptual se pueden utilizar en el diseño de artefactos tecnológicos para mejorar la motivación para actuar, la atención a la movilidad, la coordinación y la interacción interpersonal básica.

- **Experimentos de cruce perceptual para investigar las interacciones sociales entre palomas (Ware, 2011[42])**. Utilizó un método similar al de Murray y Trevarthen (1985)[32], un aparato *teleprompter* de doble circuito cerrado permitía a dos pájaros situados en habitaciones diferentes interactuar en tiempo real a través de vídeo. Se observaban comportamientos de cortejo, indicado por el hecho de que las palomas caminaban en círculos. Se comparó el comportamiento de cortejo de 12 palomas (seis machos y seis hembras) cuando veían un vídeo en tiempo real de cada una de sus parejas y cuando se trataba de una grabación anterior con los mismos participantes. Los resultados revelaron como la interacción en vivo beneficia el comportamiento de cortejo. El comportamiento de caminar en círculos de las palomas también se redujo cuando se introducía un retraso en la interacción en tiempo real. No hubo efecto del ángulo de visión. Los resultados de Ware revelan que el comportamiento de cortejo de las palomas no se basa únicamente en las señales visuales, sino que también está influenciado por la sensibilidad a las contingencias sociales existentes entre señales.

Nuestro propósito es desarrollar una plataforma que permita analizar todos los aspectos presentes en un intercambio comunicativo humano. Pretendemos analizar las propiedades que capacitan a los interlocutores para construir patrones de interacción comunes que les hagan regular mutuamente sus acciones. El trabajo realizado proporcionará un *framework* para reproducir resultados ya obtenidos y añadir otros nuevos.

Capítulo 3

Tecnologías utilizadas para implementación de la aplicación

Este capítulo esta dedicado a analizar la tecnología utilizada en la implementación de esta aplicación web. Se pueden distinguir dos partes bien diferenciadas: *front-end* y *back-end*.

3.1. Front-end

El *front-end* es la parte del software que interactúa con los usuarios. En esta aplicación, más en concreto, se refiere al programa cliente que será interpretado por el navegador del usuario, cubriendo el aspecto visual de la página web, respuesta gráfica a eventos provocados por el usuario y todos los mecanismos de recolección y gestión de la información introducida por el usuario al interactuar con la web, de dicha información se selecciona la útil para el servidor y se le envía. Los lenguajes utilizados para la implementación del *front-end* son:

1. HTML5 (*HyperText Markup Language*, versión 5)
2. CSS (*Cascading Style Sheets*)
3. JavaScript

3.1.1. HTML5

HTML5 es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML, se trata de una herramienta potente para la construcción de páginas web. Destacan nuevas directivas respecto a la anterior versión como CANVAS o AUDIO para la manipulación de objetos multimedia y directivas para manejar la Web Semántica (Web 3.0), como son HEADER o FOOTER, estas etiquetas permiten describir cual es el significado del contenido, no tienen impacto importante en la visualización, se orientan a buscadores. Se busca el firme objetivo de separar el contenido del aspecto visual de la web y un renovado énfasis en la importancia del *scripting* DOM (*Document Object Model*) para la gestión del comportamiento de la web.

3.1.2. CSS

CSS es un lenguaje de hojas de estilos usado para describir el aspecto y formato de un documento escrito en lenguaje de marcas. La descripción de estilo puede ser adjuntada en el mismo documento HTML o como un documento separado, este última opción permite reutilizar un mismo estilo para varias páginas. CSS permite personalizar el estilo de las páginas web y subraya la idea aplicada en HTML5 de separar el aspecto del contenido.

3.1.3. JavaScript

JavaScript es un lenguaje de programación interpretado, orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico, que aporta la parte dinámica de la página web, tanto en el apartado gráfico como en el funcional. Se usa principalmente en la parte del cliente pero su uso también está comenzando a extenderse al lado del servidor, ejemplo de ello es el entorno de programación en la capa del servidor Node.js, basado en JavaScript. En la aplicación del proyecto juega un rol importante tanto en la gestión de la interacción del usuario con los objetos gráficos, como en la comunicación entre el cliente y el servidor basada en la tecnología WebSocket, la cual se describe más adelante.

Al avanzar en el desarrollo de la aplicación descubrí la existencia de una biblioteca de JavaScript llamada jQuery, que simplifica el uso de JavaScript y aporta una capa de abstracción que hace que sea compatible con la mayoría de navegadores. Cabe destacar que la aplicación está desarrollada para que funcione de forma correcta en el navegador Mozilla Firefox, de tal manera, que no se garantiza que funcione adecuadamente en otros navegadores. El uso de jQuery en la aplicación del proyecto es parcial, ya que descubrí la librería con la implementación del cliente bastante avanzada, en desarrollos futuros me gustaría completarla, de esta manera facilitaría la portabilidad a otros navegadores.

3.2. Back-end

El *back-end* es la parte del software que procesa la entrada de información enviada desde el *front-end*. En este caso incluye el servidor web y todos aquellos programas necesarios para la recepción de información proveniente del cliente y su tratamiento, con la consiguiente respuesta al cliente o su almacenamiento para un estudio posterior de los datos.

Para entender las decisiones tomadas en el lado del servidor se debe recalcar que el principal problema en el desarrollo de esta aplicación era encontrar una tecnología que nos permitiese establecer una vía de comunicación para transmitir información entre clientes en tiempo real, minimizando latencias y teniendo en cuenta la sincronización. Estudié varias posibilidades y finalmente opté por la tecnología WebSockets.

WebSocket es una tecnología que permite crear un canal de intercambio de información bidireccional y asíncrono entre el servidor y el cliente. Una vez establecida la conexión entre el cliente y el servidor el flujo de datos será libre, no dependiente de peticiones repetitivas de servicio por parte del cliente al servidor (Tecnología Pull). Las principales ventajas de WebSocket son la mejora en la relación entre datos útiles enviados y datos de cabecera frente al caso de HTTP, con una lógica disminución en latencia, es soportado por la mayoría de navegadores y su tasa de mensajes recibidos/enviados por el cliente es muy alta, aún estando el servidor de intermediario. Su desventaja más importante es que no hay implementación directa cliente-cliente de transacción de datos lo que haría disminuir más la latencia. Teniendo en cuenta que los experimentos a realizar se plantean en un marco de red local, la importancia de la latencia que introduce el servidor intermedio en el canal de comunicación, pierde importancia.

El primer paso a seguir para establecer una canal entre el cliente y el servidor, siguiendo el protocolo WebSocket, comienza con el proceso conocido como WebSockets Handshake, el cliente manda una petición de conexión al servidor utilizando el protocolo HTTP (*Hypertext Transfer Protocol*), si el servidor acepta la petición, deja de usarse HTTP para intercambio de datos y pasa a utilizarse WebSockets, con lo que se constituye un canal de comunicación bidireccional y full-duplex sobre un único socket TCP, el cliente y el servidor pueden comenzar a enviarse de forma simultánea y asíncrona información. En la figura 3.1 se observa el proceso descrito.

Para un mayor estudio del protocolo WebSocket se deber acudir al RFC-6455 y para obtener información de la API a http://www.w3.org/TR/2011/WD-websockets-20110419/.

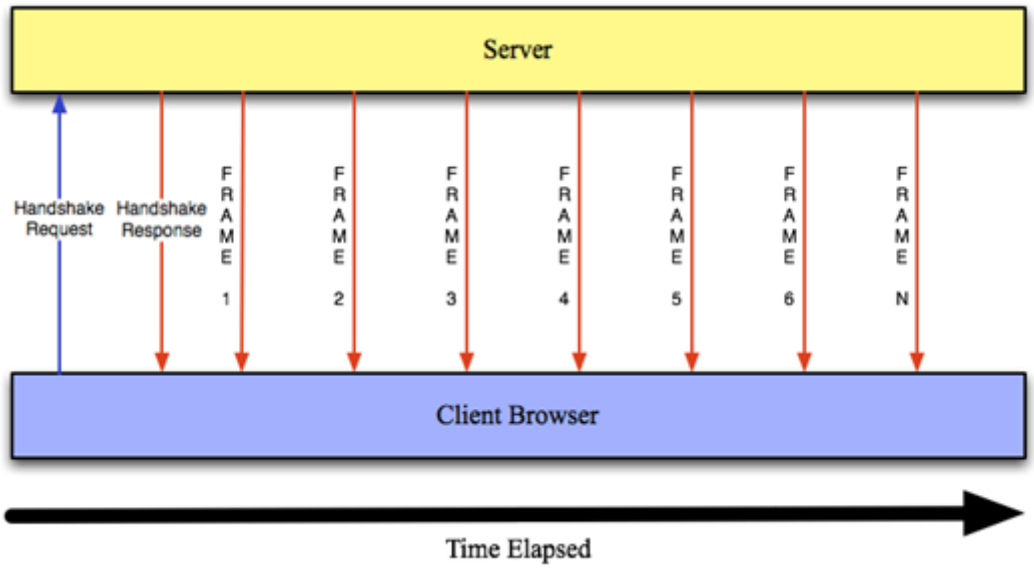

Figura 3.1: Ejemplo de Handshake e intercambio de datos del protocolo WebSocket. [Extraido de http://marakana.com/bookshelf/html5_tutorial/web_sockets.html]

Los datos que se intercambian entre el cliente y el servidor están serializados en formato JSON (*JavaScript Object Notation*). Esta notación es un formato ligero para el intercambio de datos, subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML. JSON representa mejor la estructura de los datos y requiere menos codificación y procesamiento que otros formatos. En la parte del cliente la conversión de texto a objeto es inmediata y en la parte del servidor solo hace falta un objeto traductor que ya esta implementado (Gson).

Una vez elegido el protocolo de comunicación, había que elegir el servidor a utilizar que soportase dicho protocolo. Dentro de las posibles implementaciones que investigué, me decanté por un servidor Tomcat implementado en Java, lenguaje de programación con el que tengo experiencia gracias a asignaturas de la carrera. Para la aplicación utilizo un servidor Tomcat a partir de la versión 3.2.

Para cubrir la comunicación cliente-servidor en tareas secundarias he usado una combinación Java Servlets y JSP (*Java Server Pages*). Un Servlet es un componente Web que se ejecuta dentro de un contenedor web y genera contenido dinámico. JSP es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo, es una manera alternativa, y simplificada, de construir servlets y da la posibilidad de incrustar código java en código HTML.

Para profundizar más en la tecnología utilizada para implementar la aplicación acudir al apéndice A de los anexos.

Capítulo 4

Descripción de la aplicación

En este capítulo se va a describir la aplicación desarrollada para implementar un marco de ejecución virtual para experimentos de cruce perceptual, descritos en el capítulo 2. En la primera sección se describe la aplicación de forma general, en las siguientes secciones se detalla la zona crítica de la aplicación y en la última se describe una aplicación complementaría que permite estudiar mediante gráficas los resultados almacenados durante los experimentos.

4.1. Descripción general

Durante el transcurso de la ejecución de la aplicación los usuarios navegan por una serie de páginas que conforman la aplicación web. En la figura 4.1 se observa el mapa de navegación de la aplicación.

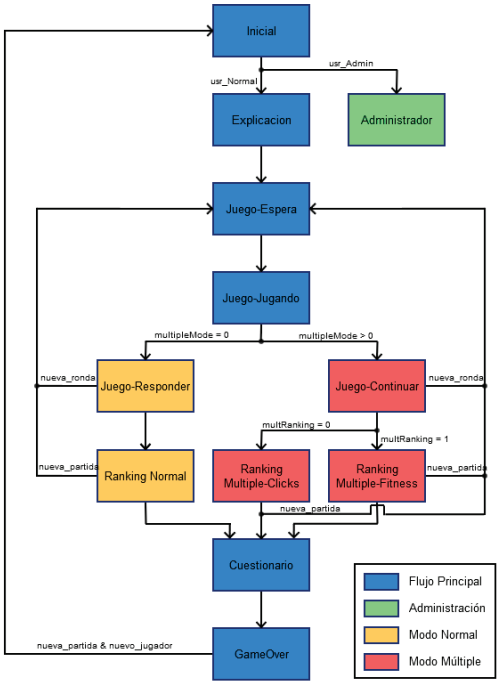

Figura 4.1: Mapa de navegación de la web implementada.

1. Primero se pasa por la página inicial donde el usuario introduce su nombre de jugador. En la figura 4.2 se puede observar la página inicial de la aplicación.

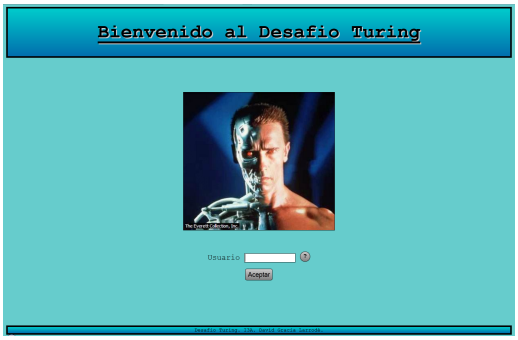

Figura 4.2: Capturas de index.html

2. Dependiendo del nombre de usuario introducido en la anterior página se avanzará a una página u otra.

   a) Si el nombre de usuario es el del administrador se accede a la página de control del comportamiento de la aplicación. Desde esta pantalla se pueden consultar y modificar los datos de configuración de la aplicación almacenados en el fichero `configuration.json`[1] y habilitar/deshabilitar una barrera que impide el paso de los usuarios normales a la página juego.html. En la tabla 4.1 se ve una lista de los parámetros de configuración con su descripción y en la figura 4.3 se muestra una captura de la página.

Parámetro	Descripción
numPairs	Identificador numérico del próximo emparejamiento entre jugadores.
maxPlayers	Número de jugadores (Personas + Bots).
maxBots	Número de bots.
waitTime	Tiempo en milisegundos del retardo del bot DELAYEDSHADOW.
pixelLength	Longitud en píxeles de distancia del bot PIXELSHADOW.
roundMax	Número de rondas del experimento.
maxTime	Tiempo en segundos de cada ronda.
botFirstMode	Indica el tipo inicial de un bot cuando se crea.
winners	Indica cuantos jugadores persona pasan a una segunda vuelta de rondas, si es igual a 0 solo hay una vuelta.
multipleMode	Indica el modo en que se juega la vuelta. Si es igual a 0 modo normal, si es 1 modo múltiple (tiempo) y si es 2 modo múltiple (píxel).
multRanking	En modo múltiple indica el tipo de ranking a generar tras una vuelta completa de rondas.
botModeLoop	Indica si el tipo de los bots cambia a lo largo de la vuelta o siempre es el mismo.
collTimeout	Indica el tiempo máximo en milisegundos en el cual, tras producirse una colisión, si se hace click se considera colisión.

Tabla 4.1: Tabla con los parámetros de configuración y su descripción.

   b) Si es otro nombre, se pasa a la página de explicación, donde se informa al usuario acerca del experimento que va a realizar a continuación. Si el administrador a deshabilitado la barrera y el número de usuarios para el experimento es el definido, se avanzará a la página de juego. En la figura 4.3 se observa la página de explicación.

[1] Para ver la estructura que tiene `configuration.json` consultar la sección D.1 del apéndice D de los anexos.

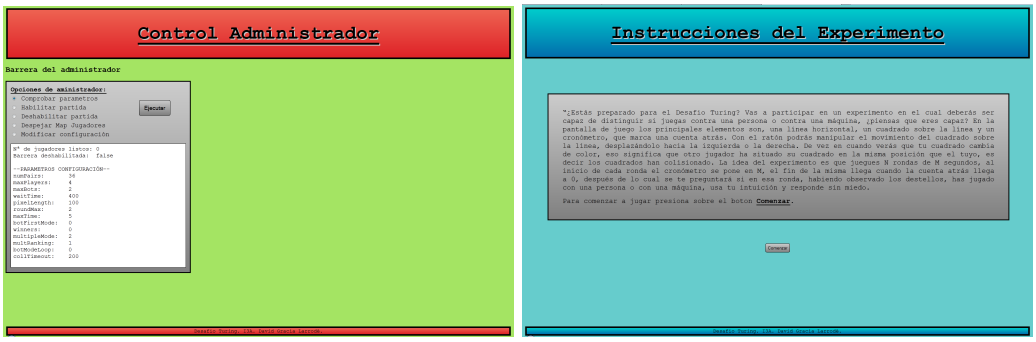

Figura 4.3: Captura de adminBarrier.html (**izquierda**) y de explicacion.html (**derecha**)

3. En la página de juego se desarrollan los experimentos y se recoge la información pertinente de los mismos. La página pasa por distintos estados:

   a) El usuario espera a que todos los jugadores estén listos (Ver figura 4.4).

   b) Se juega la ronda del experimento durante los segundos marcados por el cronómetro

   c) Según el modo de juego, se contestará a un cuestionario o simplemente se continua a la siguiente ronda, volviendo al estado de espera. Cuando se han jugado el número de rondas definidas se avanza a la siguiente página.

4. La siguiente página es la del ranking, un incentivo para que los jugadores se esfuercen durante los experimentos. Según el modo de juego en el que nos encontremos se visualizará un ranking u otro (Ver figura 4.4). El administrador puede programar que se de una segunda vuelta de rondas por lo que se volvería a la página de juego o bien terminar en la primera vuelta y avanzar a la página de cuestionario.

Figura 4.4: Captura de juego.html en espera (**izquierda**) y de ranking.jsp en modo normal (**derecha**)

5. Tras ranking se accede a la página de cuestionario (Ver figura 4.5), cuyo objetivo es recoger información de la experiencia de utilizar la aplicación por parte de los usuarios y de esa manera ajustar tiempos de ronda, corregir posibles errores, etc.

6. La última página es la de despedida (Ver figura 4.5), en ella se muestra un mensaje de fin de partida y se tiene la posibilidad de volver a la página inicial presionado un botón.

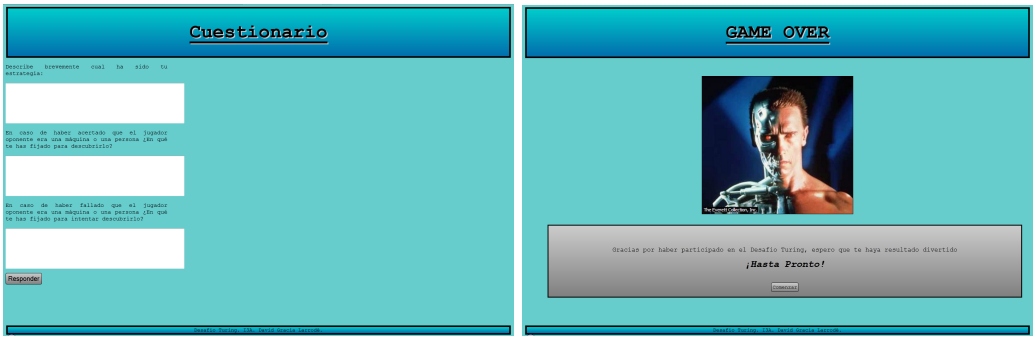

Figura 4.5: Captura de cuestionario.html (**izquierda**) y de game_over.html (**derecha**)

4.2. Descripción elementos de juego.html

La página juego.html tiene dos zonas bien diferenciadas, la zona de juego y la zona de respuesta o paso de ronda.

- La zona de juego tiene los cinco elementos que se pueden ver en la figura 4.6.

  1. Información sobre la ronda en la que nos encontramos.
  2. Explicación de cómo se debe actuar en el momento actual.
  3. Un cronómetro. Marca el tiempo de ejecución de una ronda o experimento.
  4. Según el estado de la ronda, puede aparecer:
     - La imagen de un ratón indicando de forma gráfica el movimiento a realizar durante el juego.
     - Una cuenta atrás, previa a la ronda (3, 2, 1, YA!)
     - Un marcador de colisiones activas.
  5. Una línea de 800 píxeles de longitud y un cuadrado 50 píxeles de lado. La línea representa el espacio unidimensional de experimento y el cuadrado el campo receptor del sujeto.

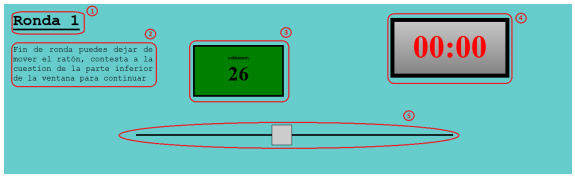

Figura 4.6: Partes de la zona de juego.

- La zona de respuesta o paso a la siguiente ronda no es visible en todo momento, solo se hace visible al terminar de jugar una ronda por completo. Como se observa en la figura 4.7, dependiendo del modo de juego en que nos encontremos aparecerá un contenido u otro.

  - Zona de respuesta a la pregunta de si ha interactuado con una persona o una máquina, según su respuesta aparece un aviso informativo de acierto o fallo.
  - Zona de paso, se muestra un mensaje que informa al usuario de que se han guardado los datos correctamente y que puede continuar presionando el botón correspondiente.

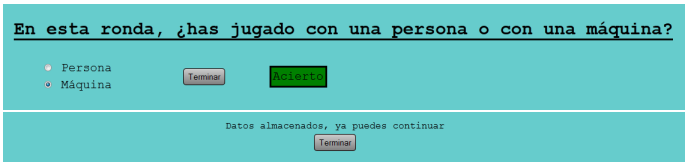

Figura 4.7: Zona de respuesta (**arriba**) y zona de paso (**abajo**)

4.3. Descripción de una ronda y estados de la aplicación cliente

Descripción de una ronda

Una vez están todos los sujetos humanos listos, aparece una cuenta atrás (3, 2, 1, YA!) que marca el comienzo del experimento de cruce perceptual. La pantalla pasa a negro, aparece el marcador de colisiones a 0 y el cronómetro se pone en marcha. A partir de este momento el usuario puede comenzar a mover su cuadrado receptor durante el resto de la ronda. Cuando se produce una colisión con el marcador de un oponente, el cual no es visible, se produce un estímulo visual (el cuadrado del usuario cambia de color fugazmente y el marcador de colisiones se incrementa) y otro auditivo (Efecto Doppler). El sonido se reproduce cuando las superficies de los cuadrados se tocan, alcanzado el máximo volumen de audio en una colisión perfecta y que va disminuyendo como se alejan, en el momento en que dejan de tocarse el sonido desaparece por completo. Una vez termina el tiempo de ronda, el usuario ya no puede mover el cuadrado, en la zona inferior según el modo de juego[2] en que nos encontremos aparece:

- Modo normal: Un cuestionario que el usuario debe responder según piense que ha interactuado con una persona o una máquina. Se confirma la respuesta con un mensaje de acierto o fallo.
- Modo múltiple: Un aviso de fin de ronda y un botón para continuar

Antes de comenzar la siguiente ronda o terminar si no hay más rondas, la información de la ronda actual se almacena en el servidor.

Estados de la aplicación cliente

En la figura 4.8 se puede ver la serie de estados por la que pasa el cliente durante el transcurso de cada ronda y una breve explicación de los mismos.

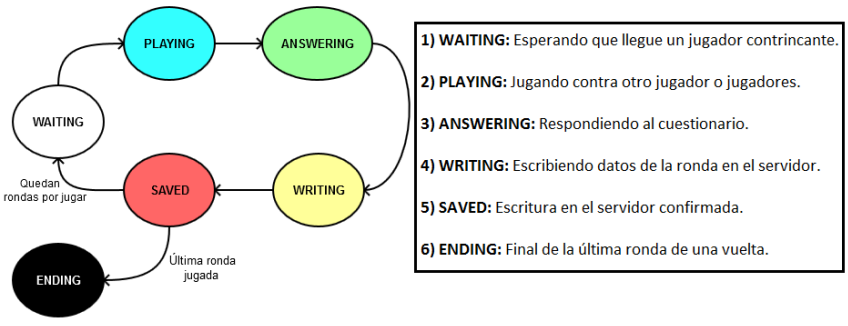

Figura 4.8: Ciclo de estados por los que pasa la aplicación cliente.

[2]La explicación de los modos de juego se encuentra en la sección 4.5 de este capítulo.

4.4. Arquitectura cliente-servidor

La función principal del sistema, está construida sobre una arquitectura de tipo cliente-servidor (WebSockets). En la figura 4.9 puede observarse dos esquemas simplificados a modo de ejemplo de las conexiones y relaciones que se crean en la arquitectura de la aplicación según el modo de juego. Se siguen estos pasos para montar la estructura:

1. Un cliente se conecta al servidor.
2. Hace que se genere un objeto de la clase *GameConnection*, que representa al cliente en el servidor.
3. El servidor, de ser necesario, genera los bots u objetos *BotPlayer*.
4. Estos, a su vez, crean objetos *BotThread* que ejecutan el bucle principal de bot en un hilo de ejecución paralelo para no bloquear la aplicación.

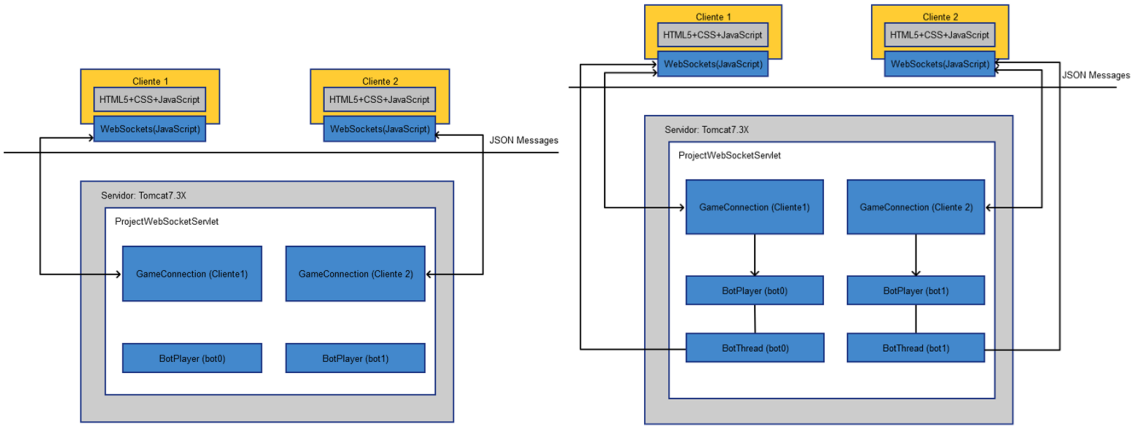

Figura 4.9: Arquitectura ronda Persona vs Persona (**izquierda**) y Arquitectura ronda Persona vs Bot (**derecha**).

4.5. Modos de juego

La aplicación soporta tres modos de juego, que permiten reproducir experimentos originales y crear otros nuevos: normal, múltiple (tiempo) y múltiple (píxel).

1. El modo normal permite representar el experimento de Iizuka y Di Paolo (2007)[25] y el de Iizuka et al.(2009[24],2012a[22]). Una ronda de este tipo es de uno contra uno, bien "persona" vs "persona" o "persona vs bot". Durante el tiempo delimitado el jugador moverá su cuadrado receptor interactuando con el de su adversario. Una vez finalizado ese tiempo el jugador debe responder a la cuestión de si ha interactuado con una máquina o una persona.
2. El modo múltiple con retardo por tiempo se corresponde a un combinación del experimento original de Auvray et al.(2009)[5] añadiéndole la dimensión temporal del de Iizuka y Di Paolo (2007)[25]. En una ronda de este modo, cada sujeto (persona) se enfrenta a otro sujeto (persona), la sombra del contrincante con un retardo en milisegundos y un valor espacial fijo. Durante el transcurso de la ronda el jugador puede mover el cuadrado para interactuar con el resto de jugadores y hacer click para indicar que piensa que la última interacción ha sido con el contrario-persona (*hit*), contra más aciertos mejor. Al finalizar la ronda la información pertinente es almacenada y el jugador puede avanzar a la siguiente ronda o terminar la vuelta.

3. El modo múltiple con retardo por longitud se corresponde con el experimento original de Auvray et al.(2009)[5]. Es igual al anterior modo múltiple salvo que cada sujeto (persona) se enfrenta a otro sujeto (persona), la sombra del contrincante con una distancia en píxeles y un valor espacial fijo distinto al asignado a su contrincante.

4.6. Modos de un bot

Los diferentes tipos de bot implementados para el proyecto son los siguientes:

- **SIMPLE:** Envía mensajes de posicionamiento del cuadrado de forma incremental cada segundo. Se usaba solo en modo normal.
- **RANDOMSIMPLE:** Envía mensajes de posicionamiento con un valor aleatorio entre 0 y 800 cada medio segundo. Se usa solo en modo normal.
- **SHADOW:** Repite la traza de posiciones de un sujeto persona jugada en una ronda anterior. Se usa solo en modo normal.
- **DELAYEDSHADOW:** Repite la traza de posiciones de la persona con la que se está jugando la ronda actual, con un retraso en milisegundos. Se usa en modo normal y múltiple con retraso por tiempo.
- **COMPLEX:** Por defecto cuando el bot recibe mensajes del jugador persona se acerca a su posición, mientras no recibe mensajes se aleja. Las políticas de acercamiento o alejamiento pueden ser reprogramadas como interese. Se usa solo en modo normal.
- **PIXELSHADOW:** Retransmite la traza de posiciones de la persona con la que se está jugando la ronda actual, menos una distancia en píxeles. Se usa solo en modo múltiple con retraso por distancia en píxeles.

4.7. Errores tratados gráficamente

Se controlan una serie de posibles eventualidades que pueden terminar en error. Si se llega a producir un fallo, se muestra al usuario por pantalla una información descriptiva del problema y, si existe, una forma de recuperarse.

- En caso de una caída del servidor mientras se está en la pantalla de juego se muestra el error de la captura superior de la figura 4.10, reescribiendo el código HTML de la misma página.
- Si se produce el intento de conexión con el servidor de un cliente con el mismo identificador que otro ya conectado, se rechaza. En la captura central de la figura 4.10 puede observarse el mensaje que se mostraría por pantalla.
- Si mientras dos jugadores humanos enfrentados juegan su ronda, uno de ellos pierde la conexión sin haber terminado la cuenta atrás del cronómetro, al jugador todavía conectado se le informa del suceso y se le da la oportunidad de repetir la ronda, como se muestra en la captura inferior de la figura 4.10.

Para una información ampliada de la aplicación principal consultar el apéndice B, en especial la subsección B.1.6. También resulta interesante las tablas comparativas del apéndice F.

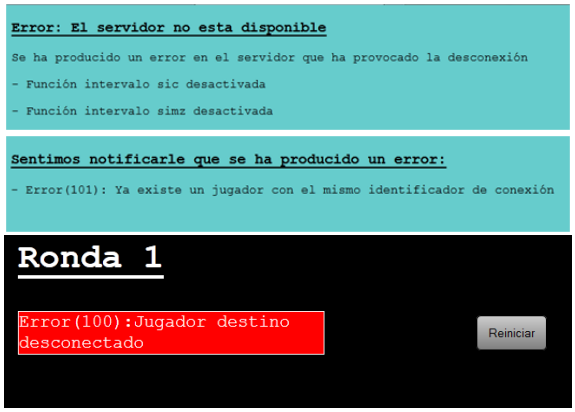

Figura 4.10: Error producido al caerse el servidor (**arriba**). Error producido al intentar conectarse con un identificador de conexión existente (**centro**). Error producido al caerse el jugador contrario (**abajo**).

4.8. Visor de gráficas

Como complemento a la aplicación principal del proyecto he desarrollado un conjunto de visores de gráficas que permiten analizar los datos recogidos durante los experimentos. Para construir estos interfaces he utilizado la herramienta software EJS (Easy Java Simulations) diseñada para la creación de simulaciones discretas por computador.

A partir de los elementos gráficos ofrecidos por el programa para confeccionar interfaces de usuario, he construido los más apropiados para el estudio de resultados de los tres modos implementados en este proyecto. Además EJS permite importar tus propias clases Java y añadir código complementario implementado por uno mismo, y de esa manera acceder a la información deseada.

En la figura 4.11 puede verse el visor construido para el estudio de trazas almacenadas durante el modo normal de juego, con una gráfica cargada.

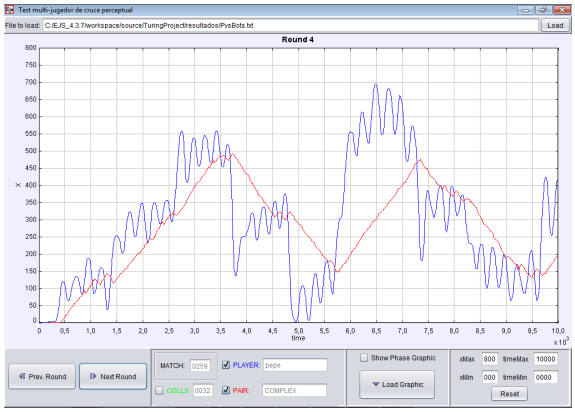

Figura 4.11: Visor gráficas modo normal

Para leer una descripción más detallada de los visores y observar ejemplos de gráficas acudir al apéndice C de los anexos.

Capítulo 5

Resultados

En este capítulo se recogen todos aquellos aspectos del proyecto que tienen que ver con la recopilación de información útil acumulada durante el transcurso del mismo, eso incluye información de rondas, cuestionarios de uso y toma de medidas de los diferentes modos de trabajo. Tiene cinco secciones bien diferenciadas: en la primera de ellas se describe la información que se almacena de cada ronda, teniendo en cuenta los diferentes modos de juego y la estructura de la información de uso de la aplicación recogida; en la segunda una descripción de los experimentos realizados para toma de medidas de prestaciones; en la tercera una descripción más en concreto de los datos del modo normal; en la cuarta del modo múltiple tanto con retardo en milisegundos, como con distancia en píxeles; en la quinta una comparativa temporal con el experimento de Iizuka y Di Paolo (2007)[25].

5.1. Información almacenada

Después de jugar una ronda, cada jugador envía al servidor un mensaje con información recogida en dicha ronda. El servidor va almacenado los diferentes mensajes de una vuelta completa en un fichero de texto, que estará formado por líneas de esos mensajes, el nombre de este fichero está formado por el nombre del jugador del que se recoge información concatenado con la fecha y hora en el instante de conectarse al servidor, además en modo múltiple tiene el prefijo "MT", en el caso de retardo en milisegundos, o "MP", en el caso de distancia en píxeles.

Los principales campos de información que se guardan identifican la ronda jugada, parámetros de la misma, al jugador del que se guarda la información, la traza de sus movimientos y al oponente u oponentes. Según el modo de juego en el que se desarrolla una ronda se almacenará un tipo de mensaje de información de ronda u otro, en modo normal será de tipo *roundInfo* y en modo múltiple *roundMMInfo*.[1]

La principal diferencia entre ambas estructuras es que en modo normal la partida es de uno contra uno, bien entre dos personas o entre una persona y un bot, mientras que en modo múltiple cada jugador persona se enfrenta a tres oponentes, una persona, su sombra, bien con retardo en milisegundos, bien con distancia en píxeles y un valor fijo, por ello difieren en los campos relacionados con este hecho. Además, en el caso normal aparecen campos relacionados con la pregunta final que el múltiple no posee, por el contrario en el modo múltiple aparece un campo en relación con los clicks realizados por el jugador que no contiene el modo normal. La principal diferencia entre los modos múltiples es el tipo de bot sombra, pudiendo ser DELAYEDSHADOW o PIXELSHADOW[2].

[1] Para más información de este tipo de mensajes consultar la sección D.5 del apéndice D de los anexos.

[2] Para ver una descripción de DELAYEDSHADOW y PIXELSHADOW acudir a la subsecciónB.1.5 del apéndice B.

Durante el desarrollo de la aplicación se han cambiado campos de estas trazas para adecuarlos a nuevas funcionalidades que han aparecido, el caso más destacado es el de inclusión de información necesaria para poder reconstruir las trazas de los jugadores oponentes. Esta información es utilizada por los visores creados con EJS (Easy Java Simulations)[3], introducidos en la sección 4.8 del capítulo 4, para construir, de forma gráfica, las trazas de sus movimientos.

Una de las páginas que componen la aplicación es "`cuestionario.html`", esta recoge información de los usuarios tras completar por completo el experimento programado. Tiene tres preguntas: escribe brevemente cual ha sido tu estrategia, en caso de haber acertado que el jugador oponente era una máquina o una persona ¿En qué te has fijado para descubrirlo? y en caso de haber fallado que el jugador oponente era una máquina o una persona ¿En qué te has fijado para intentar descubrirlo?, esto permite tener *feedback* con los usuarios, observar estrategias que siguen, modificar los parámetros de la aplicación que sean necesarios para conseguir una mejor experiencia en los experimentos, etc. Esta información se guarda en ficheros de texto, cuyo nombre esta compuesto por una "C" concatenada con el identificador de conexión del usuario y la fecha y la hora de conexión con el servidor.

5.2. Descripción medición de prestaciones

Para medir las prestaciones de la aplicación se realizó una serie de pruebas sobre la misma planteando diferentes escenarios de juego[4]. Se insertó instrumentación en el código para que se pudiera obtener tras una ronda el número de mensajes recibidos y enviados y un tiempo de ronda. Los datos medidos siempre hacen referencia a la partida y trazas de los jugadores persona. Luego en modo *off-line* con un programa habilitado al caso se hizo un análisis de las trazas de las partidas, midiendo las diferencias de tiempo entre un mensaje enviado y el siguiente, para calcular la velocidad máxima media de envío, comprobando la sensibilidad del escuchador de captura de movimiento del ratón, ya que, por cada captura se envía un mensaje.

Cada prueba realizada constaba de 10 rondas de 10 segundos cada una. Durante una partida se seguía una estrategia por la cual se movía el ratón de un lado a otro los más rápido posible, además de hacer clicks en modo múltiple, para que los datos recogidos fueran lo más fiables posible a la hora de medir los límites de la tecnología utilizada para implementar la aplicación. En modo normal se realizó un experimento persona contra bot, por cada tipo de bot y uno entre dos personas, en el caso del bot DELAYEDSHADOW, 5 pruebas, cambiando el valor del retardo. En modo múltiple, para el caso de retardo en tiempo, se hicieron 5 experimentos variando el retraso en milisegundos de la sombra. Para el caso de distancia en píxeles solo una prueba, ya que cambiar el valor de la distancia no altera los tiempos tomados como en el caso de retardo por tiempo.

De cada experimento se obtuvo 3 datos de interés: la media de mensajes enviados por segundo, la media de mensajes recibidos por segundo y la diferencia media total entre mensajes enviados. Durante la toma de medidas, se observó que la primera ronda de todos los experimentos tiene una tasa de mensajes enviados mucho mayor que el resto de rondas, achacable a que en el cliente en primera instancia no hay memoria reservada para las estructuras de datos que almacenan la información de una ronda y demás datos auxiliares, que se van creando en esta primera partida. Por ello, se consideró el primer dato de cada experimento como un dato atípico que podría mejorar las prestaciones temporales de la aplicación notablemente, no se ha incluido en las medias calculadas para que los datos sean más fidedignos y se centren en el grueso de información más representativa, ya que en las siguientes rondas de los experimentos los tiempos no difieren tanto unos de otros.

[3]Para leer una descripción más amplia de los visores implementados y ejemplos de las gráficas ir al apéndice C

[4]Para ver en que consisten los experimentos con más detalle, acudir a la sección E.1 del apéndice E, donde puede verse un resumen esquematizado de los mismos.

Para una descripción más en detalle de la medición de prestaciones acudir a las secciones E.2, E.3 y E.4 del apéndice E de los anexos.

5.3. Experimento modo normal

En esta sección se describe la información recogida en este modo de juego. En primer lugar la obtenida por las visitas realizadas que pudieron probar la aplicación y en segundo la acumulada por la medición de prestaciones descrita en la sección 5.2.

5.3.1. Información de las visitas

En febrero de este año se realizaron una serie de visitas, al campus Río Ebro, por parte de alumnos de educación secundaria que pudieron probar durante parte de su visita el modo normal de la versión de la aplicación desarrollada hasta ese momento. Estas visitas ayudaron mucho a la hora de testear la aplicación, se corrigieron errores según se iban planteando, se añadieron campos a la traza de datos que se almacenaba, se implemento la página de administrador, para tener control de la aplicación sin modificar manualmente el fichero `configuration.json`[5] y de los propios jugadores creando una barrera de acceso a la página `juego.html` y surgió la idea del ranking y de una cuenta atrás previa a cada ronda para hacer más atractiva la aplicación. Se creó la página de cuestionario para poder recoger información de primera mano del uso de la aplicación que los usuarios, conocer sus estrategias, si el número de rondas y el tiempo de duración de cada una era el correcto, etc.

Algunas de las estrategias que parecieron interesantes fueron:

- Quedarse quieto y analizar la rapidez y ritmo del sonido de las colisiones.
- Observar el número de colisiones y si tenían lugar en distintos sitios. Si había poca colisiones en distinto lugar se trata de una máquina.
- Ir despacio y observar si los cambios son bruscos, en ese caso se trata de una máquina.
- Si ante mis movimientos el oponente cambia su comportamiento, es una persona.

Para ver que bots[6] engañaban más a los jugadores se analizo las trazas obtenidas. Como puede observarse en la tabla 5.1, si bien no es muy grande el número de rondas estudiadas, se puede distinguir que hay tipos de bot más fáciles de detectar, como es el caso de los SHADOW, de lo que se puede deducir lo importante que resulta la sincronización entre jugadores y los cambios originados por comportamientos del oponente, ya que aunque se trata de la reproducción de la traza de una persona, al no estar sincronizada con el jugador persona de la ronda actual, su comportamiento no se adapta a los cambios, es siempre el mismo. A raíz de estos resultados y hablar con los usuarios se cambiaron las políticas del modo COMPLEX, para pasar de acercarse al receptor del jugador humano tanto cuando se movía como cuando se estaba quieto, a acercarse solo cuando se mueva y alejarse cuando este quieto.

	RANDOMSIMPLE	SHADOW	DELAYEDSHADOW	COMPLEX
% Aciertos	64,10	73,68	50	53,85
% Fallos	35,90	26,32	50	46,17

Tabla 5.1: Tabla porcentajes de acierto y fallo según el tipo de bot.

[5]Para ver la estructura que tiene `configuration.json` consultar la sección D.1 del apéndice D de los anexos.
[6]Para obtener información del comportamiento de cada bot acudir a la subsección B.1.5 del apéndice B de los anexos.

5.3.2. Información de la medición de prestaciones

En modo normal la media de mensajes enviados por segundo mínima es de 53,0603 y la máxima 60,7565, lo que nos hace movernos entre 17,84 y 16,46 milisegundos por mensaje enviado. En el caso de la media de mensajes recibidos por segundo hay una máxima de 88,7562 en el caso del bot SHADOW y una mínima de 1,9983 en el bot RANDOMSIMPLE, estos datos no son realmente significativos, ambos modos tienen una traza constante, solo envían mensajes, no los reciben, siempre van a darse en las mismas cantidades salvo latencia añadida por el medio de comunicación, además el bot SHADOW tiene la traza de una primera ronda por eso ese valor tan alto. Si observamos el resto de experimentos, no constantes, se tiene una máxima de 60,4881 y una mínima de 45,1400. Para el tercer parámetro en estudio, la diferencia media entre mensajes enviados, tiene un máximo de 19,0702 milisegundos entre mensaje y mensaje enviado y un mínimo de 16,5686 milisegundos entre mensajes enviados, como se observa, son valores similares a los obtenidos a través de la media de mensajes enviados por segundo, dos formas de calcular la velocidad de envío de la aplicación.

En el experimento 4 y 4b, quería comprobar si añadir el guardado de la traza del bot COMPLEX, durante una ronda, para luego poder representar esta traza en el visor de gráficas correspondiente, merecía la pena y no añadía una latencia significativa. Como se puede comprobar en los datos de la figura 5.1 los valores obtenidos son semejantes, incluso algo mejores en el caso 4b, lo que justifico con un mejor comportamiento del servidor y/o del jugador persona. En la subsección E.5.1 del apéndice E se añade la tabla de medias conjunta de los dos jugadores persona del experimento 5.

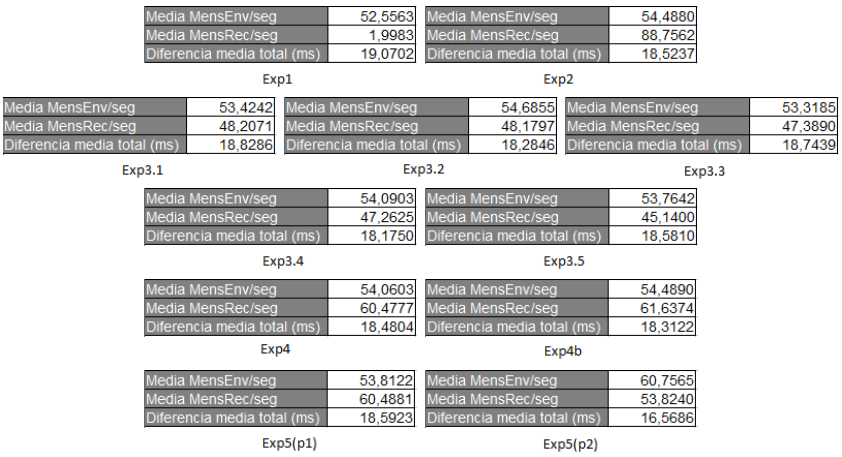

Figura 5.1: Medidas obtenidas en los experimentos 1-5 en modo normal.

Uno de los casos más interesantes dentro de este modo es el tercer experimento, en el bot DELAYEDSHADOW, aparecen las mayores latencias, tanto por el hecho de que el servidor espera la traza del jugador persona, para reenviarla con un retraso marcado, como por el propio retraso generado en el servidor a través de *sleeps*. Como se ha observado antes la máxima diferencia entre mensajes enviados es, redondeando, de unos 20 milisegundos, lo que necesitamos para completar el ciclo de ida, de cliente al servidor, y vuelta, del servidor al cliente, es este segundo camino. Como se observa en la figura 5.2 se muestra una tabla con la media de mensajes recibidos entre el tiempo real. Llamo tiempo real al tiempo total de la ronda menos el retraso correspondiente del experimento. Se obtiene una media de 48,6905 mensajes recibidos por segundo, por lo que se están recibiendo mensajes del servidor cada 20,5379 milisegundos. Sumando este tiempo con el anterior obtenemos un parámetro clave de la aplicación, lo que le cuesta a un mensaje enviado desde un cliente fuente llegar a un cliente destino, es decir, los mensajes llegan al receptor con una desincronización máxima de 40-50 milisegundos.

	Media MensRec/TReal
EXP3.1 – waitTime = 100ms	48,6935
EXP3.2 – waitTime = 200ms	49,1621
EXP3.3 – waitTime = 300ms	48,8536
EXP3.4 – waitTime = 400ms	49,2301
EXP3.5 – waitTime = 500ms	47,5130
Media Total	48,6905

msegMensRec	20,5379

Figura 5.2: Tabla de mensajes recibidos en tiempo real del experimento 3.

5.4. Experimento modo múltiple

En esta sección se describe la información recogida con respecto al modo de juego múltiple, tanto con retardo en milisegundos, como con distancia en píxeles. Este modo no ha sido probado durante las visitas, ya que no estaba implementado en las fechas en las que se produjeron. La información obtenida, por tanto viene de la medición de prestaciones descrita en la sección 5.2 de este capítulo.

5.4.1. Información de la medición de prestaciones

Como puede observarse en las tablas una de las diferencias entre el modo normal y el múltiple, es que el volumen de mensajes recibidos por segundo es prácticamente el doble, debido al hecho de que cada jugador humano esta recibiendo dos flujos de datos pertenecientes a un jugador persona y un bot sombra. Se tiene un máximo de 62,9977 mensajes enviados por segundo y un mínimo de 32,7985, por lo que nos movemos entre los 15,87 y los 30,49 milisegundos por mensaje enviado. El mínimo anterior lo achaco a un problema con la superficie por la que se movía el ratón, que ofrecía más resistencia que en anteriores pruebas. En cuanto al número medio de mensajes recibidos por segundo, hay un máximo de 119,8763 y un mínimo de 92,5227, como he indicado antes casi se duplican los datos del modo normal. El tiempo entre mensajes enviados se mueve entre 16,0804 y 31,1188 milisegundos, este segundo dato coincide con el peor caso de mensajes enviados por segundo, motivado por la misma causa. En la subsección E.5.2 del apéndice E se añaden la tablas de medias conjunta de los dos jugadores persona de los experimento 6 y 7 respectivamente.

Media MensEnv/seg	56,0645
Media MensRec/seg	107,6231
Diferencia media total (ms)	17,9398

Exp6.1(p1)

Media MensEnv/seg	56,4494
Media MensRec/seg	107,6667
Diferencia media total (ms)	18,3906

Exp6.1(p2)

Media MensEnv/seg	54,5050
Media MensRec/seg	119,8763
Diferencia media total (ms)	18,4454

Exp6.2(p1)

Media MensEnv/seg	62,9977
Media MensRec/seg	104,9429
Diferencia media total (ms)	16,0804

Exp6.2(p2)

Media MensEnv/seg	56,4058
Media MensRec/seg	62,4218
Diferencia media total (ms)	17,4133

Exp6.3(p1)

Media MensEnv/seg	32,7985
Media MensRec/seg	108,3031
Diferencia media total (ms)	31,1188

Exp6.3(p2)

Media MensEnv/seg	52,0170
Media MensRec/seg	109,7044
Diferencia media total (ms)	19,3055

Exp6.4(p1)

Media MensEnv/seg	58,5046
Media MensRec/seg	99,3032
Diferencia media total (ms)	17,3130

Exp6.4(p2)

Media MensEnv/seg	48,7137
Media MensRec/seg	111,0951
Diferencia media total (ms)	21,1655

Exp6.5(p1)

Media MensEnv/seg	59,1329
Media MensRec/seg	92,5227
Diferencia media total (ms)	16,9424

Exp6.5(p2)

Media MensEnv/seg	52,6362
Media MensRec/seg	105,4029
Diferencia media total (ms)	19,1043

Exp7(p1)

Media MensEnv/seg	53,2934
Media MensRec/seg	105,1486
Diferencia media total (ms)	18,8959

Exp7(p2)

Figura 5.3: Medidas obtenidas en los experimentos 6 y 7 en modo múltiple.

Al igual que en el modo normal el caso más interesante dentro del modo múltiple es el sexto experimento, ya que dentro de los componentes del experimento hay dos bots sombra de tipo DELAYEDSHADOW, por lo que aparecen las mayores latencias. La máxima diferencia entre mensajes enviados es de unos 30 milisegundos, para obtener el tiempo de la vuelta he realizado una serie cálculos cuyo resultado puede verse en la figura 5.4, en ella se muestra una tabla con la media de mensajes recibidos entre el tiempo real por experimento. Se obtiene una media de 105,4876 mensajes recibidos por segundo, por lo que se están recibiendo mensajes del servidor cada 9,4798 milisegundos, hay que tener en cuenta que se incluyen los mensajes del oponente persona y del bot sombra. Como la cantidad de mensajes enviados por el bot sombra siempre va a ser inferior al del jugador persona, del flujo de mensajes que llegan al jugador destino representarán algo menos de la mitad, cogiendo la mitad de la media anterior obtendríamos un tiempo entre mensajes recibidos provenientes del bot sombra de 18,96 milisegundos, redondeando unos 20 milisegundos. Sumando tiempos, los mensajes que le llegan a un jugador desde otro llevarán como máximo una desincronización de 50 milisegundos.

	Media MensRec/TReal
EXP6.1 – waitTime = 100ms	108,7298
EXP6.2 – waitTime = 200ms	114,7001
EXP6.3 – waitTime = 300ms	87,9966
EXP6.4 – waitTime = 400ms	108,8528
EXP6.5 – waitTime = 500ms	107,1586
Media Total	105,4876
msegMensRec	9,4798

Figura 5.4: Tabla de mensajes recibidos en tiempo real del experimento 6.

5.5. Tiempos aplicación vs Experimento (Iizuka y Di Paolo, 2007[25])

Como se ha descrito en las secciones anteriores un jugador detecta el movimiento del cuadrado de otro jugador con una desincronización máxima de 50 milisegundos. Si observamos la gráfica de la figura 5.5, que hace referencia al experimento realizado por Iizuka y Di Paolo (2007)[25], vemos la diferencia entre las dos trayectorias de los agentes para el caso de interacciones en vivo (línea continua) y para el caso de interacciones con una grabación (línea discontinua). En principio ambas trazas no difieren mucho, la diferencia entre trayectorias no supera las 100 unidades de longitud, pero a partir de 400 milisegundos de desfase la diferencia para el caso unilateral aumenta considerablemente perdiendo la coordinación. En el caso de la aplicación no añade un desfase superior a 50 milisegundos por lo que se podrá llegar a la coordinación de interacciones sin problema.

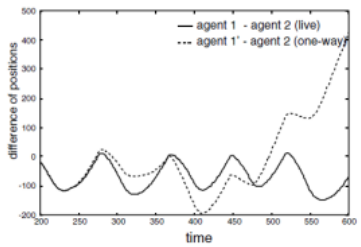

Figura 5.5: Diferencia entre las dos trayectorias en cada condición. Cuando la línea cruza 0 significa que dos agentes se cruzan entre sí.

Capítulo 6

Conclusiones

6.1. Objetivos alcanzados

El objetivo de este proyecto era conseguir un marco de ejecución virtual para realizar experimentos de cruce perceptual. Para la consecución de dicho objetivo se han realizado las siguientes tareas:

1. En primer lugar se ha construido un *framewok* simple para realizar experimentos de cruce perceptual, en este caso los experimentos están formados únicamente por dos personas, una contra otra. Esta primera construcción ha servido de base a las implementaciones que se sucedieron de forma posterior.

2. En segundo lugar se implementó un *framewok* partiendo del anterior que permite experimentos persona contra persona y persona contra bot. En este marco se sigue teniendo un integrante por lado del experimento, pero se añade un nuevo tipo de participante, el bot. Al final de cada experimento cada persona responde si ha interactuado con otra persona o con una máquina. Se crearon los siguientes tipos de bots: RANDOMSIMPLE (envían posiciones aleatorias de forma continua); SHADOW (representan la traza de movimientos de un experimento anterior); DELAYEDSHADOW (repiten los movimientos que este haciendo su oponente humano con un cierto retardo en tiempo); COMPLEX (siguen una política de movimiento según la posición del participante humano). Este marco conforma el modo de juego normal de la aplicación.

3. En tercer lugar se mejoró el *framewok* anterior permitiendo realizar experimentos con más de un integrante por lado de los mismos. Para ser más exactos cada participante humano se enfrenta a otro participante humano, a un objeto móvil y a un objeto fijo. Durante el experimento la persona podrá hacer click cuando crea que esta interactuando con otra persona. El objeto móvil, puede ser de dos tipos: un bot sombra del oponente humano con un cierto retardo en tiempo o con una diferencia de distancia impuesta. Esta implementación conforma el modo múltiple con retardo por tiempo y con retardo por distancia, respectivamente.

4. Se ha implementado un visor de gráficas que permite estudiar los experimentos a partir de la información almacenada de los mismos en formato JSON. El interfaz se ha desarrollado utilizando Easy Java Simulations.

6.2. Experimentos que se representan

Cada modo de juego implementado se inspira en uno de los experimentos originales o en una combinación de los mismos. El modo normal permite representar el experimento de Iizuka y Di Paolo (2007)[25] y el de Iizuka et al.(2009[24],2012a[22]). El modo múltiple con retardo por longitud se corresponde con el experimento original de Auvray et al.(2009)[5]. El modo múltiple con retardo por tiempo se corresponde a un combinación del experimento original de Auvray et al.(2009) añadiéndole la dimensión temporal del de Iizuka y Di Paolo (2007)[25].

La medición de prestaciones realizada a la aplicación, a parte de confirmar el buen funcionamiento de la tecnología WebSockets, demuestra que un jugador detecta el movimiento del cuadrado de otro jugador con una desicronización máxima de 50 milisegundos. Al comparar con el experimento de Iizuka y Di Paolo[25], descrito en la sección 2.3 del capítulo 2, se observa que en el original con una distorsión superior a 400 milisegundos se pierde la dinámica de cruce perceptual, por lo que se deduce que con la aplicación desarrollada se podrá llegar a la coordinación de interacciones sin problema.

6.3. Trabajo futuro

En esta sección listo las mejoras que me gustaría aplicar en el futuro a la aplicación y posibles líneas de trabajo a seguir. En primer lugar describo las mejoras, que desde mi punto de vista, harían más completa a la aplicación y en segundo lugar los proyectos que han ido surgiendo durante el proceso de implementación de la aplicación y que partirían del estado actual de la misma.

6.3.1. Posibles mejoras de la aplicación

Hay dos mejoras que me gustaría aplicar al proyecto:

- La primera de ellas consistiría en hacer que la aplicación pueda ser ejecutada en todos los navegadores. Actualmente funciona de forma correcta en Mozilla Firefox. Para ello utilizaría la librería jQuery de JavaScript, que simplifica el uso de JavaScript y es compatible con la mayoría de los navegadores actuales.

- La segunda sería una modificación en la administración de la aplicación. En este momento para poder acceder a la página de administración solo es posible si la aplicación se ejecuta en el equipo donde reside el servidor y además se debe introducir el nombre de usuario correspondiente al administrador. Lo que me gustaría hacer es hacer la administración más flexible y segura, que se pudiera acceder a la página de servidor desde cualquier equipo, en principio dentro de la LAN, y que la identificación del usuario sea mediante usuario y contraseña.

6.3.2. Líneas de trabajo futuras

- En el ámbito de la inteligencia artificial, desarrollar un bot para el modo de juego normal de la aplicación. Este bot sería manipulado por una red neuronal, para ser más exactos una CTRNN (*Continuous Time Recurrent Neural Network*). Una CTRNN es un sistema de ecuaciones diferenciales ordinarias que modela los efectos en una neurona por la entrada de un tren de impulsos. Las CTRNNs son computacionalmente más eficientes que simular directamente cada impulso en una red ya que no modelan activaciones a este nivel de detalle. En un entorno controlado se utilizará un algoritmo evolutivo y una ecuación de *fitness* para llegar a obtener la CTRNN más adecuada. Habrá que hacer modificaciones en el cliente para que el flujo de datos que llegue al bot sea continuo. Finalmente el bot manipulado por la CTRNN será añadido al servidor para que se pueda jugar contra él.

- Una muestra de la aplicación de cruce perceptual ha interesado a miembros del departamento de psicología de la universidad de las Islas Baleares (UIB) que estudian patologías como la sociopatía. La sociopatía, también conocida como trastorno de personalidad antisocial (TPA), es una patología que provoca que las personas que la padecen pierdan la noción de la importancia de las normas sociales, como son las leyes y los derechos individuales. Los sociópatas son personas que padecen un mal de índole psiquiátrico, un grave cuadro de personalidad antisocial que les hace rehuir a las normas preestablecidas; no saben o no pueden adaptarse a ellas. La aplicación serviría como método de *screening* o filtrado, de tal manera que analizando los parámetros de las trazas obtenidas se pudiera encontrar patrones que permitieran identificar indicios de sociopatía de individuos en estudio. Los tests actuales pueden ser engañados por los sujetos de prueba que responden lo que se espera de una persona sana, ya que aunque no les importa, son conscientes de lo que esta mal y lo que esta bien. En el caso de la aplicación de cruce perceptual, que lleva a la mínima expresión la interacción entre personas, esto no ocurre ya que no hay manera de distinguir a priori entre personas y bots, lo que evitaría estrategias de engaño al experimento.
- Al igual que en el anterior caso, tras una muestra de la aplicación de cruce perceptual, existe otro interés en el proyecto. Se piensa que puede ser una manera de estudiar los mecanismos de afiliación entre primates. La afiliación es un mecanismo de defensa en que el individuo acude a los demás en busca de ayuda o apoyo, lo que significa compartir los problemas sin tratar de atribuirlos a los demás. Con la ayuda pulseras lectoras de intensidad se podrá distinguir cuando los primates piensan que están interactuando entre iguales, ya que no pueden indicarlo de forma convencional.

6.4. Valoración personal y problemas encontrados

Este proyecto comenzó como un trabajo de prácticas que terminó derivando en el actual proyecto de fin de carrera. Con este origen, eminentemente práctico, la elección de uso de metodologías ágiles, frente al paradigma clásico, para el desarrollo del proyecto ha sido más que conveniente. El proyecto ha permitido abordar el trabajo, tanto desde un punto de vista ingenieril, como científico, debido al contexto en que se mueve la aplicación. El aprendizaje de una nueva forma de construir interfaces usando tecnologías web (HTML5, CSS, JavaScript) ha resultado enriquecedor y ha sido interesante aprender como funciona la tecnología WebSocket para resolver el problema de las comunicaciones en tiempo real. Tener la posibilidad de probar la aplicación durante un periodo de visitas, ayudó a resolver errores y buscar diferentes caminos para superar los problemas en el desarrollo que, de otra manera, hubieran sido díffíciles de encontrar.

Los problemas más importantes que se han encontrado a lo largo de la realización del proyecto han sido: montar la estructura WebSockets de forma correcta, ya que la documentación encontrada resultaba vaga en algunos aspectos; identificar las variables y funciones de posible acceso concurrente durante el transcurso de una ronda, este problema se detecto durante las pruebas realizadas durante las visitas, gracias a ello se pudo solventar.

Parte II

Anexos

Anexo A

Tecnología utilizada para el desarrollo de la aplicación

A.1. Conocimientos adquiridos y tecnología usada en la aplicación

A la hora de analizar la tecnología utilizada en la implementación de esta aplicación web se pueden distinguir dos partes bien diferenciadas: *front-end* y *back-end*.

A.1.1. Front-end

El *front-end* es la parte del software que interactúa con los usuarios. En esta aplicación, más en concreto, se refiere al programa cliente que será interpretado por el navegador del usuario, cubriendo el aspecto visual de la página web, respuesta gráfica a eventos provocados por el usuario y todos los mecanismos de recolección y gestión de la información introducida por el usuario al interactuar con la web, de dicha información se selecciona la útil para el servidor y se le envía.

Los lenguajes utilizados para la implementación del *front-end* son:

1. HTML5 (*HyperText Markup Language*, versión 5)
2. CSS (*Cascading Style Sheets*)
3. JavaScript

HTML5

HTML5 es la quinta revisión importante del lenguaje básico de la *World Wide Web*, HTML, se trata de una herramienta potente para la construcción de páginas web. Destacan nuevas directivas respecto a la anterior versión como CANVAS o AUDIO para la manipulación de objetos multimedia y directivas para manejar la Web Semántica (Web 3.0), como son HEADER o FOOTER, estas etiquetas permiten describir cual es el significado del contenido, no tienen impacto importante en la visualización, se orientan a buscadores. Se busca el firme objetivo de separar el contenido del aspecto visual de la web y un renovado énfasis en la importancia del *scripting* DOM (*Document Object Model*) para la gestión del comportamiento de la web.

CSS

CSS es un lenguaje de hojas de estilos usado para describir el aspecto y formato de un documento escrito en lenguaje de marcas. La descripción de estilo puede ser adjuntada en el mismo documento HTML o como un documento separado, esta última opción permite reutilizar un mismo estilo para varias páginas. CSS da versatilidad a la hora de personalizar el estilo de las páginas web y subraya la idea aplicada en HTML5 de separar el aspecto del contenido.

JavaScript

JavaScript es un lenguaje de programación interpretado, orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico, que aporta la parte dinámica de la página web, tanto en el apartado gráfico como en el funcional. Se usa principalmente en la parte del cliente pero su uso también está comenzando a extenderse al lado del servidor, ejemplo de ello es el entorno de programación en la capa del servidor Node.js, basado en JavaScript. En la aplicación del proyecto juega un rol importante tanto en la gestión de la interacción del usuario con los objetos gráficos, como en la comunicación entre el cliente y el servidor basada en la tecnología WebSocket, la cual se describe más adelante. Como fue avanzando el desarrollo de la aplicación descubrí la existencia de una biblioteca de JavaScript llamada jQuery, que permite facilitar el uso de JavaScript simplificando la manera de interactuar con los documentos HTML, manipular el árbol del DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX (*Asynchronous JavaScript And XML*) a páginas web. Lo que más útil me ha resultado han sido sus selectores para acceder a los elementos del documento. En la tabla A.1 se observan ejemplos de la potencia de sus selectores.

Selector jQuery	**Selección significado**
`$('p:first')`	Selecciona el primer '<p>' de la página.
`$('img[src$=.png]:first'`	Selecciona el primer '<img>' de la página que tiene un atributo src acabado en .png.
`$('p:last')`	Selecciona el último '<p>' de la página.
`$('li:first-child')`	Selecciona todos los ' ' que son primeros hijos.
`$('li:last-child')`	Selecciona todos los ' ' que son últimos hijos.
`$('li:only-child')`	Selecciona todos los ' ' que sean hijos únicos.
`$('tr:nth-child(odd)')`	Selecciona todos los '<tr>' que sean impares.
`$('div:nth-child(3n)')`	Selecciona cada tercer ' '.
`$('p[title^="H"]')`	Selecciona todos los '<p>' cuyo atributo title comienza por H.

Tabla A.1: Ejemplos de uso de selectores de jQuery

Además de simplificar el uso de JavaScript, jQuery aporta una capa de abstracción que hace que sea compatible con la mayoría de navegadores. Cabe destacar que la aplicación está desarrollada para que funcione de forma correcta en el navegador Mozilla Firefox, de tal manera, que no se garantiza que funcione adecuadamente en otros navegadores. El uso de jQuery en la aplicación del proyecto es parcial, ya que descubrí la librería con la implementación del cliente bastante avanzada, en desarrollos futuros me gustaría completarla, de esta manera implementaría de forma sencilla la portabilidad a otros navegadores.

A.1.2. Back-end

El *back-end* es la parte del software que procesa la entrada de información enviada desde el *front-end*. En este caso incluye el servidor web y todos aquellos programas necesarios para la recepción de información proveniente del cliente y su tratamiento con la consiguiente respuesta al cliente o su almacenamiento para un estudio posterior de los datos.

Para entender las decisiones tomadas en el lado del servidor se debe recalcar que el principal problema en el desarrollo de esta aplicación era encontrar una tecnología que nos permitiese establecer una vía de comunicación para transmitir información entre clientes en tiempo real, minimizando latencias y teniendo en cuenta la sincronización. Estudié varias posibilidades, las más adecuadas son WebSockets y RTC PeerConnection.

WebSocket es una tecnología que permite crear un canal de intercambio de información bidireccional y asíncrono entre el servidor y el cliente. Una vez establecida la conexión entre el cliente y el servidor el flujo de datos será libre, no dependiente de peticiones repetitivas de servicio por parte del cliente al servidor (Tecnología Pull). Las principales ventajas de WebSocket son:

- Mejora en la relación entre datos útiles enviados y datos de cabecera frente al caso de HTTP.
- Por lo anterior disminuye la latencia.
- Soportado por la mayoría de navegadores.
- Tasa de mensajes recibidos/enviados por el cliente muy alta, aún estando el servidor de intermediario.

Sus desventajas más importantes:

- No hay implementación directa cliente-cliente de transacción de datos.
- De lo anterior se podría disminuir más la latencia.

RTC PeerConnection es una tecnología que permite crear un canal bidireccional "*peer to peer*" entre dos clientes. Soporta tanto vídeo como sonido para implementar videoconferencia sin necesidad de intercambio de datos con el servidor. Sus principales ventajas son:

- Conexión directa entre clientes, sin pasar datos al servidor.
- Por lo anterior disminuye la latencia.
- Sincronización más fácil de implementar

Sus desventajas más importantes:

- Tecnología joven, poca documentación.
- Solo la soportan navegadores con plataforma base WebKit (Safari, Google Chrome,...).

Entre las dos opciones termine eligiendo WebSockets, ya que, siendo todavía joven es la más veterana de ambas, ofreciendo una fiabilidad mayor que RTC PeerConnection¸ además esta última solo funciona para navegadores con motor gráfico WebKit. Teniendo en cuenta que los experimentos a realizar se plantean en un marco de red local, la importancia de la latencia que introduce entre clientes el servidor al hacer de intermediario del canal de comunicación, pierde importancia.

El primer paso a seguir para establecer una canal entre el cliente y el servidor es el proceso conocido como WebSockets Handshake, por el cual el cliente envía una petición de negociación al servidor y aguarda una respuesta del mismo, que puede tanto aceptar como rechazar la propuesta. Este intercambio inicial de información se hace utilizando el protocolo HTTP (*Hypertext Transfer Protocol*), en este mensaje se está pidiendo un cambio a nivel de aplicación del protocolo HTTP al WebSocket, mucho menos pesado, ya que la cabecera de HTTP es mucho mayor, introduce en la trama enviada más información no útil (metadatos). Si el servidor acepta la petición, se constituye un canal de comunicación bidireccional y full-duplex sobre un único socket TCP, por lo que cliente y servidor pueden comenzar a enviarse de forma simultánea y asíncrona información. En la figura puede verse un diagrama que representa este proceso.

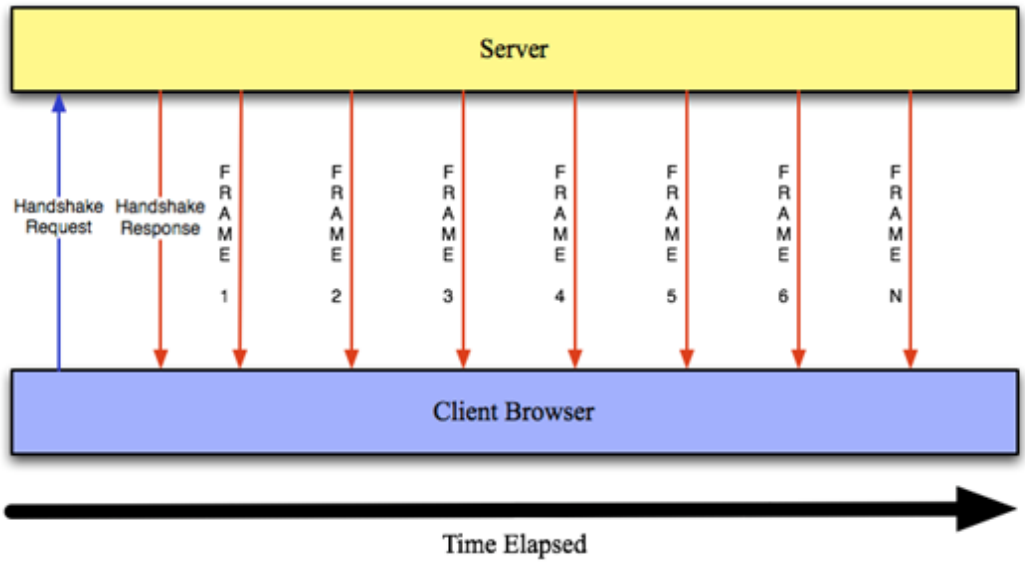

Figura A.1: Ejemplo de Handshake e intercambio de datos del protocolo WebSocket. [Extraido de http://marakana.com/bookshelf/html5_tutorial/web_sockets.html]

Como se ha descrito anteriormente la trama WebSocket es mucho más ligera, cada trama comienza con un byte 0x00 y termina con un byte 0x0F, entre ambos bytes se encuentra la información útil en formato de codificación UTF-8. Se pueden mandar tramas de datos tanto en texto como en binario, el caso de texto está soportado, sin embargo si se quisiera enviar información binaria, en la actualidad no esta soportado, habría que delegar tanto en el cliente como en el servidor la tarea de tratar dicha información. En la aplicación del proyecto se envía la información en modo texto, para ser más exactos en el formato para intercambio de datos JSON, el cual se define en un párrafo posterior. Para una mayor estudio acerca del protocolo WebSocket se deber acudir al RFC-6455 y para obtener información sobre la API del protocolo a http://www.w3.org/TR/2011/WD-websockets-20110419/.

JSON (*JavaScript Object Notation*) es un formato ligero para el intercambio de datos, subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML. El porque de la utilización de JSON frente a XML, un formato de serialización de datos más extendido y veterano, es que JSON representa mejor la estructura de los datos y requiere menos codificación y procesamiento. En la parte del cliente, gracias a JavaScript, la conversión de texto a objeto es inmediata y en la parte del servidor solo hace falta un objeto de clase *Gson* para hacer la conversión de texto a objeto y viceversa.

Una vez elegido el protocolo de comunicación, había que elegir el servidor a utilizar que soportase dicho protocolo. Dentro de las posibles implementaciones que investigué, me decanté por aquellas implementadas en Java, ya que tengo más experiencia en el uso de este lenguaje de programación, para ser más exactos estudie ejemplos de implementación para servidores Tomcat, GlashFish y Jetty. Fuera de Java investigué Node.js, comentado en un punto anterior, lo que me daba la posibilidad de utilizar JavaScript en el lado del servidor, lo que habría dado más uniformidad al código, al utilizar menos lenguajes de programación, pero con la complejidad del aprendizaje de los nuevos elementos añadidos a la API de JavaScript. Finalmente para la aplicación utilizo un servidor Tomcat a partir de la versión 3.2.

Para cubrir la comunicación cliente-servidor con la actividad principal de la aplicación, la página de juego, como se ha explicado con anterioridad, he utilizado WebSockets, para cubrir dicha comunicación en tareas secundarias he usado una combinación Java Servlets y JSP (*Java Server Pages*). Un Servlet es una clase en lenguaje Java usada para ampliar la funcionalidad de los servidores web a los que se accede vía modelo de programación *request-response*, un componente Web que se ejecuta dentro de un contenedor web y genera contenido dinámico. JSP es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo, es una manera alternativa, y simplificada, de construir servlets, pero con la posibilidad de añadir código java en forma de scripts como complemento al código HTML, de esta manera se puede generar el contenido de la página a partir de objetos en el servidor, un ejemplo sería dibujar una tabla de la que a priori no se conoce su número de filas, un factor que cambia según el contexto de ejecución, con JSP incrustaríamos código Java en la página que recorriera dicha tabla y generase el código HTML correspondiente a la tabla actual.

A.2. Tecnología WebSockets del lado del servidor

Lenguaje	Soporte WebSockets
C/C++	libwebsockets, Mongoose, POCO C++ Libraries, Tufão, Wslay, QtWebsocket
Erlang	Yaws, cowboy
Go	go.net/websocket, webrocket
Haskell	websockets
Java	Apache Tomcat 7, AutobahnAndroid, Play Framework, Atmosphere, Bristleback, GlassFish 3.1, Grizzly, HLL WebSockets, JBoss 7, Jetty 7, jWebsocket, Netty 3.3, MigratoryData WebSocket Server
.Net Framework	SignalR, Internet Information Services (IIS) 8, ASP.NET 4.5, Windows Communication Foundation 4.5 through NetHttpBinding, Fleck, SuperWebSocket, XSockets.NET
Clijure	http-kit, aleph
Nginx	Proxy (since version 1.3.13), Push Stream (3-rd party module)
Node.js	Socket.IO, WebSocket-Node
Objetive-C	SocketRocket, BLWebSocketsServer
Perl	Mojolicious, PocketIO
PHP	php-websocket, Ratchet, Hoa\Websocket
Python	AutobahnPython, WebSocket-for-Python, txWS, Gevent Websocket
Ruby	EM-WebSocket
Other	apache-websocket, mod_websocket for lighttpd, nginx supports websocket since version 1.3

Tabla A.2: Esta tabla presenta el software del lado del servidor que soporta el protocolo WebSockets, clasificado por lenguaje de implementación. [Extraído de http://en.wikipedia.org/wiki/WebSocket]

A.3. Diferencias entre HTML5 y HTML4/XHTML

Etiqueta	Atributos	Comentarios
<!-- -->	Estándar o ninguno	
<!DOCTYPE>	Estándar o ninguno	
<a>	href | target | rel | hreflang | media | type	Atributo Añadido: media Atributo cambiado: Target
<abbr>	Estándar o ninguno	
~~<acronym>~~		Etiqueta Eliminada
<address>	Estándar o ninguno	
~~<applet>~~		Etiqueta eliminada
<area>	Estándar o ninguno	
<article>	Atributos globales	Nueva etiqueta
<aside>	Atributos globales	Nueva etiqueta
<audio>	autobuffer | autoplay | controls | loop | src	Nueva etiqueta
<b>	Atributos globales	Etiqueta cambiada
<base>	Estándar o ninguno	
~~<basefont>~~		Etiqueta eliminada
<bb>	Estándar o ninguno	
<bdo>	Estándar o ninguno	
~~<big>~~		Etiqueta eliminada
<blockquote>	Estándar o ninguno	
<body>	Estándar o ninguno	
 	Estándar o ninguno	
<button>	Estándar o ninguno	
<canvas>	height | width	Nueva etiqueta
<caption>	Estándar o ninguno	
~~ ~~		Etiqueta eliminada
<cite>	Atributos globales	Etiqueta cambiada
<code>	Estándar o ninguno	
<col>	Estándar o ninguno	
<colgroup>	Estándar o ninguno	
<command>	checked | default | disabled | hidden | icon | label | radiogroup | type	Nueva etiqueta
<datagrid>	Estándar o ninguno	
<datalist>	Atributos globales	Nueva etiqueta
<dd>	Estándar o ninguno	
<del>	Estándar o ninguno	
<details>	open	Nueva etiqueta
<dialog>	Atributos globales	Nueva etiqueta
~~<dir>~~		Etiqueta eliminada
	Estándar o ninguno	

Tabla A.3: Tabla de diferencias entre HTML5 y HTML4 (parte 1). En amarillo aquellas etiquetas introducidas en esta nueva versión, en azul las etiquetas que han sido cambiadas todo o en parte y en gris las etiquetas eliminadas de esta versión. Si bien en la práctica los navegadores no lo están teniendo en cuenta para evitar perder cuota de mercado. [Extraído de http://es.wikipedia.org/wiki/HTML5]

Etiqueta	Atributos	Comentarios
<dfn>	Estándar o ninguno	
<dl>	Estándar o ninguno	
<dt>	Estándar o ninguno	
	Estándar o ninguno	
<embed>	height | src | type | width	Nueva etiqueta
<fieldset>	Estándar o ninguno	
<figure>	Atributos globales	Nueva etiqueta
~~<font>~~		Etiqueta eliminada
<footer>	Atributos globales	Nueva etiqueta
<form>	Estándar o ninguno	
~~<frame>~~		Etiqueta eliminada
~~<frameset>~~		Etiqueta eliminada
<h1> ... <h6>	Estándar o ninguno	
<head>	Estándar o ninguno	
<header>	Atributos globales	Nueva etiqueta
<hgroup>	Atributos globales	Nueva etiqueta
<hr>	Ninguno	Etiqueta cambiada
<html>	Estándar o ninguno	
<i>	Ninguno	Etiqueta cambiada
<iframe>	Estándar o ninguno	
<img>	Estándar o ninguno	
<input>	accept | alt | auto-complete | autofocus | cheked | disabled | form | formaction | formenctype | formmethod | formnovalidate | formtarget | height | list | max | maxlength | min | multiple | name | pattern1 placeholder | readonly | required | size | src | step | type | value | width	Etiqueta cambiada: Añadidos 13 elementos a type
<ins>	Estándar o ninguno	
~~<isindex>~~		Etiqueta eliminada
<kbd>	Estándar o ninguno	
<label>	Estándar o ninguno	
<legend>	Estándar o ninguno	
	Estándar o ninguno	
<link>	Estándar o ninguno	
<mark>	Atributos globales	Nueva etiqueta
<map>	Estándar o ninguno	
<menu>	Estándar o ninguno	
<meta>	Estándar o ninguno	
<meter>	high | low | max | min | optimum | value	Nueva etiqueta
<nav>	Atributos globales	Nueva etiqueta
~~<noframes>~~		Etiqueta eliminada
<noscript>	Estándar o ninguno	
<object>	Estándar o ninguno	

Tabla A.4: Tabla de diferencias entre HTML5 y HTML4 (parte 2). En amarillo aquellas etiquetas introducidas en esta nueva versión, en azul las etiquetas que han sido cambiadas todo o en parte y en gris las etiquetas eliminadas de esta versión. Si bien en la práctica los navegadores no lo están teniendo en cuenta para evitar perder cuota de mercado. [Extraído de http://es.wikipedia.org/wiki/HTML5]

Etiqueta	Atributos	Comentarios
	Estándar o ninguno	
<optgroup>	Estándar o ninguno	
<option>	Estándar o ninguno	
<output>	form	Nueva etiqueta
<p>	Estándar o ninguno	
<param>	Estándar o ninguno	
<pre>	Estándar o ninguno	
<progress>	max | value	Nueva etiqueta
<q>		
<ruby>	cite	Nueva etiqueta
<rp>	Atributos globales	Nueva etiqueta
<rt>	Atributos globales	Nueva etiqueta
~~<s>~~		Etiqueta eliminada
<samp>	Estándar o ninguno	
<script>	Estándar o ninguno	
<section>	cite	Nueva etiqueta
<select>	Estándar o ninguno	
<small>	Atributos globales	Etiqueta Cambiada
<source>	media | src | type	Nueva etiqueta
	Estándar o ninguno	
~~<strike>~~		Etiqueta eliminada
	Estándar o ninguno	
<style>	Estándar o ninguno	
<sub>	Estándar o ninguno	
<sup>	Estándar o ninguno	
<table>	Estándar o ninguno	
<tbody>	Estándar o ninguno	
<td>	Estándar o ninguno	
<textarea>	Estándar o ninguno	
<tfoot>	Estándar o ninguno	
<th>	Estándar o ninguno	
<thead>	Estándar o ninguno	
<time>	datetime | pubdate	Nueva etiqueta
<title>	Estándar o ninguno	
<tr>	Estándar o ninguno	
~~<tt>~~		Etiqueta eliminada
<u>	Define texto que debe tener un estilo diferente del texto normal 3	
	Estándar o ninguno	
<var>	Estándar o ninguno	
<video>	src | poster | autobuffer | autoplay | loop | controls | width | height	Nueva etiqueta
~~<xmp>~~		Etiqueta eliminada

Tabla A.5: Tabla de diferencias entre HTML5 y HTML4 (parte 3). En amarillo aquellas etiquetas introducidas en esta nueva versión, en azul las etiquetas que han sido cambiadas todo o en parte y en gris las etiquetas eliminadas de esta versión. Si bien en la práctica los navegadores no lo están teniendo en cuenta para evitar perder cuota de mercado. [Extraído de http://es.wikipedia.org/wiki/HTML5]

A.4. Diagramas de tecnologías de intercambio de información entre cliente-servidor

En las siguientes figuras se muestran diagramas de intercambio de información de protocolos, anteriores a WebSockets, que daban una peor solución a la comunicación cliente servidor en tiempo real.

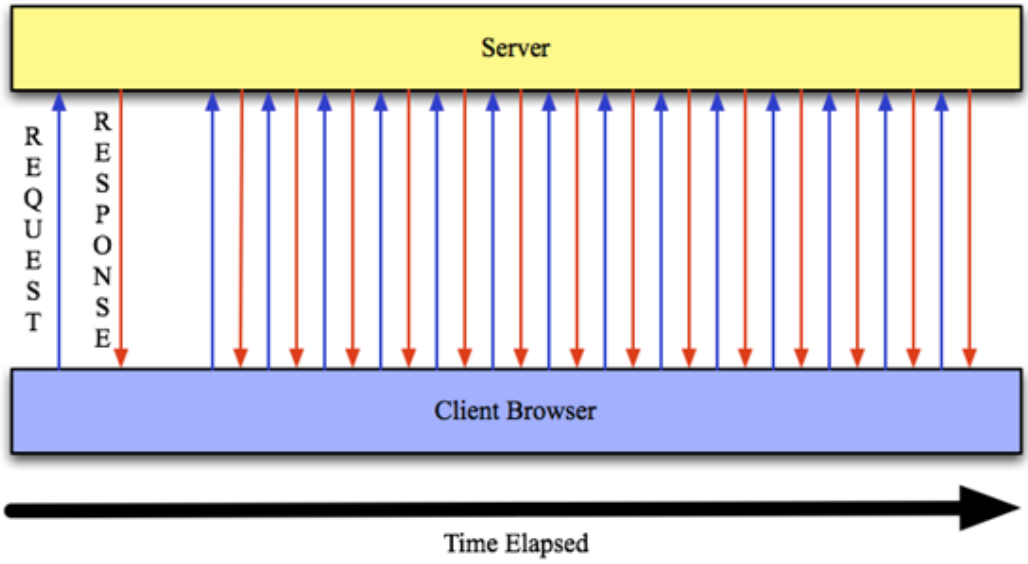

Figura A.2: Ejemplo de Handshake e intercambio de datos del protocolo *Polling*. [Extraido de http://marakana.com/bookshelf/html5_tutorial/web_sockets.html]

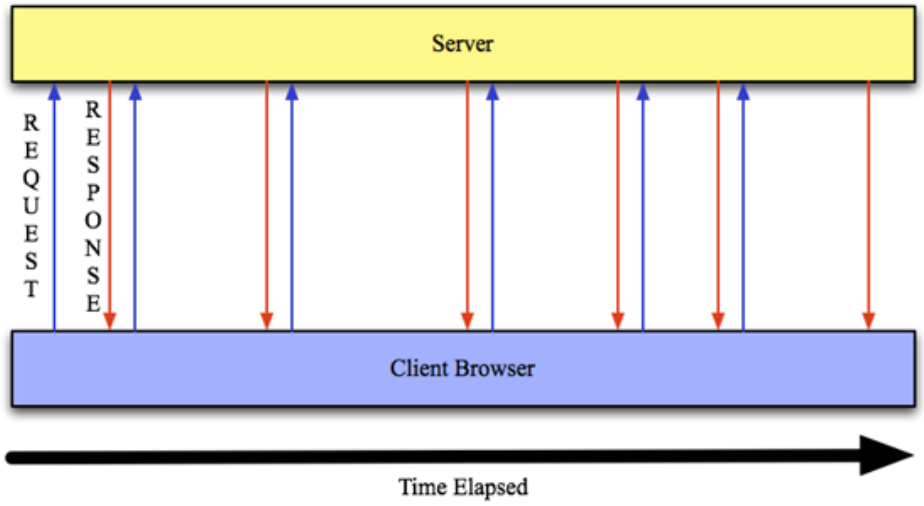

Figura A.3: Ejemplo de Handshake e intercambio de datos del protocolo *Long Polling*. [Extraido de http://marakana.com/bookshelf/html5_tutorial/web_sockets.html]

Anexo B

Información detallada de la aplicación

B.1. Descripción de la aplicación

En este apartado se va a describir la aplicación desarrollada para implementar un marco de ejecución virtual para experimentos de cruce perceptual, descritos en el capítulo 2. En primer lugar se describe la aplicación de forma general pasando por cada una de las páginas que la componen. En segundo lugar se describirá más en detalle la zona crítica de la aplicación donde se desarrollan los experimentos, analizando la estructura cliente-servidor, modos de juego, estados de una ronda, bots implementados, características y errores controlados. En tercer lugar se muestran las reglas del filtro para parámetros de configuración. En cuarto lugar los bocetos iniciales de la página web. En quinto lugar las capturas de las páginas de la aplicación web.

B.1.1. Descripción general

Comienzo la descripción de las diferentes páginas de la aplicación con la página de presentación de la aplicación, en ella se introduce el nombre de jugador que va a tener el usuario durante el experimento. Esta información se pasa al servidor, el cual verifica si se trata de un usuario normal o del administrador, redirigiendo a la página correspondiente. El usuario administrador debe cumplir que se conecta desde la máquina que contiene al servidor y que el nombre introducido es el correspondiente al administrador. En una futura implementación me gustaría añadir más seguridad a la identificación del administrador y más libertad para conectarse desde cualquier máquina. En la figura B.1 se muestra una captura de la página inicial.

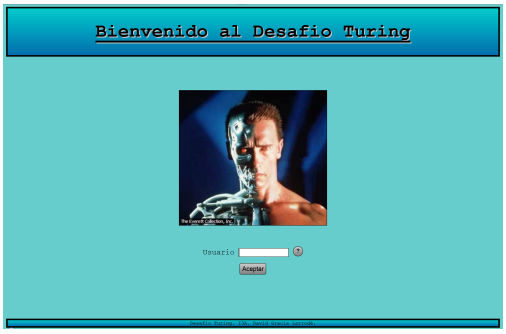

Figura B.1: Captura de index.html

Si el usuario se identifica como administrador, la siguiente página es la de control del comportamiento de la aplicación. Esta página se comunica con un servlet en el lado del servidor que ejecuta la operación correspondiente indicada por el cliente. Desde esta pantalla se pueden consultar y modificar los datos de configuración de la aplicación almacenados en el fichero `configuration.json` y habilitar/deshabilitar una barrera que impide el paso a los usuarios normales a la página juego.html. Las características de configuración son las siguientes:

- **numPairs:** Identificador numérico del próximo emparejamiento entre jugadores.
- **maxPlayers:** Número de jugadores (Personas + Bots).
- **maxBots:** Número de bots.
- **waitTime:** Tiempo en milisegundos del retardo del bot DELAYEDSHADOW.
- **pixelLength:** Longitud en píxeles de distancia del bot PIXELSHADOW.
- **roundMax:** Número de rondas del experimento.
- **maxTime:** Tiempo en segundos de cada ronda.
- **botFirstMode:** Indica el tipo inicial de un bot cuando se crea.
- **winners:** Indica cuantos jugadores persona pasan a una segunda vuelta de rondas, si es igual a 0 solo hay una vuelta.
- **multipleMode:** Indica el modo en que se juega la vuelta. Si es igual a 0 modo normal, si es 1 modo múltiple (tiempo) y si es 2 modo múltiple (píxel).
- **multRanking:** En modo múltiple indica el tipo de ranking a generar tras una vuelta completa de rondas.
- **botModeLoop:** Indica si el tipo de los bots cambia a lo largo de la vuelta o siempre es el mismo.
- **collTimeout:** Indica el tiempo máximo en milisegundos en el cual, tras producirse una colisión, si se hace click se considera colisión.

Se ha añadido un filtro en el servidor a la hora de modificar datos de configuración, de tal manera que si se incumple alguna de las reglas fijadas no se produzca modificación alguna. La información de los errores llega al cliente en forma de número potencia de dos, el cuál al pasar a binario, marca con unos aquellos indices del vector de mensajes que se mostrarán por pantalla. En la figura B.2 se observa una captura de la página de administración.

Si el usuario es un jugador normal se llega a la página de explicación, donde hay una breve descripción del experimento a realizar para orientar al usuario en el transcurso del mismo. Si el administrador no ha deshabilitado la barrera, no se podrá pasar de esta página, en el momento en que este deshabilitada se podrá continuar a la página de juego. La figura B.2 contiene una captura de la página de explicación.

La siguiente página es la de juego, en ella se va a centrar el grueso de la aplicación. Según el estado en que se encuentre la ronda y el modo de juego, se mostrará una información u otra. Tiene dos zonas bien diferenciadas, la zona de juego y la zona de respuesta o paso de ronda.

En la zona de juego se muestra información sobre la ronda en la que nos encontramos, según el estado aparece una explicación de cómo se debe actuar, un cronómetro, en una zona determinada, aparece la imagen de un ratón indicando de forma gráfica el movimiento a realizar durante

el juego o un marcador de colisiones activas dependiendo del estado y por último una línea de 800 píxeles y un cuadrado de 50 píxeles de lado en la zona inferior, el cuadrado, que representa el campo receptor del usuario, podrá desplazarse a lo largo de la línea gracias al movimiento del ratón. Durante el transcurso de una ronda de juego la página va pasando por una serie de estados, que se explican en un apartado posterior, el tiempo de una ronda lo regula el cronómetro, tras una aviso de que va a comenzar la ronda en la zona del marcador, el crono empieza su cuenta atrás, en ese momento el color de fondo de la pantalla pasa a negro, de esta manera se consigue un mejor contraste de los cambios de color en el cuadrado, aparece el marcador de colisiones inicializado a cero, el usuario podrá mover su cuadrado durante el tiempo estipulado. La entrada sensitiva del *framework* es visual y sonora, frente a la táctil del experimento original. Cuando se produce una colisión con el cuadrado receptor del contrario, el cual no es visible en su ventana, el cuadrado del usuario cambia de color fugazmente y el marcador de colisiones se incrementa. Se jugó con el contraste entre colores para reforzar la percepción de colisiones. Se añadió también un aviso sonoro con efecto *Doppler,* en caso de colisión, se reproduce el sonido cuando las superficies de los cuadrados se tocan, alcanzado el máximo volumen del audio cuando se produce una colisión perfecta entre receptores y va disminuyendo como se alejan, en el momento en que dejan de tocarse el sonido desaparece por completo.

La zona de respuesta o paso a la siguiente ronda no es visible en todo momento, solo se hace visible al terminar de jugar una ronda por completo. Si el parámetro de configuración *multipleMode* es igual 0, estamos en modo normal (1 vs 1) por lo que se le hace al usuario la pregunta de si ha interactuado con una persona o una máquina. Si es mayor que 0, nos encontramos en modo múltiple (3 vs 3) se informa al usuario de que se han guardado los datos correctamente y que puede continuar presionando el botón correspondiente. Si se ha jugado la última ronda de la vuelta actual al presionar el botón con el texto "Terminar" se avanza a la página de ranking, en caso contrario el botón contendrá el texto "Continuar" y se iniciará la nueva ronda.

La página del ranking, es una forma de incentivar a los jugadores para que realicen los experimentos concentrados. Según el modo en el que nos encontremos se visualizará un ranking u otro.

- Modo normal - Se muestra `ranking.jsp`, el orden de aparición de los participantes depende del número de veces que se acertó la pregunta de final de ronda, cuantas más veces mejor. En caso de empate se tendría en cuenta el mayor número de colisiones activas con el contrincante.
- Modo múltiple - Según el valor que tome el parámetro de configuración *multRanking:*
  - Si es igual a 0 se muestra `rankingHits.jsp`. Se ordena a los jugadores de mayor a menor número de clicks tras un estímulo del contrincante (persona). En caso de empate se tiene en cuenta un mayor número de colisiones con humanos.
  - Si es igual a 1 se muestra `rankingFitness.jsp`. En este caso se ordena a los jugadores atendiendo a una función de *fitness* igual a $(f = colisionesHumanas/colisionesTotales)$, inspirada en la función de adecuación que utilizan para entrenar las CTRNNs Iizuka y Di Paolo (2007)[25], simplificada sería $f = colisionesHumanas/20$, cogiendo como constante no superable, el número de estímulos activos total, en vez de 20.

Si el parámetro de configuración *winners* es mayor que cero significa que hay una segunda vuelta de rondas por lo que se volvería a la página de juego. En caso de que sea igual a cero o ya se haya producido la segunda vuelta se avanza a la página de cuestionario.

Tras terminar todas las rondas de todas las vueltas se pasa a la página de cuestionario cuyo objetivo es recoger información de la experiencia de utilizar la aplicación por parte de los usuarios y de esa manera ajustar tiempos, corregir posibles errores, etc. En la figura B.2 se ve una captura

del cuestionario.

Tras el cuestionario se llega a la página de despedida, en ella se muestra un mensaje de fin de partida y existe un botón que nos permite volver a la página de comienzo. Por último, en la figura B.2 se observa una captura de la misma.

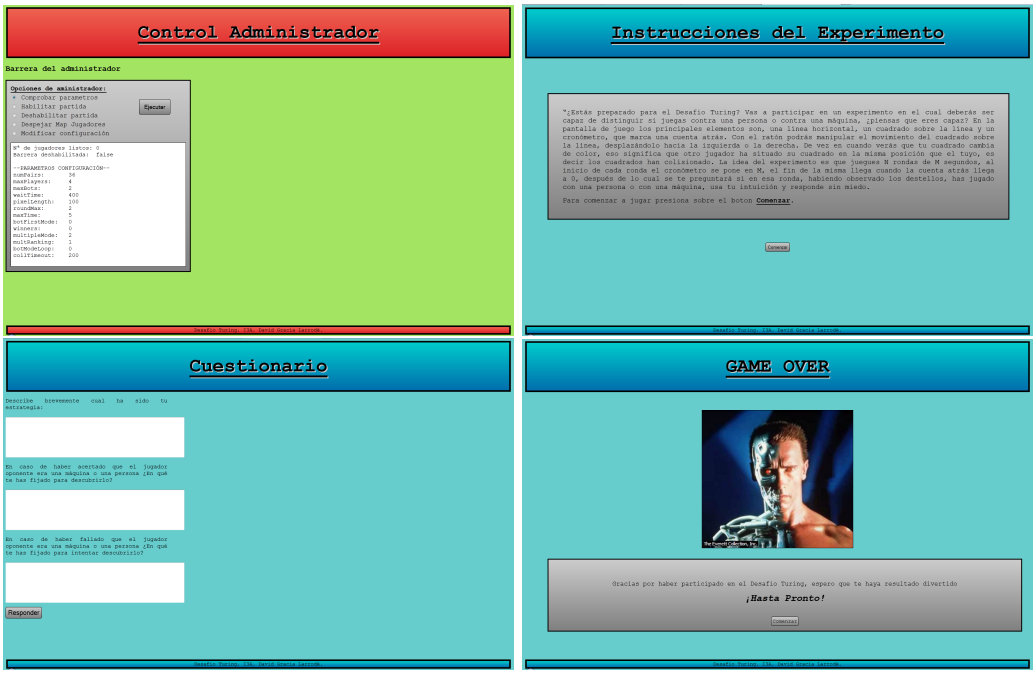

Figura B.2: Captura de adminBarrier.html (**arriba-izquierda**), de explicacion.html (**arriba-derecha**), de cuestionario.html (**abajo-izquierda**) y de game_over.html (**abajo-derecha**)

B.1.2. Arquitectura cliente-servidor

La función principal del sistema, está construida sobre una arquitectura de tipo cliente-servidor basada en la tecnología WebSockets.

En la parte del cliente mediante una combinación de HTML5, CSS y JavaScript se gestiona el aspecto gráfico, el movimiento del cuadrado receptor sobre la línea al mover el ratón, la cuenta atrás antes de una ronda, el cronómetro, el marcador de colisiones activas, cambios en la información de ayuda según el estado, aparición y desaparición de la zona de respuesta o de cambio de ronda, y por su puesto, con JavaScript, la implementación de WebSockets en la parte del cliente, gracias a la cual el cliente puede conectarse y desconectarse del servidor, enviar mensajes (JSON) y recibir y procesar los originados en el servidor.

En la parte del servidor, se controla el ciclo de estados de la ronda en el cliente, el reparto de emparejamientos entre participantes, el intercambio de información de posicionamiento del receptor entre jugadores persona con jugadores persona y/o jugadores máquina, la creación de bots, gestión de su comportamiento según el modo en que se encuentren, almacenado de información de ronda y cálculos para generar el ranking correspondiente. El comportamiento del servidor al conectarse un cliente es el de generar un objeto de la clase *GameConnection*, que representa al cliente en el servidor, en esta clase es donde se implementa el comportamiento del servidor frente interacciones originadas en el cliente, al conectarse o desconectarse, la recepción y envío de mensajes, al igual

que se hace con JavaScript en la parte del cliente. Si el cliente que se conecta es el primero, es el encargado de generar todos los bots u objetos *BotPlayer* indicados por el parámetro *maxBots*. Cada objeto *BotPlayer* crea a su vez un objeto *BotThread* en el cual se ejecuta el bucle principal del comportamiento del bot, mientras se está jugando una ronda, en un hilo de ejecución paralelo para no bloquear la aplicación.

En la figura B.3 pueden observarse dos esquemas simplificados a modo de ejemplo de las conexiones y relaciones que se crean en la arquitectura de la aplicación según el tipo de ronda. El primero es una ronda entre dos jugadores persona y el segundo una ronda entre un jugador persona y otro jugador máquina:

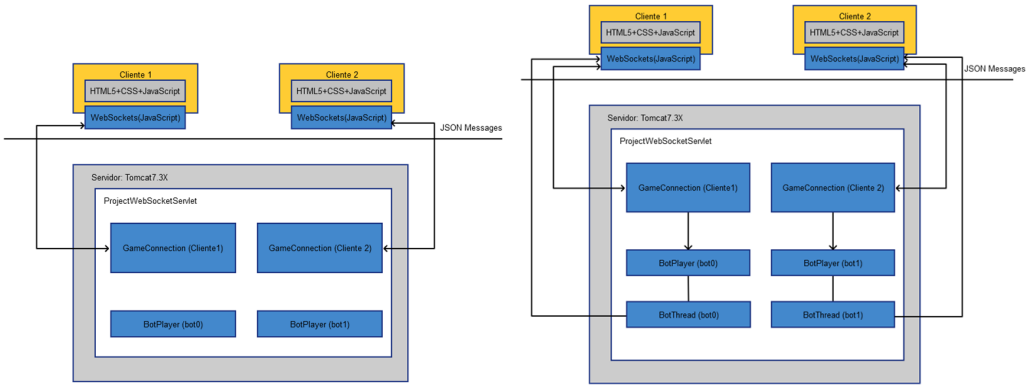

Figura B.3: Diagrama que hace referencia a la arquitectura en una ronda entre dos jugadores persona (**izquierda**). Diagrama que hace referencia a la arquitectura en una ronda entre un jugador persona y otro jugador máquina (**derecha**):

B.1.3. Modos de juego

La aplicación soporta tres modos de juego, que permiten reproducir experimentos originales y crear otros nuevos: normal, múltiple (tiempo) y múltiple (píxel)..

El modo normal permite representar el experimento de Iizuka y Di Paolo (2007)[25] y el de Iizuka et al.(2009[24],2012a[22]). Una ronda de este tipo es de uno contra uno, bien entre "persona vs persona" o "persona vs bot". Durante el tiempo delimitado el jugador moverá su cuadrado receptor interactuando con el de su adversario. Una vez finalizado ese tiempo el jugador debe responder a la cuestión de si ha interactuado con una máquina o una persona. Dependiendo del valor del parámetro *botModeLoop*, la evolución de los bots cambia. Si es igual a 0, el modo de los bots siempre será el mismo, el que marque la propiedad *botFirstMode*, no cambiará durante el transcurso de la rondas de una vuelta. Si es mayor que cero, los bots irán pasando por los diferentes modos disponibles (RANDOMSIMPLE, SHADOW, DELAYEDSHADOW y COMPLEX) según avancen las rondas

El modo múltiple con retardo por tiempo se corresponde a un combinación del experimento original de Auvray et al.(2009)[5] añadiéndole la dimensión temporal del de Iizuka y Di Paolo (2007)[25]. En una ronda de este modo, cada jugador humano se enfrenta a un contrincante persona, la sombra del mismo con un retardo en milisegundos marcado por el parámetro de configuración *waitTime* y un valor fijo de posición del cuadrado. Durante el transcurso de la ronda el jugador puede mover el cuadrado receptor para interactuar con el resto de jugadores y hacer click para indicar que piensa que la última interacción ha sido con el jugador humano (*hit*), contra más aciertos mejor. Al finalizar la ronda la información pertinente es almacenada y el jugador puede avanzar a

la siguiente ronda o terminar la vuelta.

El modo múltiple con retardo por longitud se corresponde con el experimento original de Auvray et al.(2009)[5]. En una ronda de este modo, cada jugador humano se enfrenta a un contrincante persona, la sombra del mismo con una distancia en píxeles marcada por el parámetro de configuración *pixelLength* y un valor fijo de posición del cuadrado. Durante el transcurso de la ronda el jugador puede mover el cuadrado receptor para interactuar con el resto de jugadores y hacer click para indicar que piensa que la última interacción ha sido con el jugador humano (*hit*), contra más aciertos mejor. Al finalizar la ronda la información pertinente es almacenada y el jugador puede avanzar a la siguiente ronda o terminar la vuelta.

B.1.4. Estados de la aplicación cliente

La aplicación cliente pasa por una serie de estados en el desarrollo de cada ronda. Son los siguientes:

- **WAITING:** Esperando que llegue un jugador contrincante.
- **PLAYING:** Jugando contra otro jugador o jugadores.
- **ANSWERING:** respondiendo al cuestionario.
- **WRITING:** Escribiendo datos de la ronda en el servidor.
- **SAVED:** Escritura en el servidor confirmada.
- **ENDING:** Final de la última ronda de una vuelta.

En caso de error también existe un estado:

- **ERROR:** Control de error enviado por el servidor.

La figura B.4 contiene un esquema gráfico del ciclo de estados que sigue la aplicación. Para que todo éste ciclo se vaya produciendo, el cliente y el servidor intercambian una serie de datos como puede observarse en el diagrama de paso de mensajes, para el caso de una ronda en modo normal, de la figura B.5.

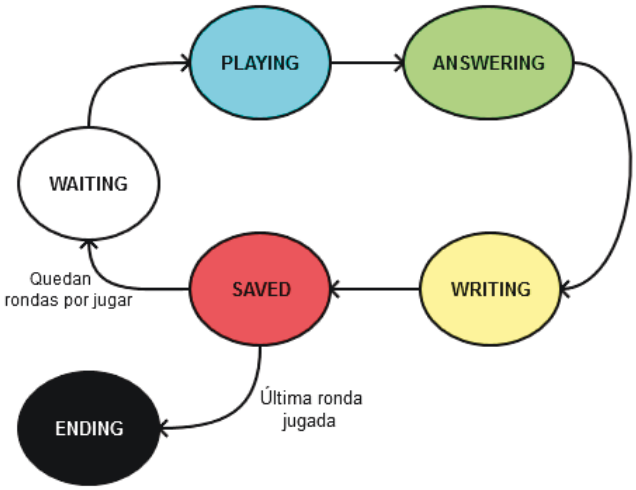

Figura B.4: Gráfico de estado por los que pasa el cliente

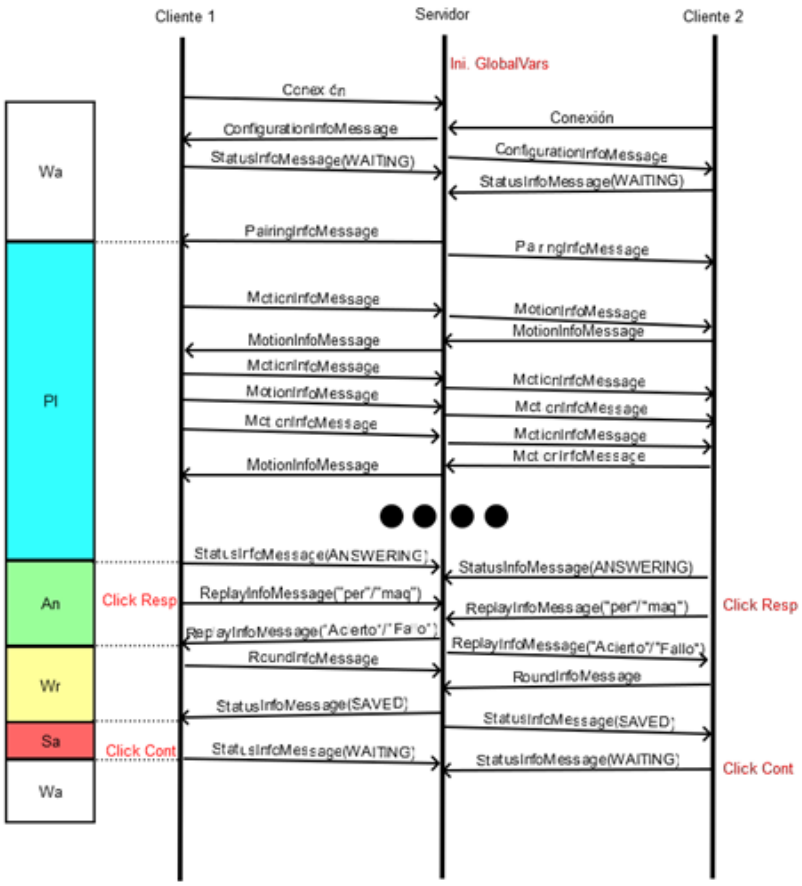

Figura B.5: Diagrama de paso de mensaje entre cliente y servidor durante una ronda en modo normal

B.1.5. Modos de un bot

Los diferentes tipos de bot implementados para el proyecto son los siguientes:

- **SIMPLE:** Solo envía "*MotionInfoMessage*"[1] variando las Xs con un incremento constante de 30 píxeles, cada segundo. Se usaba solo en modo normal.
- **RANDOMSIMPLE:** Solo envía "*MotionInfoMessages*", las Xs se envían cada medio segundo con un valor aleatorio entre 0 y 800. Se usa solo en modo normal.
- **SHADOW:** Solo envía "*MotionInfoMessages*", el valor de las Xs y su instante de lanzamiento siguen la traza de un jugador persona realizada anteriormente. Se usa solo en modo normal.
- **DELAYEDSHADOW:** Recibe y envía "*MotionInfoMessages*", repite la traza de Xs de la persona con la que está jugando la ronda actual, con un retraso en milisegundos definido por el parámetro de configuración *waitTime*. Se usa en modo normal y múltiple con retraso por tiempo.

[1]Nota: "*MotionInfoMessage*" es un mensaje de posicionamiento de cuadrados que intercambian los jugadores durante una ronda al mover el cuadrado.

- **COMPLEX:** Recibe y envía "*MotionInfoMessages*", este bot sigue políticas de comportamiento frente a los mensajes que le lleguen de jugador oponente persona. Hay definidas dos políticas: acercarse al oponente o alejarse del mismo. La primera posición del bot es 0, cuando al bot le llega una Xs modifica el valor de su Xs, sumando si el oponente esta a la derecha o restando si esta a la izquierda un número aleatorio entre 1 y una velocidad máxima. Durante el tiempo que no recibe mensajes del oponente, el bot no está ocioso, si no que incrementa una variable que delimita la velocidad de envío de mensajes, de tal manera que cuando es múltiplo de un número grande dado se lanza un mensaje que sigue la política de acercarse, de esta manera no se satura la comunicación con mensajes del bot. Las políticas de acercamiento o alejamiento puede ser reprogramadas como interese, las actuales son las que pienso ofrecen mayor dificultad al jugador persona. Se usa solo en modo normal.
- **PIXELSHADOW:** Recibe y envía "*MotionInfoMessages*", retransmite la traza de Xs de la persona con la que está jugando la ronda actual, menos una distancia en píxeles marcada por el parámetro pixelLength. Si el resultado de la diferencia es negativo, se recalcula el valor de X sumándole 800 de esta manera aparece por el otro extremo. Se usa solo en modo múltiple con retraso por distancia en píxeles.

B.1.6. Características y limitaciones

A continuación una lista de características y limitaciones de la aplicación:

- El servlet WebSocket y conexión al servidor durante una vuelta de rondas están centrados en una única página, juego.html.
- La aplicación se ejecuta de forma correcta en el navegador Mozilla Firefox, no se asegura su buen funcionamiento en otros navegadores.
- El nº de jugadores debe ser un nº par, por implementación.
- Hasta que no se completa el nº máximo de jugadores no empieza la partida, ya que es el último jugador persona en entrar el que reparte los mensajes de emparejamiento. Igual pasa al final de una ronda no terminal.
- Los mensajes transmitidos entre cliente-servidor en ambos sentidos son cadenas de texto JSON (*JavaScript Object Notation*). También se usa esta notación para almacenar la información de la ronda de un jugador persona en un fichero de texto externo.
- El reparto de parejas, en modo normal, se hace de la forma siguiente:
  1. Se construyen una lista de jugadores personas y otra de jugadores bots, a partir de los "Maps" de la clase principal.
  2. Se coge el primer bot persona y usando la función "*Math.random()*" de java se elige a su pareja que puede ser tanto persona como máquina.
  3. Se envía a los clientes persona un "*PairingInfoMessage*"[2] y se llama a la función que inicia el bucle de los bots.
  4. Se borran de sus listas correspondientes los jugadores emparejados.
  5. Se repite el bucle hasta que todos los jugadores persona tienen pareja.
- El reparto de parejas, en modo múltiple, se hace de la forma siguiente:
  1. Se construyen una lista de jugadores personas y otra de jugadores bots DELAYEDSHADOW o PIXELSHADOW, a partir de los "Maps" de la clase principal.

[2] "*PairingInfoMessage*" es el tipo de mensaje que el servidor envía a los jugadores persona para informarles de los emparejamientos formados.

2. Se generan dos números distintos usando la función "*Math.random()*", los cuales indican los dos jugadores persona del emparejamiento.
3. Se cogen los dos primeros bots de la lista se asignan individualmente como bots sombra a cada jugador persona del emparejamiento.
4. Si se esta en modo múltiple con retraso en milisegundos, se genera un número aleatorio con la función java con un valor comprendido entre 0 y 800, este es el valor fijo que se asocia a ambos jugadores persona. Si por el contrario, se esta en modo múltiple con distancia en píxeles, se generan otros dos números aleatorios en vez de solo uno, uno de 0 a 400 y otro de 401 a 800, que son los valores fijos que se asocian cada uno a un jugador persona distinto.
5. Se envía a los clientes persona un "*PairingInfoMessage*" y se llama a la función que inicia el bucle de los bots.
6. Se borran de sus listas correspondientes los jugadores emparejados. Se repite el bucle hasta que todos los jugadores persona tienen pareja.

- La evolución de los modos de un bot en modo normal, durante las sucesivas rondas jugadas por el mismo, siguen un orden secuencial, por implementación:
  - RANDOMSIMPLE, SHADOW, DELAYEDSHADOW y COMPLEX.
- Existe un fichero de configuración con notación JSON llamado `configuration.json`, que el servidor lee nada más arrancar para tomar los valores iniciales de variables globales, sus valores por defecto son:
  - `{"numPairs":0,"maxPlayers":2,"maxBots":1,"waitTime":400,"pixelLength":100,"roundMax":4, "maxTime":15,"botFirstMode":0,"winners":0, "multipleMode":0,"multRanking":0,botModeLoop":1, "collTimeout":200}`

B.1.7. Errores tratados gráficamente

Hay una serie de errores que se controlan y muestran de forma gráfica para informar al cliente del problema. En caso de una caída del servidor mientras se está en la pantalla de juego se muestra el error de la figura B.6, reescribiendo el código HTML de la misma página.

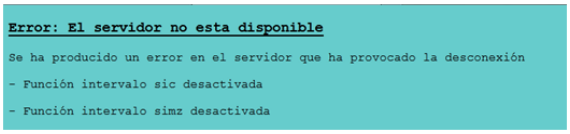

Figura B.6: Error producido al desconectarse de forma inesperada el servidor.

Si se produce el intento de conexión con el servidor de un cliente con el mismo identificador que otro ya conectado, se rechaza, un ejemplo de cómo puede darse el caso es si en un mismo navegador se abre una nueva pestaña y se intenta conectar. En la figura B.7 puede observarse el mensaje que se mostraría por pantalla.

Si mientras dos jugadores humanos enfrentados juegan su ronda, uno de ellos pierde la conexión sin haber terminado la cuenta atrás del cronómetro, al jugador todavía conectado se le informa del suceso y se le da la oportunidad de repetir la ronda, como se muestra en la figura B.8.

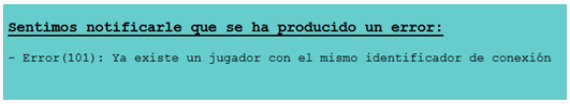

Figura B.7: Error producido al intentar conectarse un segundo jugador persona con el mismo identificador de conexión.

Figura B.8: Error producido al desconectarse el jugador oponente mientras se estaba jugando.

B.2. Filtro de parámetros de configuración

Como se ha comentado en la subsección B.1.1 de este capítulo, en la página de administración hay implementado un filtro, para que en caso de que se incumpla una norma no se modifique dato alguno. Las reglas son:

1. *maxPlayers* debe ser mayor igual que 2
2. *maxPlayers* debe ser múltiplo de 2
3. *maxBots* debe ser menor igual que el nº de jugadores humanos
4. *numPairs* debe ser mayor igual que 0
5. *waitTime* debe ser mayor que 0
6. *pixelLength* debe ser mayor que 0
7. *roundMax* debe ser mayor que 0
8. *maxTime* debe ser mayor que 0
9. *botFirstMode* debe ser un valor entre 0 y 3, sin incluir el 1
10. *botFirstMode* debe ser distinto de 1 -> Modo SHADOW dependiente de un estado anterior
11. *winners* debe ser mayor igual que 0
12. *winners* debe ser menor igual que el nº de jugadores humanos

13. *multipleMode* debe ser un valor entre 0 y 2
14. En modo múltiple el número de jugadores humanos debe ser múltiplo de 2
15. En modo múltiple *maxBots* debe ser igual que el nº de jugadores humanos
16. *multRanking* debe ser un valor entre 0 y 1
17. *botModeLoop* debe ser mayor igual que 0
18. *collTimeout* debe ser mayor que 0

B.3. Bocetos iniciales de la página web

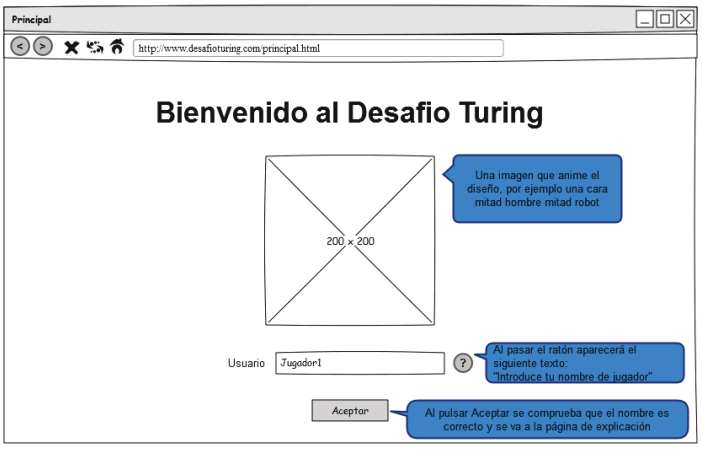

Figura B.9: Boceto de la página inicial de la aplicación.

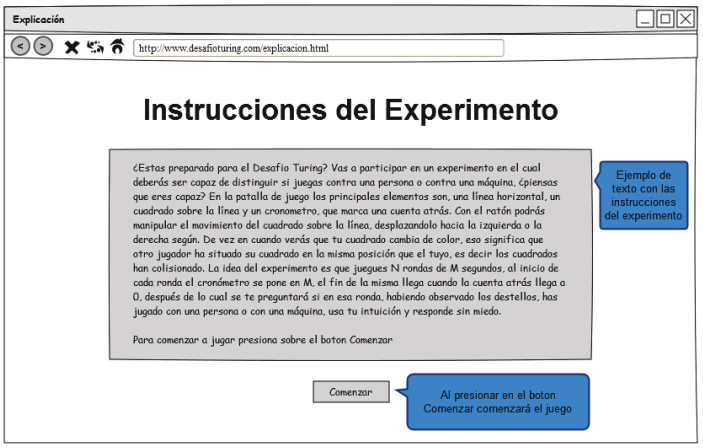

Figura B.10: Boceto de la página de explicación de la aplicación.

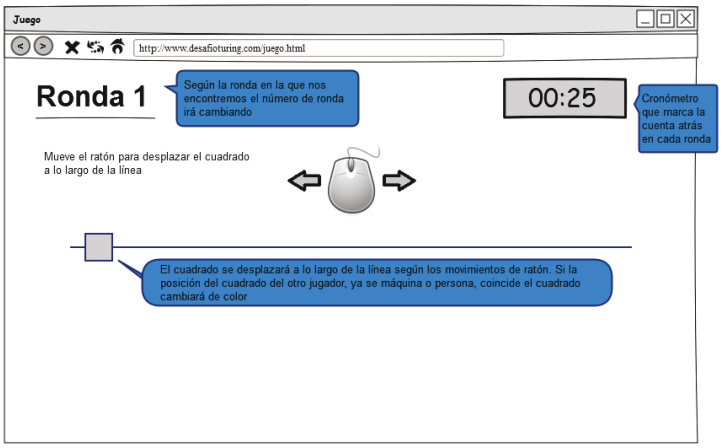

Figura B.11: Boceto de la página de juego durante una ronda.

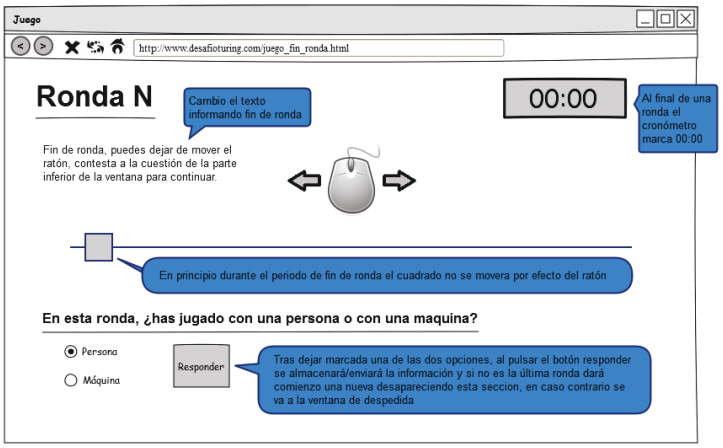

Figura B.12: Boceto de la página de juego al responder el cuestionario.

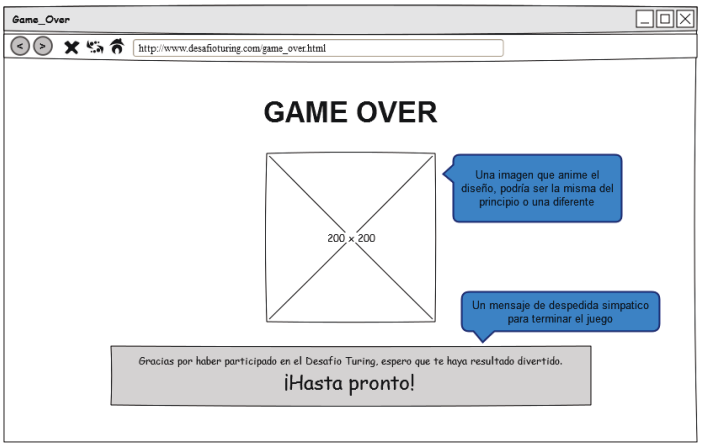

Figura B.13: Boceto de la página final de la aplicación

B.4. Capturas de las páginas de la aplicación

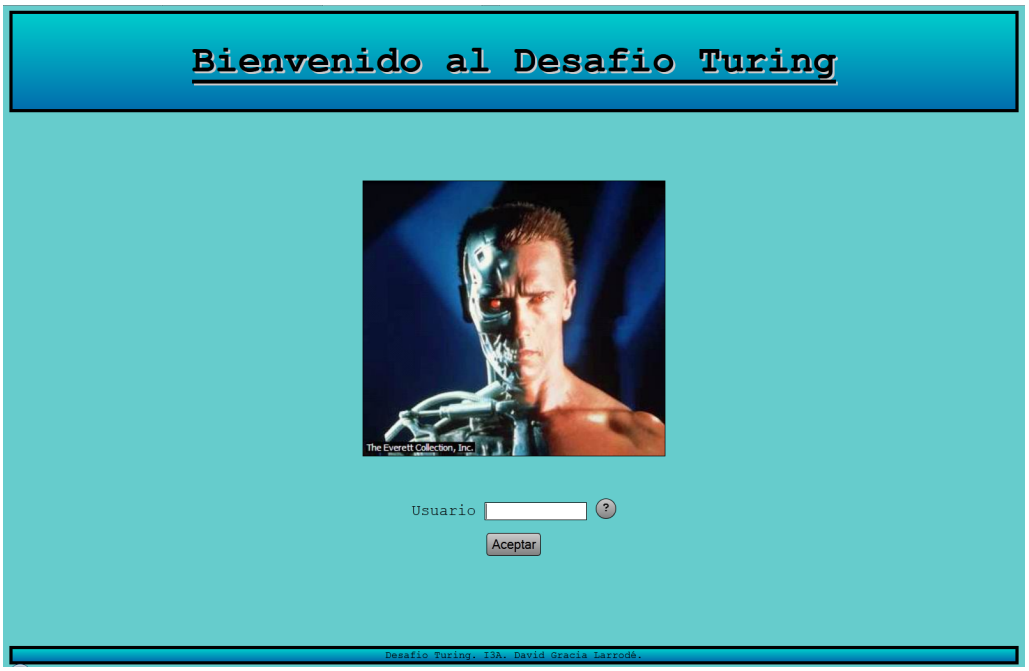

Figura B.14: Captura de la página inicial de la aplicación.

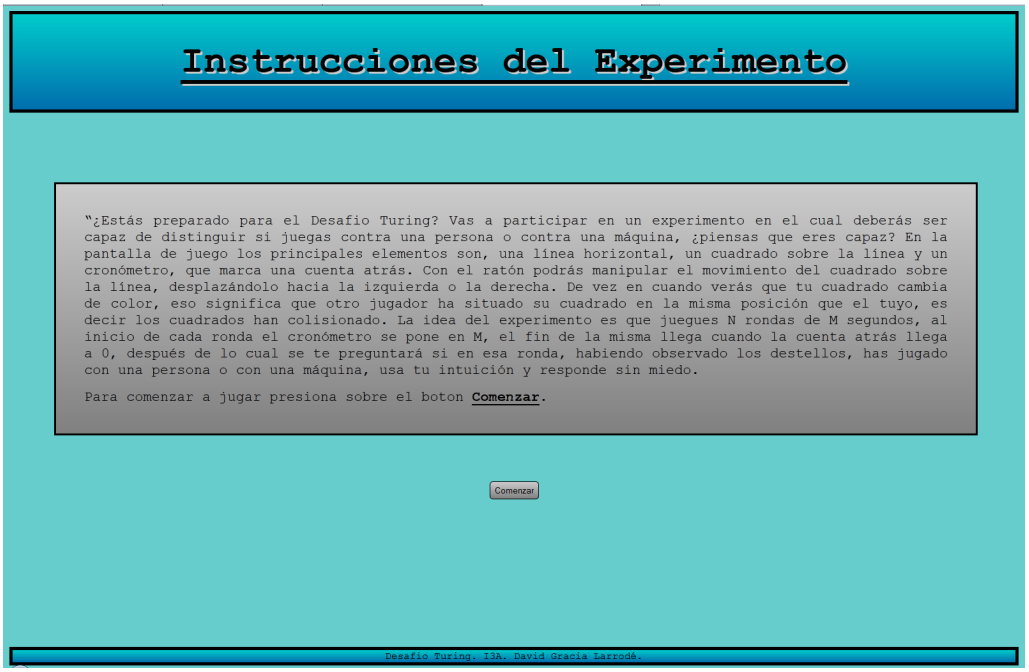

Figura B.15: Captura de la página de explicación de la aplicación.

Figura B.16: Captura de la página de juego esperando contrincante.

Figura B.17: Captura de la página de juego durante una ronda.

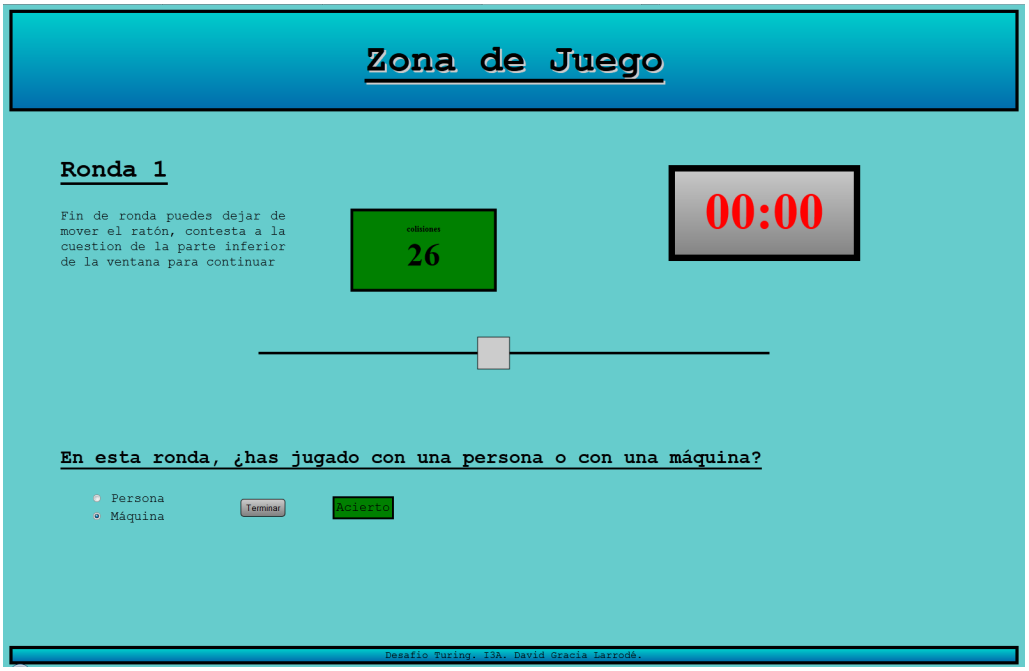

Figura B.18: Captura de la página de juego respondiendo cuestionario (Modo normal).

Figura B.19: Captura de la página de juego continuar siguiente ronda (Modo múltiple).

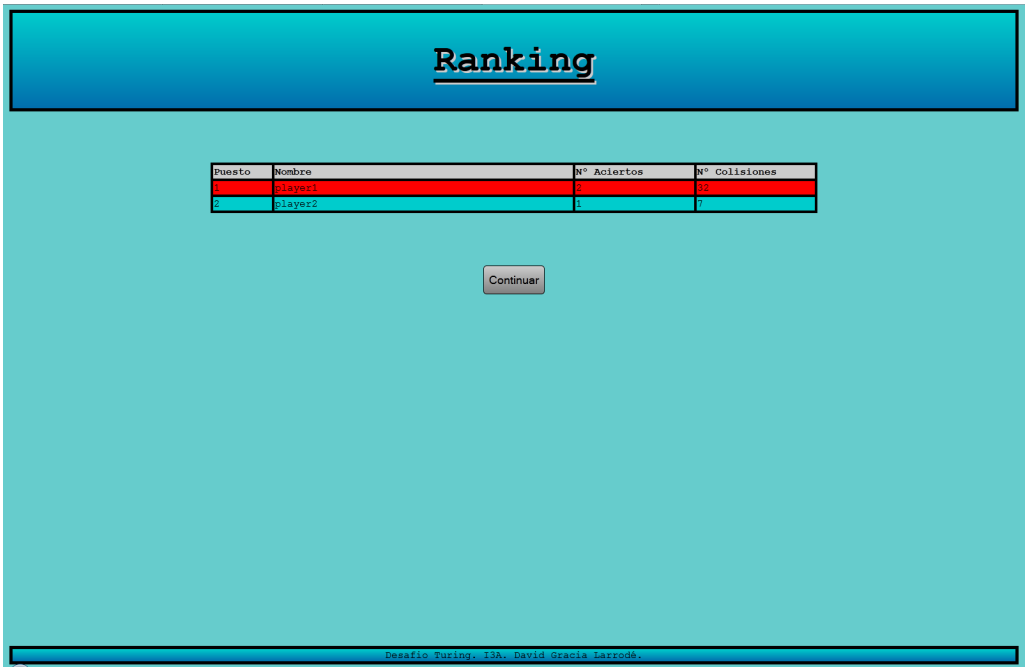

Figura B.20: Captura de la página del ranking en modo normal.

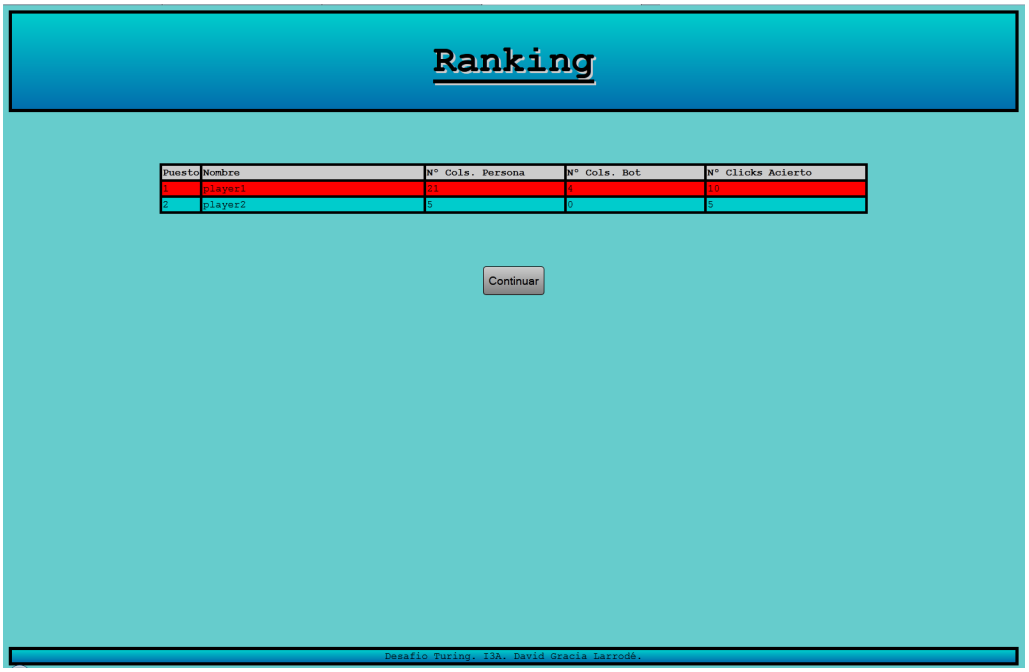

Figura B.21: Captura de la página del ranking en modo múltiple ordenado por clicks acertados.

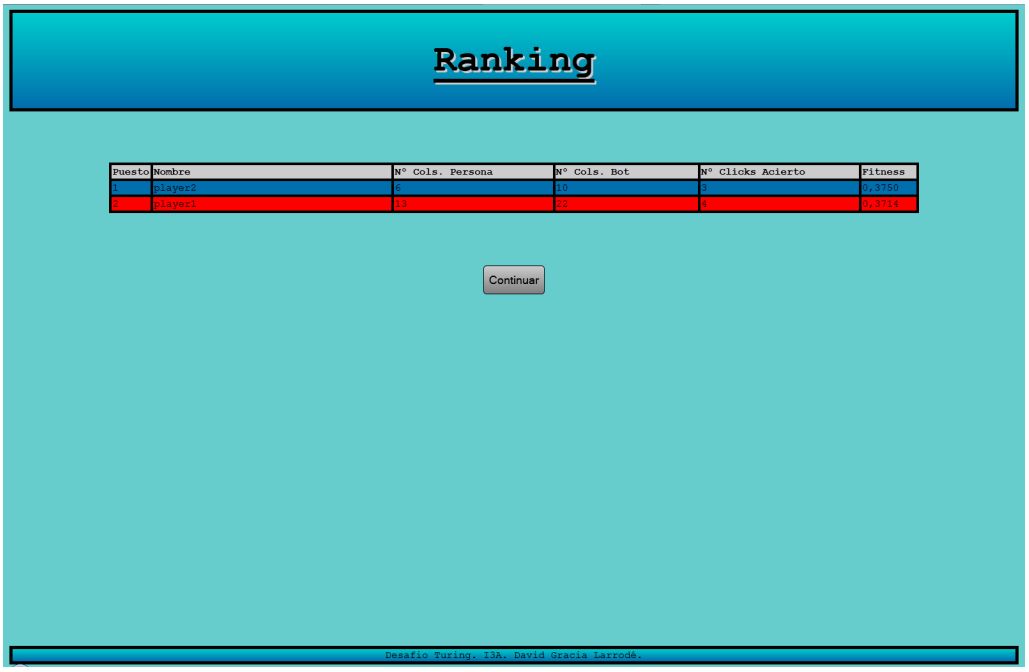

Figura B.22: Captura de la página del ranking en modo múltiple ordenado por fitness.

Figura B.23: Captura de la página del cuestionario.

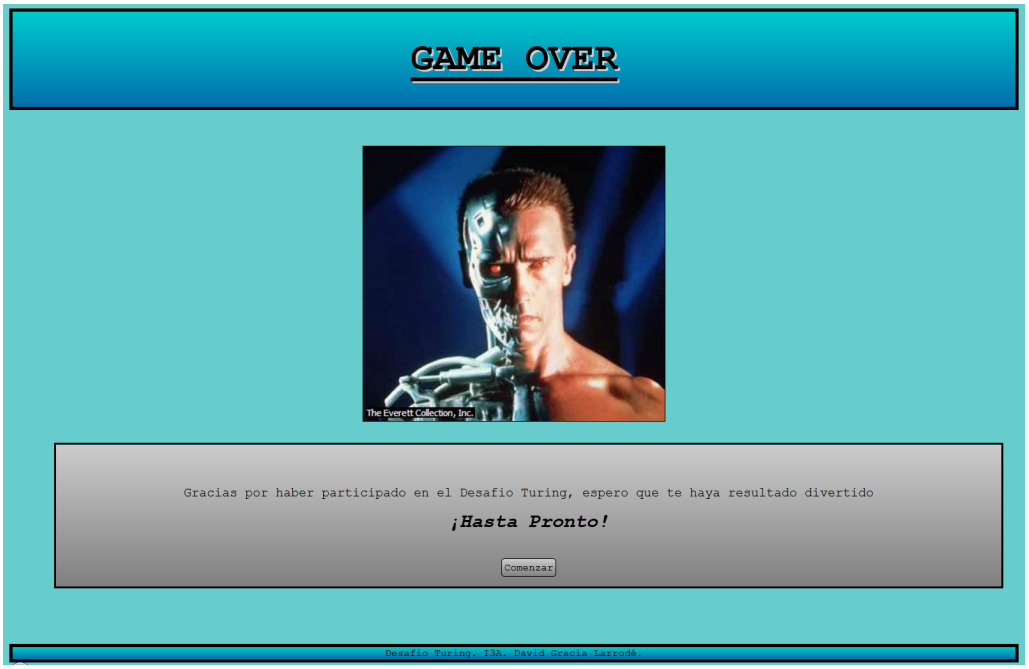

Figura B.24: Captura de la página final de la aplicación.

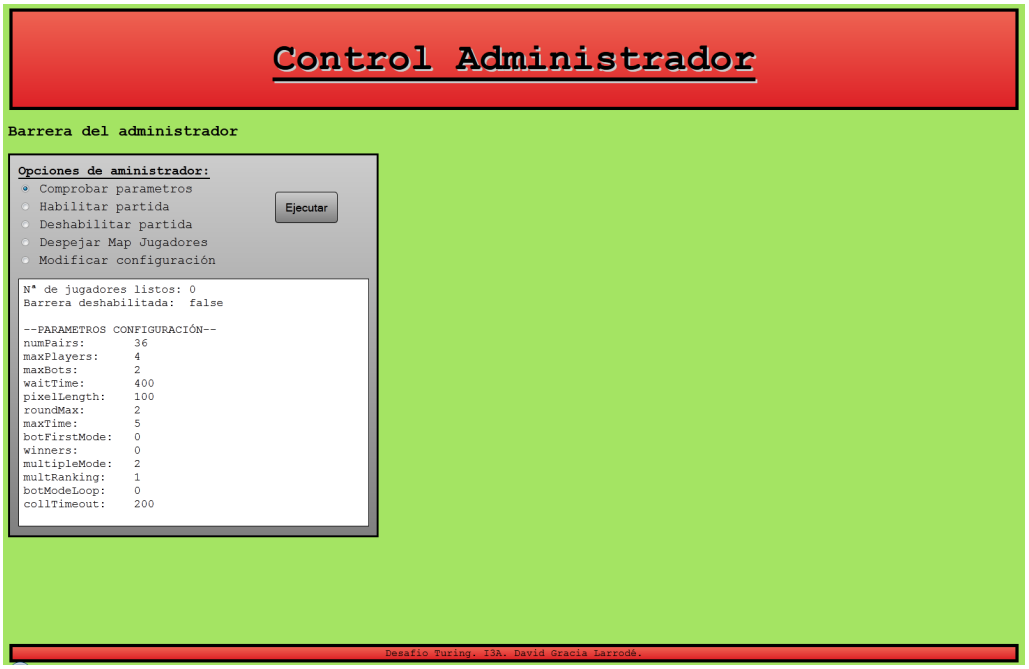

Figura B.25: Captura de la página de administración.

Anexo C

Gráficas

C.1. Visor gráficas

Como complemento a la aplicación principal del proyecto he desarrollado un conjunto de visores de gráficas que permiten analizar los datos recogidos durante los experimentos. Para construir estos interfaces he utilizado la herramienta EJS (*Easy Java Simulations*). *Easy Java Simulations* es una herramienta de software diseñada para la creación de simulaciones discretas por computador. Esto significa que EJS es un programa que ayuda a crear otros programas; más precisamente, simulaciones científicas. Ha sido concebido para personas que están más interesadas en el contenido de la simulación, en el fenómeno mismo que se simula, que en los aspectos técnicos necesarios para construir la simulación. EJS crea aplicaciones Java que son independientes y multiplataforma, o applets que se pueden visualizar usando cualquier navegador Web, que pueden leer datos a través de la red y ser controlados usando scripts incluidos en las páginas HTML.

A partir de los elementos gráficos ofrecidos por el programa para confeccionar interfaces de usuario, he construido los más apropiados para el estudio de resultados de los tres modos implementados en el proyecto. Utilizando la potencia que ofrece EJS para poder importar mis propias clases Java y añadir código complementario implementado por mí, he podido recorrer las trazas de datos almacenadas durante los experimentos en formato JSON y utilizar dicha información para construir gráficas útiles para su posterior estudio.

La figura C.1 es una captura de la versión para estudio de resultados en modo normal, como puede observarse el interfaz construido para este modo tiene tres zonas bien diferenciadas. En la zona superior hay un campo de entrada de texto donde introducir la ruta del fichero JSON a estudiar y un botón “Load” para cargar la primera ronda del experimento. En la zona intermedia es la zona donde se muestra la gráfica estudiada, en el eje de ordenadas se representa la posición del receptor en píxeles y en el de abscisas el instante temporal en milisegundos. La zona inferior se divide en otras cuatro zonas: la más a la izquierda contiene un botón para cargar la gráfica de la siguiente ronda del experimento y otro para cargar la anterior; la contigua muestra información de la partida, el número de experimento, nombre del jugador del que se almaceno la información, el de su contrincante y el número de colisiones activas acaecidas, los tres últimos campos tienen además un “checkbox” para hacer visible o no la traza correspondiente; la siguiente contiene un “checkbox” para hacer visible una nueva ventana donde se muestra un gráfico de fase, la cual describo en un párrafo posterior, también hay un botón para cargar el gráfico correspondiente a la ronda actual en la ventana antes mencionada; en la zona más a la derecha se muestra información editable sobre los límites de la gráfica, al cambiar dichos valores se podrá hacer un zoom en la parte de la gráfica que más interese, también contiene un botón de “Reset” que reinicializa los limites de las dimensiones al caso inicial por defecto.

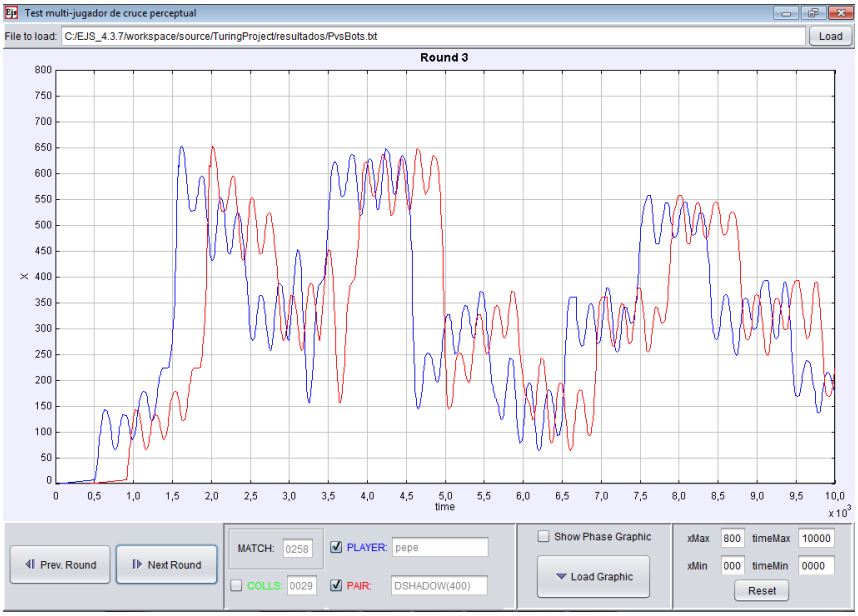

Figura C.1: Captura del visor para trazas generadas en modo normal.

La ventana del gráfico de fase, mencionada antes, es compartida por todos los visores. Como se puede observar en la figura C.2 la ventana del gráfico de fase tiene dos zonas bien diferenciadas. La zona superior contiene la gráfica, en el eje de abscisas se representa la posición del receptor en píxeles y en el de ordenadas el diferencial de la posición del cuadrado respecto del diferencial del tiempo. La zona inferior contiene a su vez dos zonas. La de la izquierda muestra el nombre del jugador del que se guardo la traza actual y el del su contrincante, también tienen un "checkbox" para hacer invisible o visible la traza de cada uno en la gráfica; en la de la derecha se muestra información editable sobre los límites de la gráfica, al cambiar dichos valores se podrá centrar la atención en la parte de la gráfica que más interese, también contiene un botón de "Reset" que reinicializa estos valores a los de por defecto.

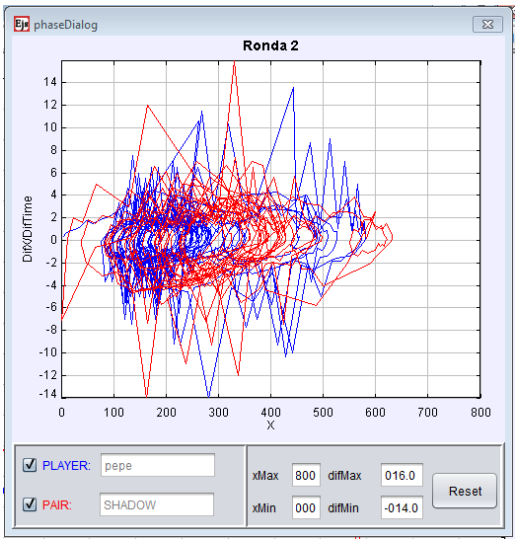

Figura C.2: Captura de la ventana de gráfico de fase.

En la figura C.3 se muestra una captura del interfaz para modo múltiple con retraso de tiempo en milisegundos. Como puede observarse este modo también tiene tres zonas bien diferenciadas. La zona superior es igual a la del otro visor, permite cargar el fichero deseado de datos a estudiar. En la zona intermedia se muestra la gráfica estudiada, al igual que el visor anterior, en el eje de ordenadas se representa la posición del receptor en píxeles y en el de abscisas el instante temporal en milisegundos, pero en este visor pueden aparecer muchas más trazas, ya que existen más interlocutores en este experimento y más parámetros a manejar. La zona inferior se divide en tres zonas: la más a la izquierda se divide en dos, la superior contiene un botón para cargar la gráfica de la siguiente ronda del experimento y otro para cargar la anterior, en la inferior se muestra información editable sobre los límites de la gráfica, al cambiar dichos valores se podrá hacer un zoom en la parte del gráfica que más interese, también contiene un botón de "Reset" que reinicializa los limites de las dimensiones al caso inicial por defecto; la zona contigua muestra información de la partida, el número de experimento, nombre del jugador del que se almaceno la información, el de su contrincante, el número de clicks total¸ los acertados y los fallidos, el número de colisiones activas total, las ocurridas con humanos y las acaecidas con bots, salvo el identificador de partida, el resto de campos tiene además un "checkbox" para hacer visible o no la traza correspondiente y los mismo para las trazas de la sombra del jugador que guardo la información actual, la de su contrincante y el valor fijo; la última contiene un "checkbox" para hacer visible una nueva ventana donde se muestra un gráfico de fase, la cual ya he descrito en un párrafo previo.

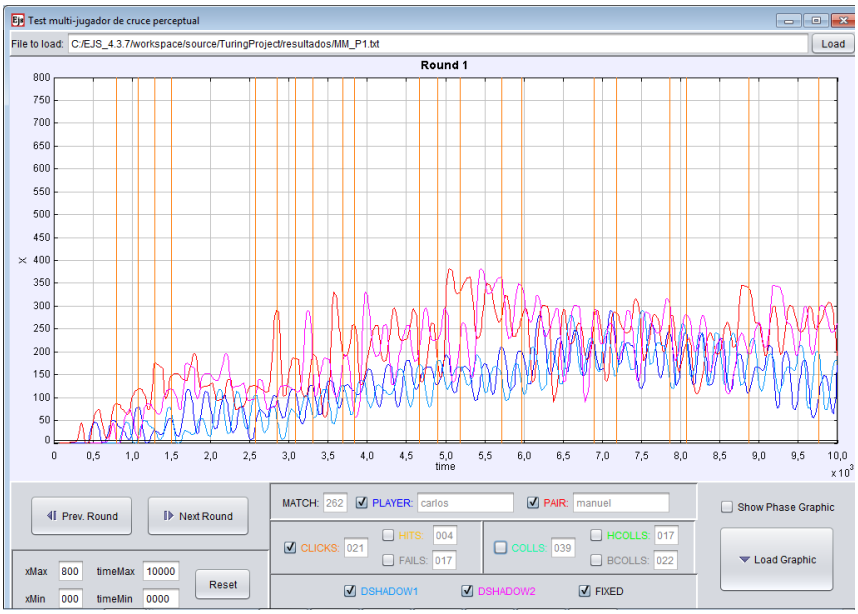

Figura C.3: Captura del visor para trazas generadas en modo múltiple con retraso de tiempo en milisegundos.

La figura C.4 contiene el último visor a estudiar, él de modo múltiple con distancia en píxeles. Como puede observarse en la imagen el interfaz de este modo es prácticamente el mismo que el del visor múltiple con retraso en milisegundos, salvo por la parte central y derecha de la zona inferior del mismo. Aparecen nuevos campos como son el del número de clicks y colisiones con la sombra del contrincante y con el valor fijo asignado al jugador que guardo la información y aparece una nueva traza de valor fijo, ya que en este modo los dos jugadores tienes diferente valor fijo asignado. Como se observa han desaparecido el número de clicks fallo y de colisiones con bots en total, ya que son parámetros que se pueden calcular a partir del valor de los incluidos. La zona más a la derecha queda dividida en dos zonas, la superior con el "checkbox" para mostrar la ventana del gráfico de fase y el botón de carga del gráfico en sí mismo; en la zona inferior un "checkbox" para mostrar

otra ventana con una tabla de ratios, como ejemplo la ventana de la figura C.5, contiene una tabla con tres filas y tres columnas. La columnas representan a cada uno de los jugadores adversario, el jugador persona (PAIR), la sombra del mismo (SHADOW2) y el valor fijo (FIXED). Las filas representan las ratios a calcular, la primera el porcentaje de colisiones activas con el jugador-columna respecto del total, la segunda el porcentaje de clicks acertados con el jugador-columna respecto del total y la tercera el cociente del número de clicks acertados con el jugador-columna entre en número de colisiones activas con el mismo.

Figura C.4: Captura del visor para trazas generadas en modo múltiple con distancia en píxeles.

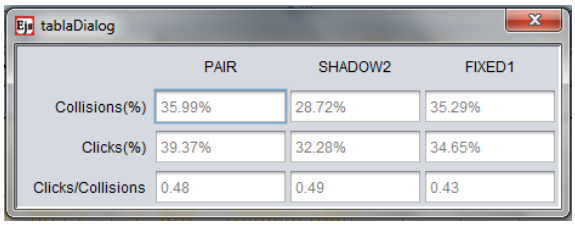

Figura C.5: Captura de la ventana de tabla de ratios en modo múltiple con distancia en píxeles.

C.2. Gráficas EJS

C.2.1. Gráficas posición receptor - tiempo

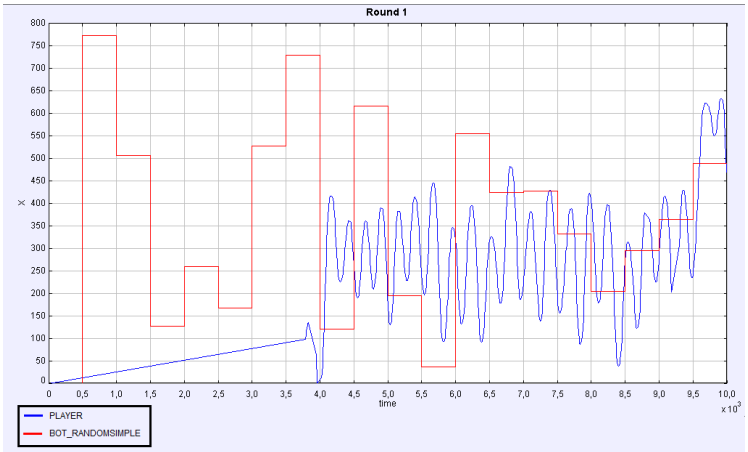

Figura C.6: Modo normal: Persona vs Bot RANDOMSIMPLE

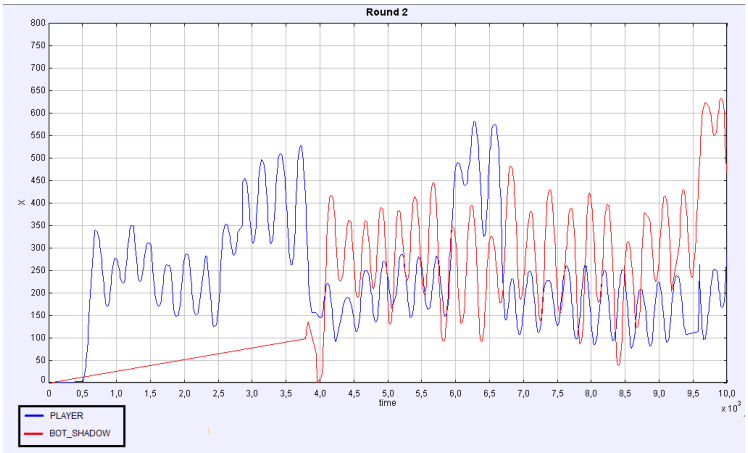

Figura C.7: Modo normal: Persona vs Bot SHADOW

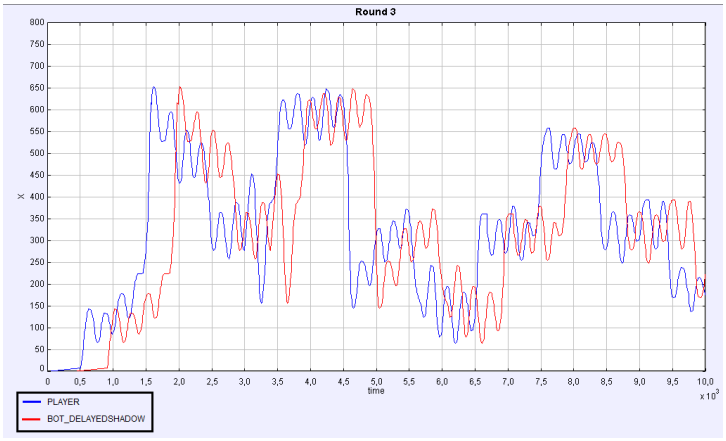

Figura C.8: Modo normal: Persona vs Bot DELAYEDSHADOW

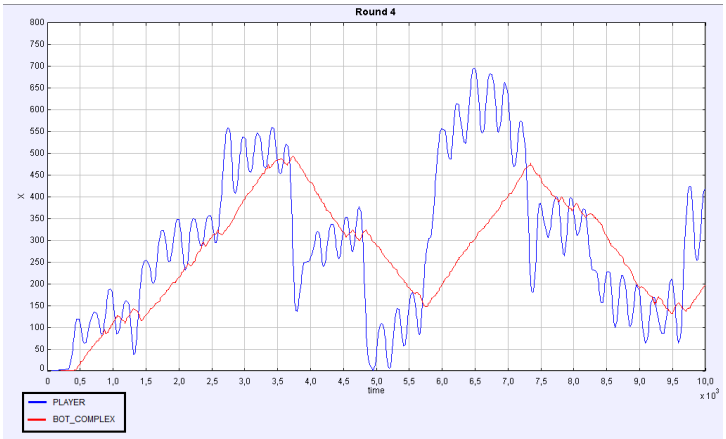

Figura C.9: Modo normal: Persona vs Bot COMPLEX

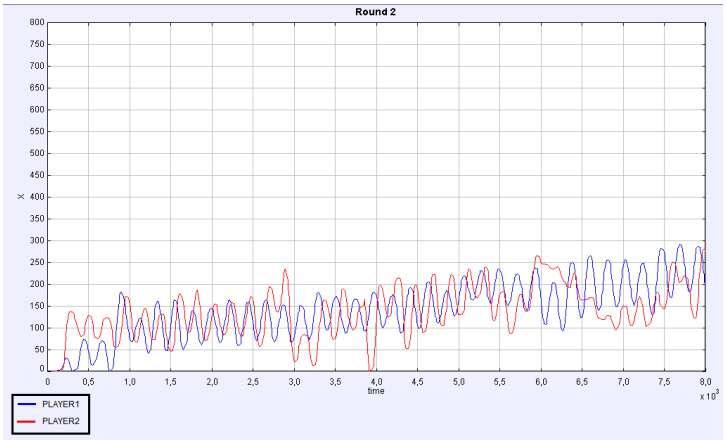

Figura C.10: Modo normal: Persona vs Persona

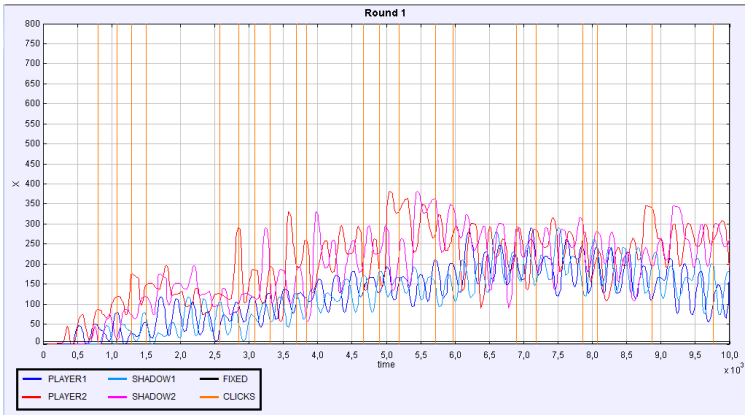

Figura C.11: Modo Múltiple: Retardo de sombra en milisegundos

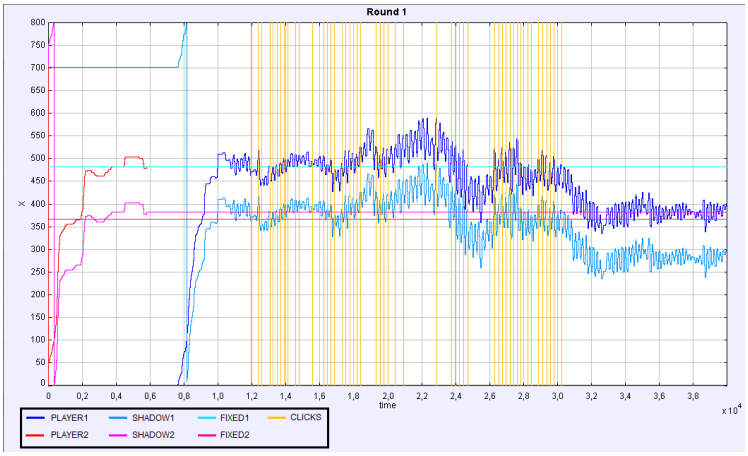

Figura C.12: Modo Múltiple: Retardo de sombra en píxeles

C.2.2. Gráficas de fase

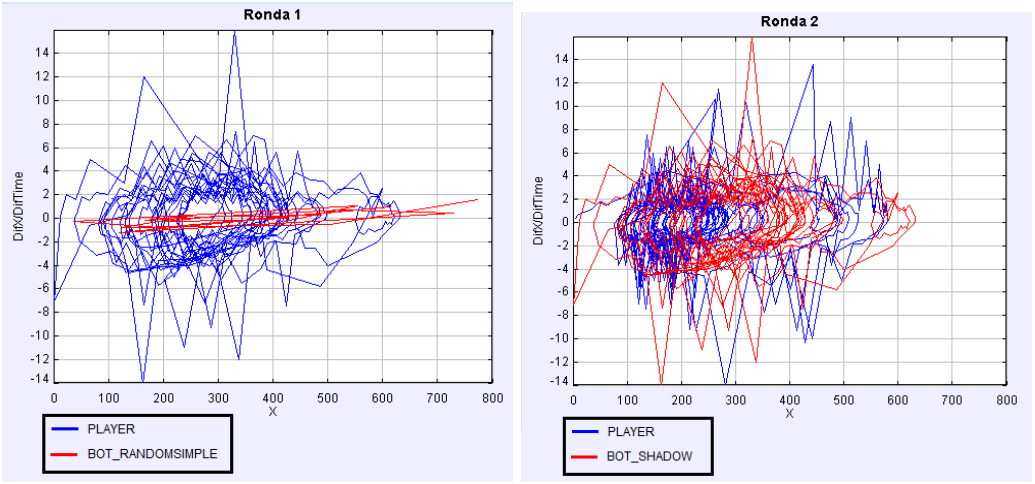

Figura C.13: Modo normal: Persona vs Bot RANDOMSIMPLE (**izquierda**) y Persona vs Bot SHADOW (**derecha**)

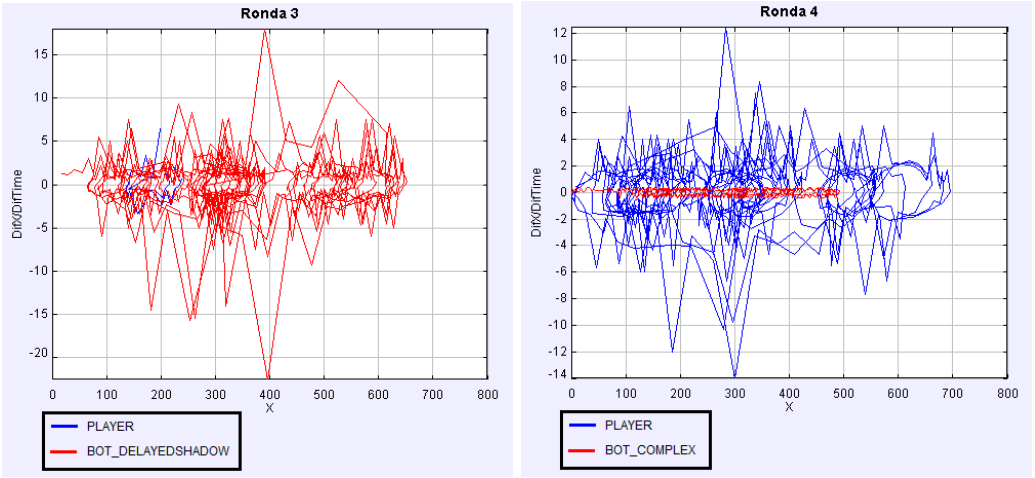

Figura C.14: Modo normal: Persona vs Bot DELAYEDSHADOW (**izquierda**) y Persona vs Bot COMPLEX (**derecha**)

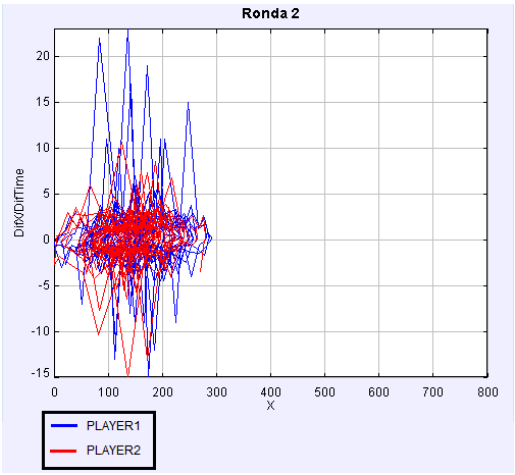

Figura C.15: Modo normal: Persona vs Persona

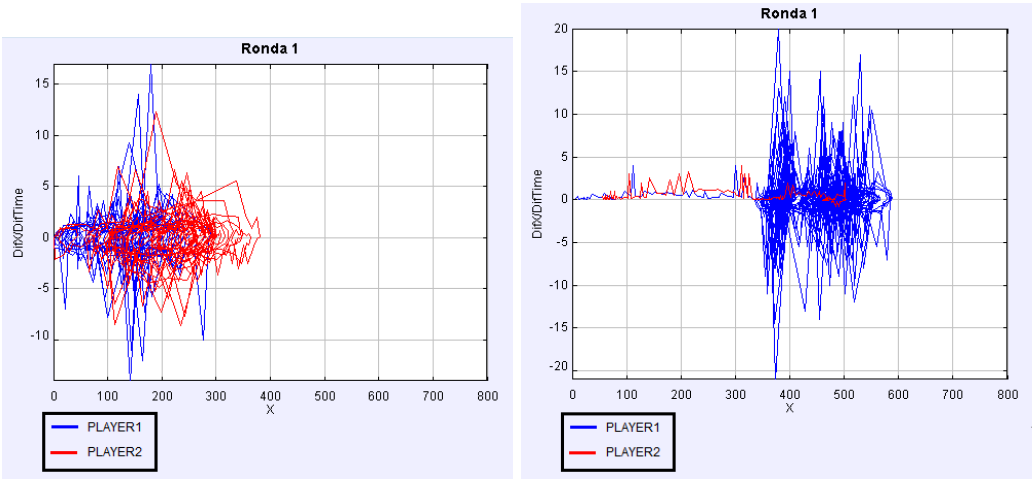

Figura C.16: Múltiple: Retardo de sombra en milisegundos (**izquierda**) y Retardo de sombra en píxeles (**derecha**)

Anexo D

Datos en formato JSON

Los datos intercambiados entre cliente y servidor WebSocket, según el trabajo que desempeñan, se pueden dividir en datos de configuración de la aplicación, de emparejamiento de jugadores durante las rondas, del estado del cliente, de errores producidos, de información útil de la ronda que se almacena, del cuestionario del modo normal y de movimiento del cuadrado receptor.

D.1. Configuración

En la parte de configuración he utilizado dos estructuras JSON distintas. Una de ellas es la de los mensajes que se envían al principio de cada vuelta de rondas a los clientes para que puedan configurar sus variables clave, como son el número de rondas de la vuelta, la duración de cada ronda, el modo de juego, el tipo de ranking y la sensibilidad ante clicks tras una colisión. La tabla D.1 contiene la estructura de un mensaje de tipo *configurationInfo*.

La otra estructura de configuración es el contenido del fichero de configuración "`configuration.json`", un fichero de texto que determina los parámetros clave para que la aplicación funcione de una manera u otra. Este fichero puede ser modificado bien de forma manual o a través de la pagina de administración "`adminBarrier.html`". De forma automática también es modificado al terminar una vuelta de rondas completa, ya que se actualiza el valor del parámetro *numPairs*, almacenando la id del próximo emparejamiento a repartir. Si se produce una segunda vuelta de rondas debido a que el valor del parámetro *winners* es mayor que cero, la propia página de ranking debe modificar en número de jugadores necesarios, actualizando el valor de *maxPlayers* y de *maxBots* en el fichero de configuración. En la tabla D.2 puede observarse la estructura del fichero.

`{"configurationInfo":{`	//Nombre identificativo del tipo de mensaje.
`"roundMax":valorRoundMax,`	//Número de rondas del experimento.
`"maxTime":valorMaxTime,`	//Tiempo en segundos de cada ronda.
`"multipleMode":valorMultipleMode,`	//Indica el modo en que se juega la vuelta. Si es igual a 0 modo //normal, si es 1 modo múltiple(tiempo) y si es 2 modo //múltiple(píxel).
`"multRanking":valorMultRanking,`	//En modo múltiple indica el tipo de ranking a generar tras una //vuelta completa de rondas.
`"collTimeout":valorCollTimeout}}`	//Indica el tiempo máximo en milisegundos en el cual, tras //producirse una colisión, si se hace click se considera colisión.

Tabla D.1: Estructura del tipo de mensaje *configurationInfo*, con comentarios de sus componentes.

{"numPairs":valorNumPairs,	//Identificador numérico del próximo emparejamiento entre //jugadores.
"maxPlayers":valorMaxPlayers,	//Número de jugadores (Personas + Bots).
"maxBots":valorMaxBots,	//Número de bots.
"waitTime":valorWaitTime,	//Tiempo en milisegundos del retardo del bot //DELAYEDSHADOW.
"pixelLength":valorPixelLength,	//Longitud en píxeles de distancia del bot PIXELSHADOW.
"roundMax":valorRoundMax,	//Número de rondas del experimento.
"maxTime":valorMaxTime,	//Tiempo en segundos de cada ronda.
"botFirstMode":valorBotFirstMode,	//Indica el tipo inicial de un bot cuando se crea.
"winners":valorWinners,	//Indica cuantos jugadores persona pasan a una segunda vuelta //de rondas, si es igual a 0 solo hay una vuelta.
"multipleMode":valorMultipleMode,	//Indica el modo en que se juega la vuelta. Si es igual a 0 modo //normal, si es 1 modo múltiple(tiempo) y si es 2 modo //múltiple(píxel).
"multRanking":valorMultRanking,	//En modo múltiple indica el tipo de ranking a generar tras una //vuelta completa de rondas.
"botModeLoop":valorBotModeLoop,	//Indica si el tipo de los bots cambia a lo largo de la vuelta o //siempre es el mismo.
"collTimeout":valorCollTimeout}	//Indica el tiempo máximo en milisegundos en el cual, tras //producirse una colisión, si se hace click se considera colisión.

Tabla D.2: Estructura del fichero `configuration.json`, con comentarios de sus componentes.

D.2. Emparejamiento

Dentro de esta sección solo se utiliza una estructura, al comienzo de cada ronda, el tipo de mensajes enviados desde servidor a los clientes para transmitirles el identificador de emparejamiento de esa ronda y un valor fijo solo utilizable en modo múltiple. La tabla D.3 contiene la estructura de un mensaje de tipo *pairingInfo*.

{"pairingInfo":{	//Nombre identificativo del tipo de mensaje.
"pairId":valorPairId,	//Identificador numérico del emparejamiento asociado.
"fixedX":valorFixedX}}	//Valor fijo a utilizar como tercer jugador en modo múltiple.

Tabla D.3: Estructura del tipo de mensaje *pairingInfo*, con comentarios de sus componentes.

D.3. Estado

Durante el transcurso de una ronda entre el servidor y el cliente se intercambian información sobre estado del cliente. Si dicha información tiene como fuente el cliente sirve para hacer saber al servidor en que estado se encuentra el cliente y actuar en consecuencia. Si el sentido es el contrario sirve para indicar al cliente que debe pasar al estado que le indica el servidor y con ello desencadenar los procesos asociados al mismo. En la tabla D.4 se observa la estructura de un mensaje de tipo *statusInfo*.

`{"statusInfo":{`	//Nombre identificativo del tipo de mensaje.
`"user":valorUser,`	//Nombre del cliente destino o fuente.
`"status":valorStatus}}`	//Valor numérico que representa el tipo de estado.

Tabla D.4: Estructura del tipo de mensaje *statusInfo*, con comentarios de sus componentes.

D.4. Error

Este tipo de mensajes se envían del servidor al cliente, para que muestre de forma gráfica al usuario el tipo de error que se ha producido. Esto ocurre cuando en una ronda entre jugadores humanos uno de ellos pierde la conexión o se desconecta mientras están jugando, el servidor detecta que uno de los participantes se ha desconectado y el otro sigue jugando así que le envía un mensaje a su cliente para hacérselo saber y que se recupere del contratiempo, este error lleva asociado el código numérico 100. La otra posibilidad es cuando se intenta conectar un jugador con el mismo id de conexión que otro ya conectado, se le rechaza y envía un mensaje de error informándole de lo sucedido, este error lleva asociado el código 101. En la tabla D.5 puede verse la estructura de un mensaje de tipo *errorInfo*.

`{"errorInfo":{`	//Nombre identificativo del tipo de mensaje.
`"cod":valorCod,`	//Código numérico del tipo de error.
`"msg":valorMsg}}`	//Mensaje de texto asociado al error.

Tabla D.5: Estructura del tipo de mensaje *errorInfo*, con comentarios de sus componentes.

D.5. Información almacenada

Después de jugar una ronda, cada jugador envía al servidor un mensaje con información recogida en dicha ronda. El servidor va almacenado los diferentes mensajes de una vuelta completa en un fichero de texto, que estará formado por esos mensajes, el nombre de este fichero está constituido por el nombre del jugador del que se recoge información concatenado con la fecha y hora en el instante de conectarse al servidor, además en modo múltiple tiene el prefijo "MT", en el caso de retardo en milisegundos, o "MP," en el caso de distancia en píxeles. Según el modo de juego en el que se desarrolla una ronda se almacenará un tipo de mensaje de información de ronda u otra, en modo normal será de tipo *roundInfo* y en modo múltiple *roundMMInfo*. La principal diferencia entre ambas estructuras es que en modo normal solo hay un oponente y en modo múltiple se tiene tres oponentes, por ello difieren en los campos relacionados con este hecho. Además en el caso normal aparecen campos relacionados con la pregunta final que el múltiple no posee, por el contrario en el modo múltiple aparece un campo en relación con los clicks realizados por el jugador que no contiene el modo normal. La principal diferencia entre los modos múltiples es el tipo de bot sombra, en el caso con retardo en milisegundos es un DELAYEDSHADOW y en el de distancia en píxeles un PIXELSHADOW. En la tablas D.6 y D.7 se ve la estructura de *roundInfo* y *roundMMInfo* respectivamente

{"roundInfo":{	//Nombre identificativo del tipo de mensaje.
"matchId":valorMatchId,	//Identificador numérico del emparejamiento asociado a la //ronda.
"player":valorPlayer,	//Nombre del jugador del que se almacena la información de //ronda.
"playerId":valorPlayerId,	//Identificador del jugador del que se almacena la //información de ronda.
"pairPlayer":valorPairPlayer,	//Nombre del jugador oponente.
"pairPlayerId":valorPairPlayerId,	//Identificador del jugador oponente.
"round":valorRound,	//Número de ronda del experimento actual.
"activeCollisions":valorActiveCollisions,	//Número de colisiones activas.
"pasiveCollisions":valorPasiveCollisions,	///Número de colisiones pasivas.
"botDelay":valorBotDelay,	//Milisegundos de retardo, si el oponente es un bot en modo //DELAYEDSHADOW.
"randomList":[valorRandomList],	//Lista de posiciones del receptor oponente, si es un bot en //modo RANDOMSIMPLE.
"shadowFile":valorShadowFile,	//URI local del fichero en el cual se almacenó la información //de partida del oponente, si es un bot en modo SHADOW.
"complexList":[valorComplexList],	//Lista de pares posición del receptor e instante de tiempo //del oponente, si es un bot en modo COMPLEX.
"pairFile":valorPairfile	//URI local del fichero en el cual se esta almacenando la //información de partida actual del oponente, si es un //jugador persona.
"maxTime":valorMaxTime	//Tiempo en segundos de la ronda.
"startTime":valorStartTime	//Instante de tiempo inicial, a partir del cual se comienzan a //enviar mensajes de movimiento.
"list":[valorList]	//Lista de ternas instante temporal, posición del cuadrado //receptor y booleano de colisión activa, que representa la //traza realizada por el jugador del que se almacena la //información de ronda.
"reply":valorReply	//Respuesta del jugador del que se almacena la información //de ronda a la pregunta de fin de ronda, puede ser "per" o //"maq".
"success":valorSuccess}}	//"Acierto" si la respuesta es correcta y "Fallo" si es incorrecta.

Tabla D.6: Estructura del fichero *roundInfo*, con comentarios de sus componentes.

{"roundMMInfo":{	//Nombre identificativo del tipo de mensaje.
"multipleMode":valorMultipleMode,	//Tipo de modo múltiple, "TIME" o "PIXEL".
"matchId":valorMatchId,	//Identificador numérico del emparejamiento asociado a la //ronda.
"player":valorPlayer,	//Nombre del jugador del que se almacena la información de //ronda.
"playerId":valorPlayerId,	//Identificador del jugador del que se almacena la //información de ronda.
"humanPlayer":valorHumanPlayer,	//Nombre del jugador persona oponente.
"humanPlayerId":valorHumanPlayerId,	//Identificador del jugador persona oponente.
"dShadowBot":valorDShadowBot,	//Nombre del jugador oponente bot sombra de tipo //DELAYEDSHADOW o PIXELSHADOW.
"round":valorRound,	//Número de ronda del experimento actual.
"activeCollisions":valorActiveCollisions,	//Número total de colisiones activas.
"humanActiveCollisions":valorHActColls,	//Número de colisiones activas con oponente jugador //persona.
"botActiveCollisions":valorBActColls,	//Número de colisiones activas con oponentes no persona.
"shadowActiveCollisions":valorSActColls,	//Número de colisiones activas con oponente bot sombra.
"fixedActiveCollisions":valorFActColls,	//Número de colisiones activas con oponente valor fijo.
"pasiveCollisions":valorPasiveCollisions,	///Número de colisiones pasivas.
"botDelay":valorBotDelay,	//Milisegundos de retardo o píxeles de distancia del oponente //bot sombra.
"botFixedX":valorBotFixedX,	//Posición del cuadrado receptor del jugador fijo.
"pairFile":valorPairfile	//URI local del fichero en el cual se esta almacenando la //información de partida actual del oponente jugador //persona.
"maxTime":valorMaxTime	//Tiempo en segundos de la ronda.
"startTime":valorStartTime	//Instante de tiempo inicial, a partir del cual se comienzan a //enviar mensajes de movimiento.
"dataList":[valorDataList]	//Lista de cuaternas instante temporal, posición del //receptor, booleano de colisión activa y cadena de //identificadores de jugadores oponentes en caso de //colisionar, que representa la traza realizada por el jugador //del que se almacena la información de ronda.
"clicksList":[valorClicksList]}}	//Lista de pares instante temporal del click, cadena de //identificadores de jugadores oponentes en caso de //colisionar, que representa la traza de clicks realizada por el //jugador del que se almacena la información de ronda.

Tabla D.7: Estructura del fichero *roundMMInfo*, con comentarios de sus componentes.

D.6. Cuestionario

A la hora de contestar el cuestionario que aparece tras jugar una ronda en modo normal, el cliente envía un mensaje de este tipo para informar lo que piensa el jugador al servidor, este comprueba si la respuesta es correcta o errónea y responde con un mensaje del mismo tipo al cliente, que mostrará gráficamente el resultado de la comprobación. La tabla D.8 contiene la estructura de un mensaje de tipo *replyInfo*.

`{"replyInfo":{`	//Nombre identificativo del tipo de mensaje
`"idPlay":valorIdPlay,`	//Identificador numérico del emparejamiento asociado.
`"reply":valorReply}}`	//Contiene la respuesta del cliente o la comprobación del servidor.

Tabla D.8: Estructura del tipo de mensaje *replyInfo*, con comentarios de sus componentes.

D.7. Movimiento

Durante la duración de una ronda, valor marcado por el parámetro *maxTime*, los jugadores intercambian mensajes de posición de su cuadrado receptor, estando el servidor de intermediario. Uno de los campos del mensaje es *indice*, que identifica al jugador emisor en el cliente receptor, en modo normal siempre vale 0, en modo múltiple tiene dos posibles valores 0 para el jugador persona y 1 para su sombra. En la tabla D.9 puede observarse la estructura de un mensaje de tipo *motionInfo*.

`{"motionInfo":{`	//Nombre identificativo del tipo de mensaje.
`"indice":indice,`	//Identificador numérico del jugador.
`"x":valorX,`	//Valor numérico en píxeles de la posición del cuadrado receptor
	//del jugador emisor.
`"numMsg":valorNumMsg}}`	//Número de mensaje de movimiento.

Tabla D.9: Estructura del tipo de mensaje *motionInfo*, con comentarios de sus componentes.

Anexo E

Medición de prestaciones

E.1. Experimentos a realizar

Van a ser 10 rondas seguidas de 10 segundos cada una por experimento:

MODO NORMAL

- Exp1: Máq remota(persona) vs Bot RandomSimple (10)
- Exp2: Máq remota(persona) vs Bot Shadow (10)
- Exp3: Máq remota(persona) vs Bot DelayedShadow (50)
  - Exp3.1: waitTime = 100 ms (10)
  - Exp3.2: waitTime = 200 ms (10)
  - Exp3.3: waitTime = 300 ms (10)
  - Exp3.4: waitTime = 400 ms (10)
  - Exp3.5: waitTime = 500 ms (10)
- Exp4: Máq remota(persona) vs Bot Complex (10)
- Exp4b[1]: Máq remota(persona) vs Bot Complex (10)
- Exp5: Máq remota(persona) vs Máq remota(persona) (10)

MODO MÚLTIPLE (TIEMPO)

- Exp6: [Máq remota(persona) + Bot DelayedShadow + Fixed] vs [Máq remota(persona) + Bot DelayedShadow + Fixed] (50)
  - Exp6.1: waitTime = 100 ms (10)
  - Exp6.2: waitTime = 200 ms (10)
  - Exp6.3: waitTime = 300 ms (10)
  - Exp6.4: waitTime = 400 ms (10)
  - Exp6.5: waitTime = 500 ms (10)

[1]Igual que el experimento 4 pero llevando traza del bot para luego poder representar la partida de forma gráfica, habrá que ver si merece la pena la sobrecarga de trabajo al medir los tiempos.

MODO MÚLTIPLE (PÍXEL)

- Exp7: [Máq remota(persona) + Bot PixelShadow + Fixed1] vs [Máq remota(persona) + Bot PixelShadow + Fixed2] (10)

Total de rondas a jugar = 160

E.2. Datos a medir y metodología a seguir

Los datos a medir siempre van a hacer referencia a la partida y trazas de los jugadores persona. Habrá que añadir instrumentación para que de forma gráfica se puede obtener tras una ronda el número de mensajes recibidos y enviados y un tiempo de juego que se puede tomar del máximo del crono o con la diferencia de un par de estampillas temporales al inicio y fin de la ronda por ser más meticulosos. Estos datos irán a una hoja de cálculo donde se sacarán mensajes enviados/recibidos por unidad de tiempo. Luego en modo *off-line* con un programa habilitado al caso se hará un análisis de las trazas de las partidas, allí se medirán las diferencias de tiempo entre un mensaje enviado y el siguiente, para calcular la velocidad máxima media de envío, comprobando la sensibilidad del escuchador de captura del evento de movimiento de ratón, ya que, por cada captura se envía un mensaje. Estos datos también se almacenarán en una hoja de calculo para su posterior estudio.

E.3. Metodología de juego durante una ronda

Se modificará la evolución normal en los modos de los bots (0,1,2,3) en el modo normal, para que se puedan hacer todas las mediciones del modo a estudiar de forma seguida, lo que resulta más práctico y simple. En el caso especial del modo SHADOW (1) que es dependiente de una partida anterior para poder jugarla se le pondrá siempre la misma partida.

Durante una partida se seguirá una estrategia por la cual se moverá el ratón de un lado a otro los más rápido posible, para que los datos recogidos sean lo más fidedignos posible a la idea de medir los límites de la tecnología utilizada para implementar la aplicación.

Los nombres de los ficheros a almacenar empezaran por "e" seguido del número de experimento, en el caso del 3 y el 6 de dos números, además si hay más de un jugador persona, en el caso del 5, el 6 y el 7, se añade p1 para el primer jugador y p2 para el segundo. Ejemplos de nombre: e1_18-3-2013_5h58m11s.txt, e2_18-3-2013_6h02m19s.txt, e33_18-3-2013_6h30m08s.txt, e65p2_18-3-2013_8h02m39s.txt ...

E.4. Equipos de prueba

PC1	
S.O	Windows 7 Professional, Service Pack 1
Procesador	Intel(R) Core(TM) i7-2600 CPU @3.40GH
RAM	6.00 GB
Arquitectura	64 bits

Tabla E.1: Tabla de características del PC1.

PC2	
S.O	Windows 7 Professional, Service Pack 1
Procesador	Dual Core AMD Opteron(tm) Processor 275 2.20GHz (2 procesadores)
RAM	3.70 GB
Arquitectura	64 bits

Tabla E.2: Tabla de características del PC2.

En el PC1 se han hecho las pruebas en modo normal contra bots y representa al jugador 1 en las partidas con más de una persona. El PC2 representa al segundo jugador persona.

E.5. Medidas

Estas son las medidas obtenidas de cada experimento, como puede observarse cada tabla tiene tres datos de interés como son, en orden descendente, la media de mensajes enviados por segundo, la media de mensajes recibidos por segundo y la diferencia media total entre mensajes enviados. Algo que observé, durante la toma de medidas, es que la primera ronda de todos los experimentos tiene una tasa de mensajes enviados mucho mayor que el resto de rondas, lo achaco a que en el cliente en primera instancia no hay memoria reservada todavía para las estructuras de datos que sirven para almacenar la información de una ronda y demás datos auxiliares, que se van creando precisamente en esta primera partida. Por ello, considero el primer dato de cada experimento como un dato atípico que podría mejorar las prestaciones temporales de la aplicación notablemente, no lo he incluido en la medias calculadas para que mis datos sean más fidedignos y se centren en el grueso de información más representativa, ya que en las siguientes rondas de los experimentos los tiempos calculados no difieren tanto unos de otros.

E.5.1. Modo normal

En modo normal la media de mensajes enviados por segundo mínima es de 53,0603 y la máxima 60,7565, lo que nos hace movernos entre 17,84 y 16,46 milisegundos por mensaje enviado. En el caso de la media de mensajes recibidos por segundo hay una máxima de 88,7562 en el caso del bot SHADOW y una mínima de 1,9983 en el bot RANDOMSIMPLE, estos datos no son realmente significativos, ambos modos tienen una traza constante, solo envían mensajes, no los reciben, siempre van a darse en las mismas cantidades salvo latencia añadida por el medio de comunicación, además el bot SHADOW tiene la traza de una primera ronda por eso ese valor tan alto. Si observamos el resto de experimentos, no constantes, se tiene una máxima de 60,4881 y una mínima de 45,1400. Para el tercer parámetro en estudio, la diferencia media entre mensajes enviados, tiene un máximo de 19,0702 milisegundos entre mensaje y mensaje enviado y un mínimo de 16,5686 milisegundos entre mensajes enviados, como se observa, son valores similares a los obtenidos a través de la media de mensajes enviados por segundo, dos formas de calcular la velocidad de envió de la aplicación.

En el experimento 4 y 4b, quería comprobar si añadir el guardado de la traza del bot COMPLEX, durante una ronda, para luego poder representar esta traza en el visor de gráficas correspondiente, merecía la pena y no añadía una latencia significativa. Como se puede comprobar en los datos de la figura E.3 los valores obtenidos son semejantes, incluso algo mejores en el caso 4b, lo que justifico con un mejor comportamiento del servidor y/o del jugador persona.

Media MensEnv/seg	52,5563		Media MensEnv/seg	54,4880	
Media MensRec/seg	1,9983		Media MensRec/seg	88,7562	
Diferencia media total (ms)	19,0702		Diferencia media total (ms)	18,5237	
Exp1			Exp2		

Figura E.1: Medidas obtenidas en experimento 1 y experimento 2.

Media MensEnv/seg	53,4242		Media MensEnv/seg	54,6855		Media MensEnv/seg	53,3185	
Media MensRec/seg	48,2071		Media MensRec/seg	48,1797		Media MensRec/seg	47,3890	
Diferencia media total (ms)	18,8286		Diferencia media total (ms)	18,2846		Diferencia media total (ms)	18,7439	
Exp3.1			Exp3.2			Exp3.3		
Media MensEnv/seg	54,0903		Media MensEnv/seg	53,7642				
Media MensRec/seg	47,2625		Media MensRec/seg	45,1400				
Diferencia media total (ms)	18,1750		Diferencia media total (ms)	18,5810				
Exp3.4			Exp3.5					

Figura E.2: Medidas obtenidas en experimento 3.

Media MensEnv/seg	54,0603		Media MensEnv/seg	54,4890	
Media MensRec/seg	60,4777		Media MensRec/seg	61,6374	
Diferencia media total (ms)	18,4804		Diferencia media total (ms)	18,3122	
Exp4			Exp4b		

Figura E.3: Medidas obtenidas en experimento 4 y experimento 4b.

Media MensEnv/seg	53,8122		Media MensEnv/seg	60,7565		Media MensEnv/seg	57,2843	
Media MensRec/seg	60,4881		Media MensRec/seg	53,8240		Media MensRec/seg	57,1560	
Diferencia media total (ms)	18,5923		Diferencia media total (ms)	16,5686		Diferencia media total (ms)	17,5805	
Exp5(p1)			Exp5(p2)			Exp5(media)		

Figura E.4: Medidas obtenidas en experimento 5.

Uno de los casos más interesantes dentro de este modo es el tercer experimento, en el bot DELAYEDSHADOW, aparecen las mayores latencias, tanto por el hecho de que el servidor espera la traza del jugador persona, para reenviarla con un retraso marcado, como por el propio retraso generado en el servidor a través de *sleeps*. Como se ha observado antes la máxima diferencia entre mensajes enviados es, redondeando, de unos 20 milisegundos, lo que necesitamos para completar el ciclo de ida, de cliente al servidor, y vuelta, del servidor al cliente, es este segundo camino. Como se observa en la figura E.5 se muestra una tabla con la media de mensajes recibidos entre el tiempo real. Llamo tiempo real al tiempo total de la ronda menos el retraso correspondiente del experimento. Se obtiene una media de 48,6905 mensajes recibidos por segundo, por lo que se están recibiendo mensajes del servidor cada 20,5379 milisegundos. Sumando este tiempo con el anterior obtenemos un parámetro clave de la aplicación, lo que le cuesta a un mensaje enviado desde un cliente fuente llegar a un cliente destino, es decir, cuando dos usuarios estén jugando una ronda, los mensajes que le llegan a uno desde el otro llevarán como máximo una desincronización de 40-50 milisegundos.

	Media MensRec/TReal
EXP3.1 – waitTime = 100ms	48,6935
EXP3.2 – waitTime = 200ms	49,1621
EXP3.3 – waitTime = 300ms	48,8536
EXP3.4 – waitTime = 400ms	49,2301
EXP3.5 – waitTime = 500ms	47,5130
Media Total	48,6905
msegMensRec	20,5379

Figura E.5: Tabla de mensajes recibidos en tiempo real del experimento 3.

E.5.2. Modo múltiple

Como puede observarse en las tablas una de las diferencias entre el modo normal y el múltiple, es que el volumen de mensajes recibidos por segundo es prácticamente el doble, debido al hecho de que cada jugador humano esta recibiendo dos flujos de datos pertenecientes a un jugador persona y un bot sombra. Se tiene un máximo de 62,9977 mensajes enviados por segundo y un mínimo de 32,7985, por lo que nos movemos entre los 15,87 y los 30,49 milisegundos por mensaje enviado. El mínimo anterior lo achaco a un problema con la superficie por la que se movía el ratón, que ofrecía más resistencia que en anteriores pruebas. En cuanto al número medio de mensajes recibidos por segundo, hay un máximo de 119,8763 y un mínimo de 92,5227, como he indicado antes casi se duplican los datos del modo normal. El tiempo entre mensajes enviados se mueve entre 16,0804 y 31,1188 milisegundos, este segundo dato coincide con el peor caso de mensajes enviados por segundo, motivado por la misma causa.

Media MensEnv/seg	56,0645	Media MensEnv/seg	56,4494	Media MensEnv/seg	56,2569
Media MensRec/seg	107,6231	Media MensRec/seg	107,6667	Media MensRec/seg	107,6449
Diferencia media total (ms)	17,9398	Diferencia media total (ms)	18,3906	Diferencia media total (ms)	18,1652
Exp6.1(p1)		Exp6.1(p2)		Exp6.1(media)	
Media MensEnv/seg	54,5050	Media MensEnv/seg	62,9977	Media MensEnv/seg	58,7514
Media MensRec/seg	119,8763	Media MensRec/seg	104,9429	Media MensRec/seg	112,4096
Diferencia media total (ms)	18,4454	Diferencia media total (ms)	16,0804	Diferencia media total (ms)	17,2629
Exp6.2(p1)		Exp6.2(p2)		Exp6.2(media)	
Media MensEnv/seg	56,4058	Media MensEnv/seg	32,7985	Media MensEnv/seg	44,6022
Media MensRec/seg	62,4218	Media MensRec/seg	108,3031	Media MensRec/seg	85,3625
Diferencia media total (ms)	17,4133	Diferencia media total (ms)	31,1188	Diferencia media total (ms)	24,2661
Exp6.3(p1)		Exp6.3(p2)		Exp6.3(media)	
Media MensEnv/seg	52,0170	Media MensEnv/seg	58,5046	Media MensEnv/seg	55,2608
Media MensRec/seg	109,7044	Media MensRec/seg	99,3032	Media MensRec/seg	104,5038
Diferencia media total (ms)	19,3055	Diferencia media total (ms)	17,3130	Diferencia media total (ms)	18,3092
Exp6.4(p1)		Exp6.4(p2)		Exp6.4(media)	
Media MensEnv/seg	48,7137	Media MensEnv/seg	59,1329	Media MensEnv/seg	53,9233
Media MensRec/seg	111,0951	Media MensRec/seg	92,5227	Media MensRec/seg	101,8089
Diferencia media total (ms)	21,1655	Diferencia media total (ms)	16,9424	Diferencia media total (ms)	19,0539
Exp6.5(p1)		Exp6.5(p2)		Exp6.5(media)	

Figura E.6: Medidas obtenidas en experimento 6.

Media MensEnv/seg	52,6362	Media MensEnv/seg	53,2934	Media MensEnv/seg	52,9648
Media MensRec/seg	105,4029	Media MensRec/seg	105,1486	Media MensRec/seg	105,2757
Diferencia media total (ms)	19,1043	Diferencia media total (ms)	18,8959	Diferencia media total (ms)	19,0001
Exp7(p1)		Exp7(p2)		Exp7(media)	

Figura E.7: Medidas obtenidas en experimento 7.

Al igual que en el modo normal el caso más interesante dentro del modo múltiple es el sexto experimento, ya que dentro de los componentes del experimento hay dos bots sombra de tipo DELAYEDSHADOW, por lo que aparecen las mayores latencias. La máxima diferencia entre mensajes enviados es de unos 30 milisegundos, para obtener el tiempo de la vuelta he realizado una serie cálculos cuyo resultado puede verse en la figura E.8, en ella se muestra una tabla con la media de mensajes recibidos entre el tiempo real por experimento. Se obtiene una media de 105,4876 mensajes recibidos por segundo, por lo que se están recibiendo mensajes del servidor cada 9,4798 milisegundos, hay que tener en cuenta que se incluyen los mensajes del oponente persona y del bot sombra. Como la cantidad de mensajes enviados por el bot sombra siempre va a ser inferior al del jugador persona, del flujo de mensajes que llegan al jugador destino representarán algo menos de la mitad, cogiendo la mitad de la media anterior obtendríamos un tiempo entre mensajes recibidos provenientes del bot sombra de 18,96 milisegundos, redondeando unos 20 milisegundos. Sumando tiempos, los mensajes que le llegan a un jugador desde otro llevarán como máximo una desincronización de 50 milisegundos.

	Media MensRec/TReal
EXP6.1 – waitTime = 100ms	108,7298
EXP6.2 – waitTime = 200ms	114,7001
EXP6.3 – waitTime = 300ms	87,9966
EXP6.4 – waitTime = 400ms	108,8528
EXP6.5 – waitTime = 500ms	107,1586
Media Total	105,4876

msegMensRec	9,4798

Figura E.8: Tabla de mensajes recibidos en tiempo real del experimento 6.

E.5.3. Tiempos aplicación vs Experimento Iizuka y Di Paolo (2007)

Como se ha descrito en las secciones anteriores el tiempo máximo que tarda en llegar un mensaje de un cliente a otro es de 50 milisegundos, es decir, un jugador detecta el movimiento del receptor de otro jugador con una desicronización máxima de 50 milisegundos. Si observamos la gráfica de la figura E.9, que hace referencia al experimento realizado por Iizuka y Di Paolo (2007)[25], vemos la diferencia entre las dos trayectorias de los agentes para el caso de interacciones en vivo (línea continua) y para el caso de interacciones con una grabación (línea discontinua). En principio ambas trazas no difieren mucho, la diferencia entre trayectorias no supera las 100 unidades de longitud, pero a partir de 400 milisegundos de desfase la diferencia para el caso unilateral aumenta considerablemente perdiendo la coordinación. En el caso de la aplicación no añade un desfase superior a 50 milisegundos por lo que se podrá llegar a la coordinación de interacciones sin problema.

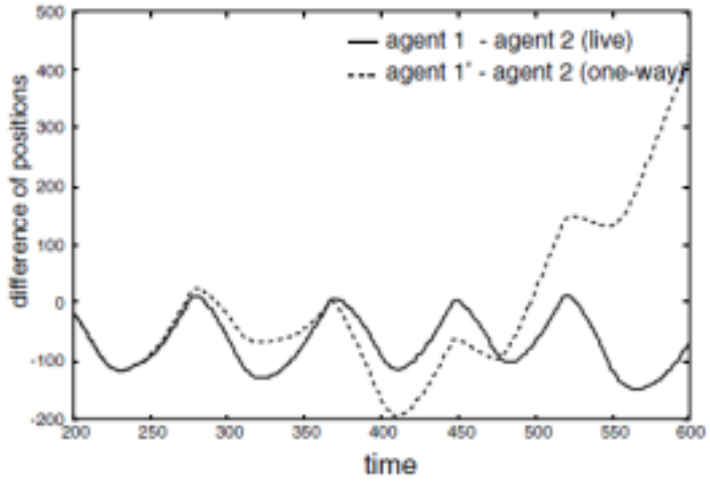

Figura E.9: Diferencia entre las dos trayectorias en cada condición. Cuando la línea cruza 0 significa que dos agentes se cruzan entre sí.

Anexo F

Parámetros comparables

En este apéndice quería añadir información que permitiera comparar parámetros del experimento de Auvray et al. (2009)[5][1] y parámetros de la aplicación desarrollada en este proyecto, a la vez que resumiera datos, que pueden resultar de interés, para el estudio de ambos casos . En las tablas F.1[2] y F.2 se muestra la información comparable de cada parte:

Parámetros	
Longitud espacio unidimensional	800 píxeles.
Tamaño información compartida	50 bytes(fuente, posición, número de mensaje) de forma asíncrona.
Longitud campo receptor	50 píxeles.
Entrada sensorial	Visual (color cuadrado, marcador colisiones) y auditiva (efecto Doppler), una mano mueve el ratón.
Oponentes por experimento	Modo Normal: 1 vs 1 (entre personas o persona y bot). Modo Múltiple: oponente persona, su sombra y valor fijo.
Distancia entre el señuelo móvil	La que se desee (configurable, por defecto 100 píxeles), también posible sombra con retardo en milisegundos(también configurable, por defecto 400 milisegundos).
tasa testeo estimulos-click	Sistema asíncrono, en el momento de hacer el click registra estímulos (colisiones). Una colisión tiene una vida máxima entre clicks de 200 milisegundos, por defecto, valor ajustable.
Personas por prueba	Modo Normal: 1 o 2 personas. Modo Múltiple: 2 personas.
Tipos de objeto no persona	Modo Normal: RANDOMSIMPLE, SHADOW, DELAYEDSHADOW y COMPLEX. Modo Múltiple: DELAYEDSAHDOW, PIXELSHADOW y valor fijo.
Formato de datos almacenados	JSON

Tabla F.1: Información de la aplicación desarrollada en el proyecto.

[1]Para tener una descripción más detallada del experimento ir a la sección 2.2 del capítulo 2 de la memoria

[2]Descripción de los tipos de bot en la subsección B.1.5 del apéndice B. Descripción de datos almacenados en la sección D.5 del apéndice D.

Parámetros	
Longitud espacio unidimensional	600 píxeles.
Tamaño información compartida	1 bit (todo o nada) de forma síncrona.
Longitud campo receptor	4 píxeles.
Entrada sensorial	Táctil (estimulador táctil), con una mano se manipula el ratón y con la otra se percibe los estímulos, ojos vendados.
Oponentes por prueba	cuerpo-objeto persona, señuelo móvil y señuelo fijo.
Distancia entre el señuelo móvil	50 píxeles.
tasa testeo estimulos-click	cada 2 segundos.
Personas por prueba	2 personas.
Tipos de objeto no persona	Sombra con distancia del oponente persona y fijo (distinto para cada persona).

Tabla F.2: Información del experimento Auvray et al. (2009)[5].

Bibliografía

[1] Abu-akel A. Impaired theory of mind in schizophrenia. Pragmatics and cognition. 1999;7:247—82. 2.1.1

[2] Astington, J. W. (1993). The Child´s Discovery of the Mind. Cambridge, MA: Harvard University Press. 1.2, 2.1

[3] Astington, J. W., Harris, P. L. & Olson, D. (Eds.) (1988).Developing theories of mind. Cambridge, UK: Cambridge University Press 1.2, 2.1

[4] Auvray M., Rhode M. (2012). Perceptual crossing: the simplest online paradigm. Front. Hum. Neurosci. 6:181. doi: 10.3389/fnhum.2012.00181. (document), 1.1, 2.2

[5] Auvray M., Lenay C., Stewart J. (2009). Perceptual interactions in a minimalist virtual environment. New Ideas Psychol. 27, 32–47. doi: 10.1016/j.newideapsych.2007.12.002. (document), 1.1, 1.2, 2, 2.2, 2.3, 2.3, 2.4, 2.4, 2, 3, 6.2, B.1.3, F, F.2

[6] Bartsch, K. y Wellman, H. M. (1995). Children Talk About the Mind. New York: Oxford University Press. 2.1

[7] Bishop, D. (1997). Uncommon understanding. Development and disorders of language comprehension in children. Hove: Psychology Press. 1.2, 2.1

[8] Brothers L. The social brain: A Project for integrating primate behavior and neurophysiology in new domain. Concepts in Neuroscience. 1990;1:27—61. 2.1.1

[9] Brown, J., Aczél, B., Jiménez, L., Kaufman, S. B., and Plaisted-Grant, K. (2010). Intact implicit learning in autism spectrum conditions. Q. J. Exp. Psychol. 63, 1789–1812. 2.4

[10] De Jaegher, H. (2009). Social understanding through direct perception? Yes, by interacting. Conscious. Cogn. 18, 535–542. 2.4

[11] Di Paolo, E., Rohde, M., and Iizuka, H. (2008). Sensitivity to social contingency or stability of interaction? Modelling the dynamics of perceptual crossing. New Ideas Psychol. 26, 278–294. 1.1, 2.4

[12] Flavell, J.H. (2004). Development of knowledge about vision. In D.T. Levin (Ed.), Thinking and seeing: Visual metacognition in adults and children. Cambridge, MA: M.I.T. Press. 2.1

[13] Flavell, J. H. (2000). Development of children's knowledge about the mental world.International Journal of Behavioral Development, 24, 15–23. 2.1

[14] Flavell, J. H., & Miller, P. H. (1998). Social cognition. In W. Damon (Series Ed.), D. Kuhn & R. S. Siegler (Eds.), Handbook of child psychology: Vol. 2. Cognition, perception, and language (5th ed., pp. 851–898). New York: Wiley 1.2, 2.1

[15] Fodor J. The modularity of mind. Cambrigde, MA: MIT Press; 1983. 2.1.1

[16] Frith CD. The Cognitive Neuropsychology of Schizophrenia. Hove: Lawrence Erlbaum Associates; 1992. 2.1.1

[17] Froese T., Di Paolo E. (2011a). The enactive approach: theoretical sketches from cell to society. Pragmat. Cogn. 19, 1–36. doi: 10.1075/pc.19.1.01fro. [Cross Ref] 2.2

[18] Froese T., Di Paolo E. A. (2010). Modelling social interaction as perceptual crossing: an investigation into the dynamics of the interaction process. Conn. Sci. 22, 43–68. doi: 10.1080/09540090903197928. 2.2

[19] Gallese P, Goldman A. Mirror neurons and the simulation theory of mind-reading. Trends in cognitive Sciencie. 1998;2:493—501. 2.1.1

[20] Garfield, J. L. ; Peterson, C. C. & Perry, T. (2001). Social cognition, language acquisition and the development of the theory of mind. Mind and Language 16 (5):494–541. 2.1

[21] Hardy-Baylé MC. Organisation de l'action, phénomenes de conscience et representation mentale de láction chez des schizophrenes. Actualités psychiatriques. 1994;20:393—400. 20. 2.1.1

[22] Iizuka, H., Ando, H., and Maeda, T. (2012a). "Emergence of communication and turn-taking behavior in nonverbal interaction (in Japanese)," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences J95-A, 165–174. 2.4, 1, 6.2, B.1.3

[23] Iizuka, H., Marocco, D., Ando, H., and Maeda, T. (2012b). Experimental study on coevolution of categorical perception and communication systems in humans. Psychol. Res. doi: 10.1007/s00426-012-0420-5. [Epub ahead of print]. 2.4

[24] Iizuka, H., Ando, H., and Maeda, T. (2009). "The anticipation of human behavior using 'Parasitic Humanoid'," in Human-Computer Interaction. Ambient, Ubiquitous and Intelligent Interaction„ ed J. Jacko (Berlin, Heidelberg: Springer-Verlag), 284–293. 2.4, 1, 6.2, B.1.3

[25] Iizuka, H. and Di Paolo, E. A. (2007). Minimal agency detection of embodied agents. Proceedings of the 9th European Conference on Artificial life ECAL 2007. Springer-Verlag. (document), 1.1, 2, 2.3, 2.5, 2.7, 2.8, 2.4, 1, 2, 5, 5.5, 6.2, B.1.1, B.1.3, E.5.3

[26] Lenay C., Stewart J. (2012). Minimalist approach to perceptual interactions. Front. Hum. Neurosci. 6:98. doi: 10.3389/fnhum.2012.00098. (document), 1.1, 2.1, 2.2, 2.3, 2.1, 2.4, 2.6, 2.4

[27] Lenay, C., Stewart, J., Rohde, M., and Ali Amar, A. (2011). You never fail to surprise me: the hallmark of the other. Experimental study and simulations of perceptual crossing. Interact. Stud. 12, 373–396. 2.4

[28] Marti, P. (2010). Perceiving while being perceived. Int. J. Des. 4, 27–38. 2.4

[29] McIntosh, D. N., Reichmann-Decker, A., Winkielman, P., and Wilbarger, J. L. (2006). When the social mirror breaks: deficits in automatic, but not voluntary, mimicry of emotional facial expressions in autism. Dev. Sci. 9, 295–302. 2.4

[30] Michael J., Overgaard S. (2012). Interaction and social cognition: a comment on Auvray et al.'s perceptual crossing paradigm. New Ideas Psychol. 30, 296–299. doi: 10.1016/j.newideapsych.2012.02.001. 2.2

[31] Mitchell, P. (1997). Introduction to theory of mind: Children, autism and apes.London: Arnold 1.2, 2.1

[32] Murray, L., & Trevarthen, C. (1985). Emotional regulations of interactions between two-month-olds and their mothers. In T. M. Field & N. A. Fox (Eds.), Social perception in infants (pp. 177-197). Norwood, NJ: Ablex. 2.3, 2.3, 2.4

[33] Perner J. Understanding the representational Mind. Cambridge; 1991. 2.1.1

[34] Premack, D., & Woodruff, G. (1978). Does the chimpanzee have a theory of mind? Behavioral and Brain Sciences, 1, 515–526 1, 1, 2.1.1

[35] Repacholi, B. M., & Slaughter, V. (Eds.). (2003). Individual differences in the theory of mind. New York: Psychology Press 2.1

[36] Rohde, M. (2010). Enaction, Embodiment, Evolutionary Robotics. Simulation Models in the Study of Human Cognition, Series: Thinking Machines, Vol. 1, Amsterdam, Beijing, Paris: Atlantis Press. 2.4

[37] Rohde, M., and Di Paolo, E. (2008). "Embodiment and perceptual crossing in 2D: a comparative evolutionary robotics study," in Proceedings of the 10th International Conference on the Simulation of Adaptive Behavior SAB 2008, (Berlin, Heidelberg: Springer-Verlag), 83–92. 2.4

[38] Sebanz, N., Knoblich, G., Stumpf, L., and Prinz, W. (2005). Far from action-blind: representation of others' actions in individuals with autism. Cogn. Neuropsychol. 22, 433–454. 2.4

[39] Sperber, D. & Wilson, D. (2002) Pragmatics, modularity and mind-reading. Mind & Language 17. 3- 23. 1.2, 2.1

[40] Sribunruangrit N., Marque C. K., Lenay C., Hanneton S., Gapenne O., Vanhoutte C. (2004). Speed-accuracy tradeoff during performance of a tracking task without visual feedback. IEEE Trans. Neural Syst. Rehabil. Eng. 12, 131–139. doi: 10.1109/TNSRE.2004.824222. 2.2

[41] Timmermans, B., Schilbach, L., and Vogeley, K. (2011). Can You Feel Me: Awareness to Dyadic Interaction in High Functioning Autism, ESCOP. Spain: San Sebastian, 29 Sep - 2 Oct. 2.4

[42] Ware, E. (2011). Interactive Behaviour in Pigeons: Visual Display Interactions as a Model for Visual Communication, PhD thesis, Centre for Neuroscience Studies, Queen's University, Kingston, ON. 2.4

[43] Wellman, H. M., Cross, D., & Watson, J. (2001). Meta-analysis of theory-of-mind development: The truth about false belief. Child Development, 72, 655–684. 2.1

[44] Wilkerson W. S. (1999). From bodily motions to bodily intentions: the perception of bodily activity. Philos. Psychol. 12, 61–77. doi: 10.1080/095150899105936. 2.2

[45] Wimmer H, Perner J. Beliefs about beliefs: representation and constraining function of wrong beliefs in young children's understanding of deception. Cognition. 1983 Jan;13(1):103-28. PubMed PMID: 6681741. 2.1

Libros técnicos consultados

- Alvárez García, Alonso. *HTML 5*. Madrid, Editorial Anaya Multimedia, D.L. 2010, 303 págs.
- Lowery, Joseph W. & Fletcher M. *HTML 5 para desarrolladores*. Madrid, Editorial Anaya Multimedia, D.L. 2011, 365 págs.
- Flanagan, David. *JavaScript : la guía definitiva*. Madrid, Editorial Anaya Multimedia, D. L. 2007, traducción de la 5ª ed. en inglés, 1168 págs.
- Perry, Bruce W. *Java Servlet and JSP cookbook*. Publicación Beijing, Editorial O'Reilly, 2004, 723 págs.

Enlaces web más consultados

- http://tools.ietf.org/html/rfc6455
- http://www.w3.org/TR/2011/WD-websockets-20110419/
- http://stackoverflow.com/
- http://www.w3schools.com/
- http://jquery.com/