UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Desarrollo de un sistema gastrolúdico multiusuario basado en realidad aumentada y estrategias de gamificación

Trabajo Fin de Máster

Máster Universitario en Ingeniería y Tecnología de Sistemas Software

Departamento de Sistemas Informáticos y Computación

Autor: Sergio Belda Galbis

Tutor: Javier Jaén Martínez

Valencia, septiembre de 2019

Curso 2018/2019

Agraïments

Done les gràcies als meus companys de màster i amics que han participat a l'avaluació experimental.

Moltes gràcies als meus pares i a la meua germana per la seua estima i suport incondicional.

Agraïsc, en especial, al tutor d'aquest treball final de màster, Javier Jaén Martínez, l'esforç i temps dedicat, així com els coneixements i idees transmeses.

Resumen

La acción de comer o beber invita al juego. Es posible jugar con los sabores, las texturas o el aspecto y tener una actividad lúdica tanto en la preparación de la comida como en su presentación y consumo. La comida, además de su función fundamental para alimentarnos, tiene la capacidad de proporcionar una interacción entre el consumidor y las propiedades que la caracterizan: forma, disposición, gusto, olor, de una manera similar a las acciones e interacciones que se dan entre un juguete y un jugador. Tanto es así, que recientemente se ha acuñado el término Gastroludología [1] para describir una nueva línea de investigación que pertenece a las áreas de HCI y gamificación que pretende estudiar las nuevas oportunidades de experiencias lúdicas y nuevas técnicas de interacción que surgen al integrar la comida en ambientes gamificados. De hecho, esta disciplina no es nueva dado que ya en el 2014 se publicó un *Special Issue* en la revista "International Journal of Human Computer Studies" titulado "Food & Interaction Design" [2] en el que se aborda la importancia de esta nueva disciplina desde el punto de vista de la interacción hombre máquina. Esta nueva disciplina puede servir para construir entornos educativos que sensibilicen sobre el problema asociado al desperdicio de comida, sobre la práctica de hábitos de comida saludables, o hasta incluso construir experiencias lúdicas que tengan lugar en restaurantes mientras se consume bebida/comida. El presente trabajo de máster parte de esta premisa y pretende estudiar el uso de las tecnologías de realidad aumentada en el contexto de las experiencias gastrolúdicas gamificadas. En el trabajo se construirá un sistema de realidad aumentada gamificado en el que poder explorar qué técnicas de gamificación en grupo pueden ser efectivas para la construcción futura de experiencias gastrolúdicas con múltiples usuarios. Se trata, pues, de un primer paso exploratorio en un área de investigación incipiente que permitirá sentar las bases de un futuro desarrollo más ambicioso de esta nueva línea de investigación.

Palabras clave: gastroludología, realidad aumentada, visión por computador, gamificación.

Resum

L'acció de menjar o beure invita al joc. És possible jugar amb els sabors, les textures o l'aspecte i tenir una activitat lúdica tant en la preparació del menjar com amb la presentació i consum. El menjar, a més de la seua funcionalitat fonamental per a alimentar-nos, té la capacitat de proporcionar una interacció entre el consumidor i les propietats que el caracteritzen: forma, disposició, gust, olor, d'una forma semblant a les accions e interaccions que es donen entre un joguet i un jugador. Tant es així, que recentment s'ha designat el terme Gastroludologia [1] per descriure una nova línia d'investigació que pertany a les àrees d'HCI i gamificació que pretén estudiar les noves oportunitats d'experiències lúdiques i noves tècniques d'interacció que surten a l'integrar el menjar en àmbits gamificats. De fet, aquesta disciplina no és nova donat que ja en el 2014 es va publicar un *Special Issue* a la revista "International Journal of Human Computer Studies" amb el títol "Food & Interaction Design" [2] on s'aborda la importància d'aquesta nova disciplina des del punt de vista de la interacció home màquina. Aquesta nova disciplina pot servir per construir entorns educatius que sensibilitzen sobre el problema associat al malbaratament de menjar, sobre la pràctica d'hàbits de menjar saludable, o fins i tot, construir experiències lúdiques que tinguin lloc a restaurants mentre es consumeix menjar o beguda. El present treball de màster parteix d'aquesta premissa i pretén estudiar l'ús de les tecnologies de realitat augmentada al context de les experiències gastrolúdiques gamificades. Al treball es construirà un sistema de realitat augmentada gamificat en el que poder explorar què tècniques de gamificació en grup poden ser efectives per a la construcció futura d'experiències gastrolúdiques amb múltiples usuaris. Es tracta, doncs, d'un primer pas exploratori en una àrea d'investigació incipient que permetrà assentar les bases d'un futur desenvolupament més ambiciós d'aquesta nova línia d'investigació.

Paraules clau: gastroludologia, realitat augmentada, visió per computadora, gamificació.

Abstract

Eating or drinking invites to play. It is possible to play with the flavors, textures or appearance and have a playful activity both in the preparation of food and with the presentation and consumption. The food, in addition to its fundamental functionality to feed us, has the ability to provide an interaction between the consumer and the properties that characterize him: shape, disposition, taste, smell, in a similar way to the actions and interactions that occur between a player and a toy. Recently, the term Gastroludology [1] has been designated to describe a new line of research that belongs to the areas of HCI and gamification that aims to study the new opportunities and interaction techniques that emerge from integrate the food in a gamified context. In fact, this idea is not new since in 2014 a Special Issue was published in the journal "International Journal of Human Computer Studies" called "Food & Interaction Design" [2] where the importance of this new discipline is addressed from the point of view of human computer interaction. This new discipline can be used to create educational environments that raise awareness about the problem associated with food waste, about eating habits, or even create playful experiences that take place in restaurants while consuming food or drink. This master's degree final project is based on this premise and aims to study the use of augmented reality technologies in the context of a gamified gastroludical experience. In this project, we will create a gamified augmented reality system in which to explore what gamification techniques in group can be effective for the future construction of gastroludical experiences with multiple users. It is therefore a first exploratory step in an incipient research area that will allow to lay the foundation for a more ambitious future development of this new line of research.

Keywords: gastroludology, augmented reality, computer vision, gamification.

Tabla de contenido

Índice de figuras

Índice de gráficos

1. Introducción

La gastroludología define una experiencia que incorpora un elemento de juego, un juego o un elemento de gamificación en cualquier forma de interacción gastronómica. Es el tema entorno al cual gira este proyecto junto a tecnologías interesantes como la realidad aumentada o la visión por computador.

En este capítulo se realiza una introducción al proyecto, mediante una descripción de la motivación y objetivos del proyecto, así como antecedentes existentes al desarrollo de este sistema e impacto esperado una vez se haya finalizado. Se expone, asimismo, cual será la estructura seguida en este documento para describir el proceso.

El siguiente documento describe el desarrollo de un proyecto gastrolúdico multiusuario basado en realidad aumentada y estrategias de gamificación.

1.1. Motivación

Este proyecto surge de la voluntad de desarrollar un sistema dentro del área de HCI[1] y explorar la nueva línea de investigación de la gastroludología. Esta línea se encuentra todavía en un estado prematuro ya que las publicaciones describen el concepto a nivel teórico a la espera del desarrollo de sistemas reales que apoyen y puedan otorgar relevancia a este concepto en el ámbito de HCI.

La motivación, por tanto, de este proyecto, es la de llevar este concepto teórico a un sistema real para poder posteriormente observar de forma empírica cómo se adapta a un contexto práctico.

Resulta interesante indagar en un concepto que tiene poca exploración tanto teórica como práctica ya que por un lado se puede percibir como un reto tanto de aprendizaje de nuevas tecnologías relacionadas con esta área de investigación, como de enfrentarse a nuevas ideas fuera de lo establecido; así como una forma de poder estar a la vanguardia de una técnica que podría tener un futuro en la investigación.

En este proyecto se utiliza como tecnología de interacción con el usuario realidad aumentada y visión por computador. Es atractivo aprender una tecnología que en los últimos años ha tenido mayor auge en el área de los dispositivos móviles, ya que será más utilizada en el tiempo y, por lo tanto, es interesante entrar en contacto con las bibliotecas que se han presentado y que irán evolucionando.

[1] *Human Computer Interaction*

1.2. Antecedentes

Como se ha comentado, existen pocos ejemplos de antecedentes en esta área de estudio, pues esta línea surge hace pocos años con la aparición de ciertos componentes y tecnologías que permitían este tipo de interacción. En el siguiente capítulo se llevará a cabo una descripción más exhaustiva de la línea de investigación y de algún caso de estudio representativo.

Si nos centramos en las tecnologías utilizadas en este proyecto, se ha utilizado realidad aumentada para dispositivos móviles, la cual no posee muchos años de vida y está siendo cada vez empleada en un mayor número de sistemas, pero prácticamente no se ha utilizado en esta área de conocimiento.

1.3. Objetivos del proyecto

El objetivo principal de este proyecto es el de desarrollar un sistema multijugador gastrolúdico. Para este fin, se emplea tecnología de realidad aumentada y visión por computador.

También podemos recalcar el propósito del desarrollo de un sistema en un área incipiente y que, por lo tanto, está a la espera de ser explorada. Es decir, realizar un proyecto en una línea que todavía está en un estado embrionario y constituir uno de los primeros acercamientos a proyectos en esta área.

Se destaca también como meta el hecho de utilizar tecnología que no ha sido empleada en esta área y que tiene una fuerte consideración en el presente. Por lo tanto, el objetivo es observar de qué forma se adapta dicha tecnología a este concepto para analizar su viabilidad en futuros proyectos.

1.4. Impacto esperado

Con el desarrollo de este proyecto se espera no solo dar a conocer esta disciplina sino apoyar e impulsar el desarrollo de otros sistemas en este ámbito, así como un aumento en la investigación científica.

Se trata, pues, de dar a conocer una línea de investigación desconocida para algunas personas, que pueden estar interesadas en indagar e investigar más sobre este concepto.

1.5. Estructura del proyecto

Este documento tiene como objetivo mostrar el desarrollo de un sistema de carácter gastrolúdico que utiliza tecnologías de realidad aumentada y visión por computador como interacción.

Se inicia con un análisis del estado del arte o situación actual del área de investigación en la que se encuadra este proyecto, al igual que de las tecnologías que son aplicadas en el desarrollo. A continuación, se realiza un análisis de los requisitos del sistema y se lleva a cabo un modelado conceptual del mismo. En el capítulo de diseño se discute acerca del diseño arquitectónico, así como del prototipado de interfaces de usuario o diseño UI[1]. Seguidamente, se describe la implementación desde el punto de vista de los elementos desarrollados y su implicación en el sistema. Posteriormente, se realiza una evaluación del sistema mediante un proceso experimental y un posterior análisis de los resultados. Se cierra con unas conclusiones sobre el sistema desarrollado y una reflexión sobre el porvenir de esta área, así como el trabajo futuro que debe ser llevado a término.

[1] *User Interface*

2. Estado del arte

En este capítulo se describe el estado del arte o análisis de la situación actual de la tecnología y conceptos principales de este proyecto. Se introduce con una descripción sobre el concepto de gastroludología y su estado actual como línea de investigación. Se trata la situación de la tecnología de realidad aumentada en la actualidad y las herramientas presentes en el desarrollo sobre dispositivos móviles, y se detalla la técnica de visión por computador y plataformas existentes.

2.1. Gastroludología

La gastroludología o la interacción lúdica con la comida es un concepto que se ha tratado en diferentes artículos científicos.

Ya en el trabajo de Comber et al. [2], en 2014, se habla del concepto de *Human Food Interaction* (HFI) como un área de investigación de la HCI, donde la comida juega un papel fundamental para la vida de las personas: en su bienestar, vida social, en la sostenibilidad ambiental, y salud. La comida sirve como punto de unión para acercar a las personas y ofrecer nuevas experiencias apasionantes. La salud y el bienestar han despertado el interés de la HCI para llevar a cabo interfaces que apoyen la nutrición, y el estilo de vida saludable.

En el trabajo de Chisik et al. [1], la gastroludología unifica la gastronomía, donde el objetivo es el placer de comer; y la ludología, donde se estudia cualquier forma de juego. La gastroludología, incorpora un elemento de juego, un juego o un elemento de gamificación en cualquier forma de interacción gastronómica, las cuales pueden ser la preparación, el acto de comer o el análisis de la comida, la cultura y lugares relacionados con ésta.

Se define una experiencia gastrolúdica como la experiencia que involucra las sensaciones que proporciona comer y beber con la interacción de una actividad o juego.

Para calificar un juego de gastrolúdico, la experiencia debe satisfacer el requisito de ser un juego y una experiencia gastronómica. Por ejemplo, no se considera un juego gastrolúdico el Pac-Man o el Candy Crush Saga pues no existe una interacción física o gustativa con la comida, ni tampoco lo sería cualquier acción cuotidiana en la que intervenga comida como puede ser pelar una pieza de fruta, ya que no se encuentra bajo ninguna regla de juego, más allá de que únicamente se puede establecer una competición.

Un objeto de juego puede ser virtual o físico, o una combinación de ambos. La experiencia de juego puede ser virtual, física o una mezcla de las dos. En una experiencia de juego física, normalmente el jugador se encuentra con elementos de juego que no varían en el tiempo, no obstante, en una experiencia gastrolúdica física los elementos pueden ser modificados, bien al ser ingeridos o alterados de otra forma.

En cambio, una experiencia virtual es aquella en que el jugador no tiene una interacción física con ningún elemento gastronómico. Las sensaciones gastronómicas experimentadas son puramente representaciones virtuales. Los jugadores pueden estar separados o en el mismo espacio físico y la interacción puede ser síncrona o asíncrona; e interactúan mediante interfaces como pantallas táctiles, teclados, altavoces o incluso dispositivos como una Digital Lollipop [3], que simula sabores mediante estimulación eléctrica en la lengua.

Las experiencias donde se mezclan elementos físicos y virtuales se clasifican en diferentes categorías. Experiencias en las que se añade un elemento comestible a una interfaz física de computador, experiencias en las que se añade componentes electrónicos a utensilios para comer o beber, como vasos o platos; utilizar sensores para detectar la actividad de comer o utilizar la comida como un elemento de interfaz.

La jugabilidad de la experiencia se puede analizar desde diferentes perspectivas. Como elemento de diversión, se incorpora un elemento de juego a una experiencia gastrolúdica, por ejemplo, jugar con sabores, texturas, formas, al igual que en la preparación se está en constante búsqueda de combinaciones, técnicas y tecnologías; jugar con el entorno, como restaurantes donde se utilizan proyecciones en la pared y mesas o sistemas de sonido; o jugar con la forma en que la comida es servida.

Hasta la fecha, hay muy pocos juegos que se ajusten a esta definición y, por lo tanto, esta área está madura para la exploración y diseño. Como un ejemplo se puede destacar *"You Better Eat to Survive!"* [4], un caso de estudio que entraría dentro de la categoría de experiencia virtual y física, pues utiliza la realidad virtual como método de interacción, así como representaciones físicas de comida.

Figura 1 Jugando You Better Eat to Survive!

Se plantean diferentes cuestiones relacionadas con este campo de estudio que pueden ser estudiadas en futuras investigaciones. En primer lugar, el contexto, dependiendo del ámbito en el que se encuentra una persona se podrá desarrollar un tipo de experiencia u otra. En segundo lugar, mantener el sabor de los elementos de juego. Por otra parte, el control en términos de aspectos nutricionales y salud. Explorar propiedades materiales de diferentes productos alimenticios junto novedosas técnicas de fabricación de la gastronomía digital como la impresión 3D o el uso de láser. Que los objetos que se utilicen puedan constituir diferentes mecanismos y estéticas para el diseño del juego. Y, por último, explorar las oportunidades de la narrativa en estas experiencias gastrolúdicas y como pueden incorporase a la experiencia general.

En el artículo de Khot et al. [5] trata como el HFI se basa en tendencias recientes de HCI. Clasifica diferentes trabajos en fases: cocina, comida, etc., para revelar áreas de exploración que pueden ser investigadas. Se nombran oportunidades que ponen de manifiesto características que la tecnología debe cumplir para poder impulsar este campo. Se trata de una revisión sistemática que muestra hallazgos de una variedad de proyectos y estudios.

El campo de HFI ha sido tratado en diferentes *workshops* y simposios, y existen diferentes grupos de investigación trabajando en esta área: *FoodCHI Special Interest Group; SIGCHI foodCHI network; Facebook FoodCHI group and; an ACM Future of Computing Academy working group on Computing and Food*. Se añade que los *early adopters* de HFI están experimentando con nuevas tecnologías emergentes como *food printing*, realidad virtual, robótica, levitación acústica, etc.

También se expone que, aunque el HFI normalmente se enmarca como una subdisciplina de HCI, se destaca la naturaleza interdisciplinar de esta área que se ha relacionado en campos de conocimiento como la antropología, ciencias médicas, etc.

2.2. Realidad aumentada en dispositivos móviles

La realidad aumentada se define como la inclusión en el mundo real de información digital que es percibida por el usuario como si formase parte de este. Esta información no se limita a ser percibida únicamente por un medio visual, sino que puede ser implementada de forma que se advierta vía sonidos o por el olfato o gusto.

Un usuario se mueve por una escena de realidad aumentada y controla la experiencia. El sistema recoge la entrada del usuario obteniendo información y su punto de vista.

Un sistema de realidad aumentada necesita tres componentes: un componente de seguimiento, un componente de registro y un componente de visualización. Se añade,

además, un cuarto componente que es el modelo espacial para almacenar información sobre el mundo real [6].

Aunque se trate de un concepto que en la actualidad se presenta como reciente y una tecnología que está creciendo ahora mismo, el origen de la realidad aumentada se remonta a 1960 con el primer visor montado en la cabeza y no seria hasta 1992 que se acuñaría el término de realidad aumentada [7]. La definición más aceptada de este concepto es la de Azuma [8] en 1997, y cuyas características son que debe combinar el mundo real y virtual, a diferencia de la realidad virtual; debe ser interactivo en tiempo real y la información se debe mostrar al usuario mediante modelos tridimensionales.

En la actualidad, la realidad aumentada se ha aplicado a múltiples ámbitos de uso, tanto profesionales como educativos. Existen múltiples publicaciones que describen la aplicación de esta tecnología en diferentes contextos, como el educativo, en sistemas de aprendizaje de idiomas [9]; o el industrial, en la asistencia y seguimiento de tareas [10]; así como el ámbito de la medicina, en el entrenamiento y estudio de diferentes operaciones [11].

Una de las áreas de la realidad aumentada es la *Mobile Augmented Reality* (MAR) [12], la cual se define como la adaptación de la realidad aumentada a dispositivos móviles que los usuarios normalmente llevan consigo, y por tanto, no es necesario depender de un espacio concreto para disfrutar de una experiencia de realidad aumentada, ya que puede ser adecuada a la zona en la que esté. Se trata de aprovechar la enorme potencia de procesamiento, sensores y cámaras que poseen algunos dispositivos como *smartphones* o *smart glasses*.

Esta tecnología se ha adaptado a diferentes contextos. Uno de ellos, es el de la industria del videojuego, con el desarrollo del famoso Pokémon Go[1] llevado a cabo por Niantic en 2016. También, existen ejemplos de aplicaciones de comercio, como el caso de Ikea Place[2], que permite colocar muebles y accesorios virtuales en cualquier habitación para observar como quedarían antes de adquirirlos; así como sistemas para automóviles [13], con el objetivo de incrementar la seguridad del usuario con el reconocimiento de señales de tráfico incluso con condiciones climáticas desfavorables.

En este contexto de realidad aumentada móvil, aparecen múltiples tecnologías para el desarrollo de aplicaciones de realidad aumentada para estos sistemas. Algunas tecnologías destacables son Vuforia, ARCore o ARKit.

[1] https://pokemongolive.com/es/

[2] http://highlights.ikea.com/2017/ikea-place/

En 2017, Apple anunció ARKit[1], una biblioteca para desarrollar aplicaciones móviles en dispositivos con sistema operativo iOS.

Figura 2 Apple ARKit 3

Podemos destacar algunas características de la última versión de esta biblioteca, como *people occlusion*, que se basa en la detección de cuerpos que se encuentran delante del objeto virtual y que por tanto deben taparlo parcialmente; el seguimiento de movimiento corporal, que permite obtener el movimiento de una persona en la escena y registrar gestos y acciones que realiza con su cuerpo para desencadenar eventos; la experiencia multiusuario, que traslada la experiencia de realidad aumentada a un contexto con más de un usuario en el que cada uno observa la información desde su perspectiva, compartiendo el mismo sistema de referencia; así como la detección de imágenes sobre las que se puede superponer información o modificar como se observan.

En 2018, aparece Google ARCore[2], la plataforma de Google que permite crear experiencias de realidad aumentada en dispositivos móviles Android.

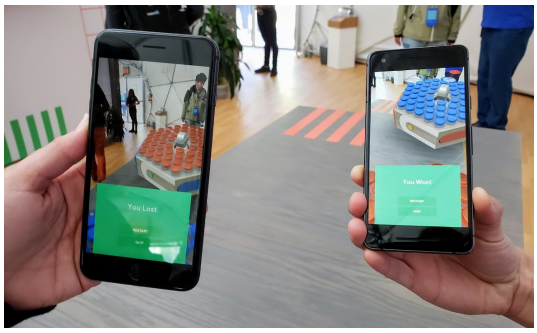


Figura 3 Google ARCore

Algunos de los conceptos fundamentales de esta tecnología son el seguimiento del movimiento en la escena, que se realiza mediante un proceso llamado *concurrent*

[1] https://developer.apple.com/documentation/arkit

[2] https://developers.google.com/ar/discover/concepts

odomentry[1] *and mapping*, para poder conocer donde se encuentra el dispositivo móvil respecto al mundo real, que utiliza puntos clave para monitorizar cualquier cambio de localización junto a los sensores de unidad de medición inercial (IMU); la detección de puntos característicos que definen planos sobre superficies horizontales y verticales, o incluso en ángulo, donde depositar objetos con los que interactuar. Asimismo, la estimación de luz, que es una característica que permite obtener los valores de iluminación de la escena real para aplicar corrección de color e intensidad sobre los objetos virtuales para proveer de una sensación de más realidad.

La interacción con la información digital se realiza de forma tradicional, es decir, con toques en la pantalla del dispositivo, y no existe ningún mecanismo actualmente en esta biblioteca para detectar gestos o movimientos de un cuerpo como modo de interacción de usuario.

Esta biblioteca también permite detectar imágenes y lanzar eventos asociados a estas ilustraciones. Esta característica permite mostrar objetos o información asociada a la imagen en la posición en la que se encuentra, respetando los límites de tamaño de esta.

Otra característica interesante son los llamados Anchor que se utilizan para poder mantener las características de posición y rotación de un objeto en una escena, aunque el usuario se mueva por ella y esté detectando nuevos planos y puntos clave. Esta característica es la base de Cloud Anchor, que permite enviar información sobre un Anchor y puntos clave cercanos de la escena a la nube para que otros usuarios puedan simultáneamente disfrutar de una experiencia de realidad aumentada compartida.

2.3. Visión por computador

La visión por computador es un campo que presenta diferentes objetivos de forma interdisciplinar. Desde el punto de vista de la biología, obtener modelos computacionales del sistema visual humano, y desde el punto de vista de la ingeniería, construir sistemas autónomos que puedan realizar algunas de las tareas que el sistema visual humano puede hacer, o mejorarlo incluso [14].

Actualmente, existen diferentes tecnologías relacionadas con la visión por computador. Podemos destacar OpenCV, y el *framework* TensorFlow.

[1] El proceso de estimar incrementalmente la posición de un robot o dispositivo.

OpenCV[1] es una biblioteca de visión por computador *open source* que posee más de 2500 algoritmos de visión por computador y *machine learning* para manipulación de imágenes, reconocimiento de caras, identificación de objetos, seguimiento de movimientos de cámara, encontrar imágenes entre similares en una base de datos, extraer modelos tridimensionales y reconocer marcadores sobre los que colocar elementos mediante realidad aumentada.

Puede ser utilizada para desarrollar proyectos para diferentes sistemas como Windows, Mac, Linux o Android, mediante C++, Python, Java o MATLAB. Se han desarrollado, además, diferentes *wrappers* para plataformas de desarrollo .NET, como, por ejemplo, Emgu CV o OpenCVSharp. Mediante esta biblioteca se pueden realizar un gran número de operaciones.

Se puede utilizar en seguimiento de objetos por color. Los objetos en una escena que posean un color uniforme pueden ser encontrados y detectar cualquier cambio en su posición o forma.

También se emplea en reconocimiento de caras para monitorizar gestos y movimientos de ojos, así como cambios de rostros, modificación del aspecto, etc.

Una característica interesante es la detección de esquinas de objetos en una imagen, lo cual puede utilizarse para extraer un modelo 3D. Para detectar los diferentes objetos que aparecen en una imagen se utiliza el reconocimiento de contornos, lo cual ayuda a separar los elementos que aparecen en una escena.

Incorpora la extracción de *keypoints* o puntos característicos de una imagen que pueden ser utilizados para conocer donde se encuentra un determinado punto en una imagen, aunque esta se transforme.

OpenCV integra también la detección de marcadores de la biblioteca de realidad aumentada ARUco, lo cual ayuda a poder depositar un objeto de realidad aumentada, conocer el ángulo de una determinada superficie o mantener en una posición fija una información digital si un usuario esta utilizando un dispositivo móvil como entrada de imagen y se está moviendo por la escena.

Una *feature* en visión por computador es un elemento que permite estudiar los componentes de una imagen, como el color, textura, superficies, formas y reflejos. Estas

[1] https://opencv.org/about/

características se pueden utilizar en reconocimiento de imagen, búsqueda, etc. Juegan un papel fundamental en crear sistemas de visión por computador de mayor calidad.

Las *features* se han vuelto más precisas. Esto se debe a un nuevo tipo de extractor de funciones llamado *Convolutional Neural Network* (CNN) y han demostrado una precisión notable en tareas complejas, como la detección de objetos y la clasificación de imágenes con alta precisión, y ahora son bastante ubicuas en aplicaciones que se extienden desde mejoras de fotos en teléfonos inteligentes hasta análisis de imágenes satelitales [15].

Las redes neuronales convolucionales[1] están compuestas de una o más capas convolucionales y seguidas de una o más capas conectadas como una red neuronal multicapa estándar. La arquitectura de una CNN está diseñada para aprovechar la estructura 2D de una imagen de entrada. Una ventaja es que son más fáciles de entrenar y tienen muchos menos parámetros que redes totalmente conectadas con el mismo número de unidades ocultas.

El desarrollo de una red neuronal convolucional puede ser llevado a cabo mediante TensorFlow. En la actualidad, ha aparecido TensorFlow Graphics[2], una biblioteca que proporciona un conjunto de capas gráficas diferenciables, cámaras, luces, materiales, etc. y TensorBoard 3D para crear modelos de aprendizaje automático. Se utiliza en *Deep Learning* para tareas de aprendizaje no supervisado en tareas de visión por computador.

[1] http://deeplearning.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/

[2] https://www.tensorflow.org/graphics

3. Análisis del problema

En este apartado se procederá a describir el sistema que va a ser desarrollado y cuales serán sus requisitos funcionales y no funcionales, así como el diagrama de casos de uso y diagrama de clases conceptual.

Se trata de una etapa importante, ya que se debe definir correctamente qué es lo que se quiere conseguir y definir cuáles son los requisitos del sistema, las características de los usuarios y sus restricciones, entre otros detalles. Si se realiza de forma eficaz se puede ahorrar esfuerzo y trabajo futuro.

3.1. Descripción del sistema

El objetivo principal de este proyecto es el de desarrollar un sistema de juego que entre dentro de la línea de investigación de la gastroludología.

Se trata de crear un juego multijugador en el que se genere una experiencia mixta, en el que intervengan tanto objetos virtuales y reales y se pueda comprobar la adecuación de la realidad aumentada para implementar este tipo de entornos gastrolúdicos.

De entre las posibles alternativas de implementación de juegos gastrolúdicos se optó por adaptar un juego que fuera conocido por la mayor parte de los usuarios y cuya complejidad no fuera alta para poder tener un prototipo implementado en un corto espacio de tiempo. A su vez, el juego debía tener suficientes elementos de reto y diversión para llevar a los usuarios a un estado de flujo y de motivación intrínseca tal y como define Mirvis et al. [16]. Es necesario incorporar en el juego elementos gastrolúdicos que permitan a los jugadores hacer uso de elementos gastronómicos como parte del juego. Finalmente, también se consideró que el juego pudiera tener un objetivo de aprendizaje de hábitos de comida saludable para su posible uso futuro con usuarios en edad infantil.

Con estas premisas, la idea principal del sistema de juego es recuperar y alterar el clásico videojuego Snake mediante realidad aumentada y convertirlo en una experiencia de carácter gastrolúdico.

En el diseño del juego se optó por una versión competitiva en la que existen dos equipos, el equipo Kind (bueno) y el equipo Wicked (malicioso), de forma que los primeros quieren favorecer la energía de la serpiente y los segundos quieren perjudicarla.

Cada jugador posee una puntuación individual. En el momento que un jugador alcance una puntuación de 0 queda eliminado de la partida. La serpiente se mueve de forma aleatoria sin salirse del borde del plano. Cuando tan solo quedan jugadores de un equipo, estos ganan la partida

A nivel de reglas de juego se optó también por un diseño sencillo pero que forzara a los usuarios a hacer uso de la realidad aumentada y de los elementos gastronómicos del juego. En este sentido las reglas del juego son las siguientes: si un jugador del equipo Kind se graba cogiendo del espacio de juego un elemento de comida o bebida saludable, su puntuación se verá incrementada y la energía de la serpiente aumenta. Si un jugador del equipo Wicked intercepta esta acción, la energía de la serpiente disminuye y la puntuación del jugador del equipo Kind decrece.

En cambio, cuando un jugador del equipo Wicked se graba cogiendo un alimento considerado no saludable, la energía de la serpiente disminuye y su puntuación aumenta, a menos que un jugador del otro equipo intercepte esta acción, con lo cual la puntuación del jugador disminuirá y la energía de la serpiente crece. La energía de la serpiente repercute en su velocidad de movimiento.

Con estas sencillas reglas se pretende que los jugadores puedan diseñar estrategias conjuntas de consumo de elementos gastronómicos o de interceptación mediante el uso de la realidad aumentada y la visión por computador para la detección de las acciones de cada usuario sobre el espacio compartido de juego en tiempo real.

3.2. Especificación de requisitos

La especificación de requisitos de un sistema software define qué requisitos debe cumplir el sistema para satisfacer al usuario y que restricciones presenta. Tiene como entrada diferentes objetivos, requisitos del sistema, propiedades relevantes del dominio, etc. Se define siguiendo una estructura definida por el modelo de especificación de requisitos del estándar IEEE 29148:2011. En este documento se describen detalladamente los requisitos funcionales y no funcionales del sistema.

3.2.1. Características de los usuarios

Únicamente existe un tipo de usuario en nuestro sistema. Este puede crear o unirse a una sala de juego, y escoger a qué equipo pertenece, así como especificar donde se depositará el tablero de juego. Los usuarios pueden obtener un ítem gastronómico o interceptar uno mediante la cámara del dispositivo móvil.

3.2.2. Restricciones

El sistema debe ser accesible desde un dispositivo móvil con sistema operativo Android compatible con el servicio de realidad aumentada ARCore. Por esta razón, se debe utilizar un motor de videojuego y bibliotecas de desarrollo compatibles. El juego debe estar disponible en cualquier momento.

El sistema debe ser multijugador y permitir a los diferentes jugadores observar la misma información digital de realidad aumentada en sus dispositivos si están en una misma sala.

Será capaz de detectar que un jugador obtiene un ítem gastronómico mediante la cámara del dispositivo.

3.2.3. Requisitos funcionales

En este apartado se definen los requisitos funcionales del sistema o servicios que se espera que el sistema proveerá. Estos describen la funcionalidad del producto.

Identificación del requisito	**RF01**
Nombre del requisito	Crear una sala de juego
Descripción del requisito	El usuario puede crear una sala de juego multijugador que será observada por otros usuarios y donde se desarrolla una partida
Requisito no funcional	RNF01
Prioridad del requisito	Alta

Identificación del requisito	**RF02**
Nombre del requisito	Unirse a una sala de juego
Descripción del requisito	Un usuario puede unirse a una sala de juego creada por otro usuario y observar cuantos jugadores están dentro de esa sala en ese momento
Requisito no funcional	RNF01
Prioridad del requisito	Alta

Identificación del requisito	**RF03**
Nombre del requisito	Escoger equipo
Descripción del requisito	En el momento en que un jugador accede a una sala de juego, puede elegir a qué equipo quiere pertenecer
Requisito no funcional	-
Prioridad del requisito	Media

Identificación del requisito	**RF04**
Nombre del requisito	Colocar tablero de juego
Descripción del requisito	El usuario que ha creado la sala de juego coloca el plano donde se desarrolla la acción de realidad aumentada. Este plano será observado por los usuarios que se unen a la sala
Requisito no funcional	RNF01
Prioridad del requisito	Media

Identificación del requisito	**RF05**
Nombre del requisito	Iniciar partida
Descripción del requisito	El usuario que ha creado la sala de juego puede iniciar la partida cuando se han unido otros usuarios a la sala de juego
Requisito no funcional	RNF01
Prioridad del requisito	Media

Identificación del requisito	**RF06**
Nombre del requisito	Capturar ítem gastronómico saludable
Descripción del requisito	Un usuario puede capturar con la cámara del dispositivo móvil la acción de obtener un ítem gastronómico saludable de la mesa de juego
Requisito no funcional	RNF03
Prioridad del requisito	Alta

Identificación del requisito	**RF07**
Nombre del requisito	Capturar ítem gastronómico no saludable
Descripción del requisito	Un usuario puede capturar con la cámara del dispositivo móvil la acción de obtener un ítem gastronómico no saludable de la mesa de juego
Requisito no funcional	RNF03
Prioridad del requisito	Alta

Identificación del requisito	**RF08**
Nombre del requisito	Dejar partida
Descripción del requisito	Un usuario puede dejar la partida en cualquier momento quedando eliminado de la misma
Requisito no funcional	
Prioridad del requisito	Media

3.2.4. Requisitos no funcionales

Los requisitos no funcionales son restricciones o atributos de los servicios y funciones ofertadas por el sistema. A continuación, se especifican requisitos no funcionales de nuestro sistema.

Identificación del requisito	**RNF01**
Nombre del requisito	Interoperabilidad
Descripción del requisito	Capacidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada
Prioridad del requisito	Alta

Identificación del requisito	**RNF02**
Nombre del requisito	Capacidad para ser modificado
Descripción del requisito	Capacidad del producto que permite que sea modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño
Prioridad del requisito	Media

Identificación del requisito	**RNF03**
Nombre del requisito	Adaptabilidad
Descripción del requisito	Capacidad del producto que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso
Prioridad del requisito	Alta

3.3. Diagrama de casos de uso

Seguidamente, se detalla el diagrama de casos de uso del sistema. Únicamente existe un actor en nuestro sistema, un jugador. Un jugador puede crear una habitación de juego, y unirse a un equipo.

Además, una vez se encuentra en un equipo, puede obtener o interceptar un ítem gastronómico.

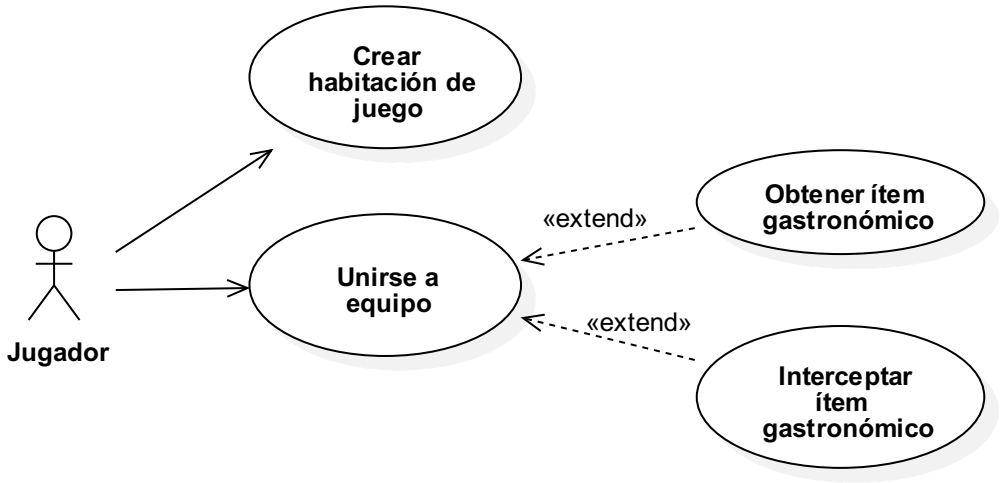


Figura 4 Diagrama de casos de uso

3.4. Diagrama de clases conceptual

A continuación, se define el diagrama de clases conceptual o modelo de dominio de nuestro sistema. Este tipo de diagrama se centra en el dominio y contexto, y no en aspectos de diseño.

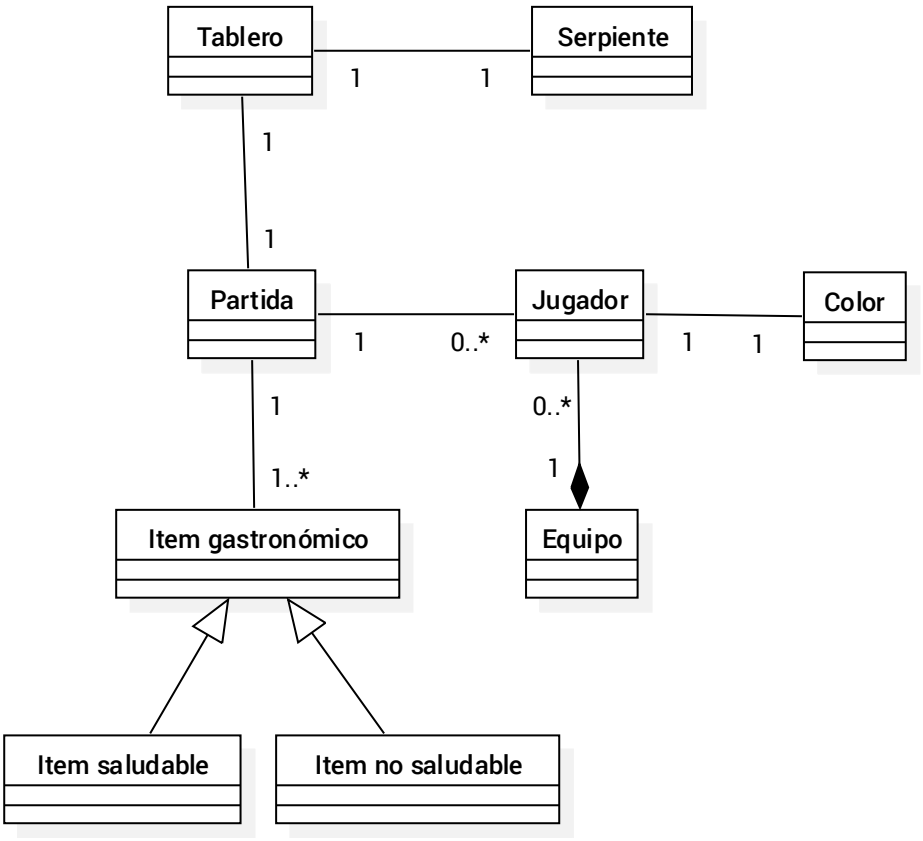


Figura 5 Diagrama de clases conceptual

4. Diseño

En este capítulo se detalla la etapa de diseño de nuestro sistema. Por un lado, se especifica el desarrollo de *mockups* en el diseño de interfaces de usuario, con el objetivo de realizar una representación del aspecto que debe presentar el sistema.

Por otro lado, se lleva a cabo un diseño arquitectónico, especificando qué arquitectura se escoge para el desarrollo del sistema multijugador.

4.1. Diseño UI

El diseño de interfaces de usuario es una etapa importante en el desarrollo de un sistema que implica el desarrollo de *wireframes* o *mockups* que permiten previsualizar cuál es el aspecto que debe presentar nuestro proyecto.

Mientras que un *wireframe* es un diseño de baja fidelidad que muestra una representación gráfica de una aplicación y muestra los elementos más esenciales y contenido, o lo que es lo mismo, la estructura de la interfaz, un *mockup* es una forma visual en media o alta fidelidad de representar un producto.

Actualmente, existe una mayor incidencia en el diseño de *mockups* ya que existen diferentes herramientas para el diseño y gestión. Algunas de estas aplicaciones son Sketch, Figma o InVision App. Para el diseño de las interfaces de usuario de este proyecto se ha utilizado Sketch.

Se han diseñado las interfaces de usuario correspondientes al menú principal y a la pantalla principal de juego. El diseño está realizado bajo el requisito de qué es un juego para móvil.

Asimismo, se ha definido la paleta de colores y fuente que deberá utilizar la interfaz del sistema.

4.1.1. Fuente

4.1.2. Paleta de colores

A continuación, se exponen muestras de colores elegidos para diferentes componentes de la interfaz de usuario.

- **Equipo:**

- **Botones:**

- **Texto:**

4.1.3. Menú inicial

La primera interfaz que observará el cliente será el menú inicial o *lobby screen* donde aparecerán las partidas a las que puede unirse el jugador. La información sobre cada partida es, su número de jugadores actuales, un botón para unirse a la partida, y el nombre de la partida. El usuario podrá crear una nueva partida mediante el botón de *"Create Room"*. Tanto si se une a una partida como crea una nueva sala, entrará al juego. En la parte inferior de la pantalla puede observar el estado de su sesión, es decir, si está conectado o desconectado del servidor multijugador. Cuando el estado es desconectado, el botón para crear una partida será deshabilitado y la lista de partidas disponibles estará vacía.

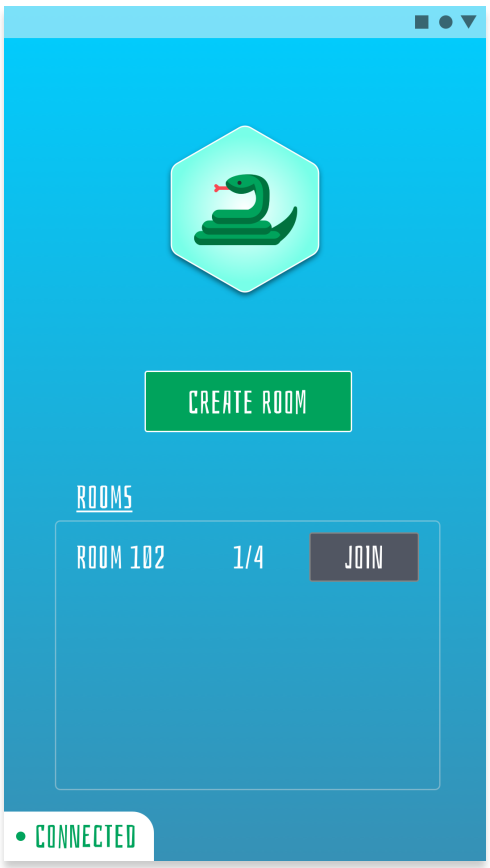


Figura 6 Prototipo de menú inicial

4.1.4. Escena de juego

La escena de juego es la interfaz donde se desarrolla la partida. Debe mostrar información sobre el jugador, el equipo al que pertenece, su color asignado y la puntuación. Esta información se agrupa en un HUD (*Head-Up Display*) que se encuentra en la parte superior izquierda.

También se debe mostrar información sobre el estado de energía de la serpiente que recorre el tablero. El valor se muestra en la parte superior central de la pantalla.

Además, el usuario puede observar información sobre la sala de juego en la que se encuentra. Estos datos corresponden al número de la sala de juego, y al número de jugadores que en ese momento se encuentran jugando.

La información sobre el posicionamiento del tablero y su registro, así como la resolución de su posición se detalla mediante la barra inferior o *snackbar*.

Figura 7 Prototipo de escena de juego

4.2. Diseño arquitectónico multijugador

Existen diferentes opciones a la hora de seleccionar la arquitectura de un sistema multijugador.

Por un lado, la arquitectura basada en un servidor dedicado. Esta arquitectura se basa en tener un servidor o servicio web dedicado al alojamiento de nuestro sistema. Este ejecuta constantemente una instancia del videojuego y está recibiendo las múltiples conexiones que realizan otros jugadores. Esto provoca una enorme dependencia al servidor dedicado, ya que conlleva el mantenimiento constante y si existe un problema de conexión, el sistema deja de estar disponible para los clientes. Además, a esto se suma el coste adicional de mantener un servidor dedicado. No obstante, esta solución ofrece múltiples ventajas ya que normalmente se elige como servidor un dispositivo con una elevada potencia de cálculo que garantiza que el rendimiento del sistema no se vea afectado por el hardware de nuestro servidor. También ofrece gran escalabilidad en sistemas que dependen de múltiples usuarios conectados al mismo tiempo.

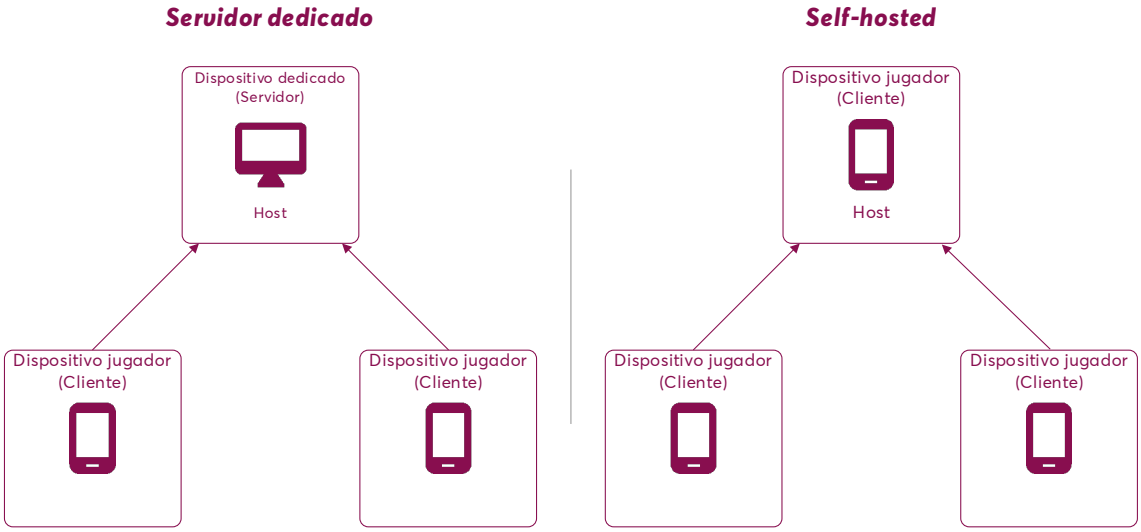


Figura 8 Diagramas de arquitectura sistema multijugador

Por otro lado, se encuentra la arquitectura en que uno de los dispositivos que ejecuta el sistema se convierte en *host* y el resto de los jugadores se convierten en clientes estableciendo una conexión con él. Esto ahorra tiempo y coste de mantenimiento, y hace menos vulnerable al sistema, al no tener un servidor que puede ser hackeado. Por otra parte, este diseño convierte al *host* en un elemento dinámico, con lo que, si un dispositivo que aloja el juego se desconecta, existen mecanismos para que otro jugador pase a ser el *host*. Esta arquitectura, sin embargo, es menos consistente, ya que pueden existir problemas de conexión debido a que participen dispositivos antiguos o con una pobre conexión a internet.

5. Implementación

En este capítulo se describe la implementación del sistema. Una vez definidos los requisitos y características funcionales del sistema, se puede proceder a la producción del software.

Se inicia especificando qué tecnologías y herramientas van a ser utilizadas en el desarrollo del proyecto, además de discutir sobre las posibilidades que existen. A continuación, se expone cómo se han integrado cada una de las tecnologías y plataformas empleadas en nuestro entorno de desarrollo, para finalizar con una especificación del desarrollo de la lógica del juego, donde se muestran GameObjects y Prefabs implementados.

5.1. Tecnologías y herramientas

Dada la naturaleza del sistema, se debe decidir qué tecnologías y herramientas van a elegirse para proceder a la implementación del sistema. Es importante conocer qué plataformas y bibliotecas están disponibles para la consecución del proyecto, y analizar qué características ofrecen, para poder descartar o adoptarlas.

5.1.1. Motor de videojuego

Al tratarse de un videojuego se debe decidir qué motor de videojuegos será utilizado. Existen dos candidatos principales, Unity y Unreal Engine.

Unity posee una documentación más legible y mejor estructurada que Unreal Engine. Unity y Unreal soportan tanto las últimas bibliotecas de realidad aumentada para dispositivos móviles como algunas de las plataformas más importantes de visión por computador.

El lenguaje de programación de Unity es C#, mientras que el de Unreal es C++. En cuanto a la curva de aprendizaje, esta es más pronunciada en el caso de Unreal.

Para el desarrollo de este proyecto se puede elegir una de las dos plataformas, pero ya que Unity posee una mejor documentación, una menor curva de aprendizaje y su lenguaje es C#, se elegirá este motor para el desarrollo del proyecto.

5.1.2. Realidad aumentada

Por otra parte, este sistema incorpora tecnología de realidad aumentada para observar los componentes u objetos de juego que forman la experiencia de juego. En este caso se debe escoger una biblioteca de realidad aumentada para dispositivos móviles.

Como hemos especificado anteriormente existen diferentes plataformas para integrar la realidad aumentada en nuestro sistema. Dada la naturalidad del juego, un sistema multijugador, se debe seleccionar una tecnología que permita visualizar la información digital representada por modelos tridimensionales en cualquier dispositivo que se encuentre participando, en tiempo real, y se debe respetar el sistema de referencia para que cada observador perciba la posición y rotación de los objetos desde su ubicación de forma correcta. Asimismo, la tecnología debe ser compatible con dispositivos móviles, cuyo sistema operativo sea Android. Por último, debe soportar el motor de videojuegos escogido.

Bajo estas condiciones, la biblioteca escogida es Google ARCore. Esta biblioteca permite disfrutar de una experiencia de realidad aumentada compartida gracias a Cloud Anchors. Con Cloud Anchors, un dispositivo envía la posición de un objeto que sirve de "ancla" y puntos característicos cercanos a la nube para almacenarlos. Esta información se puede compartir con otros usuarios en dispositivos Android o iOS en el mismo entorno. Esto permite que las aplicaciones representen los mismos objetos 3D cerca de este anclaje, lo que permite a los usuarios tener la misma experiencia AR simultáneamente.

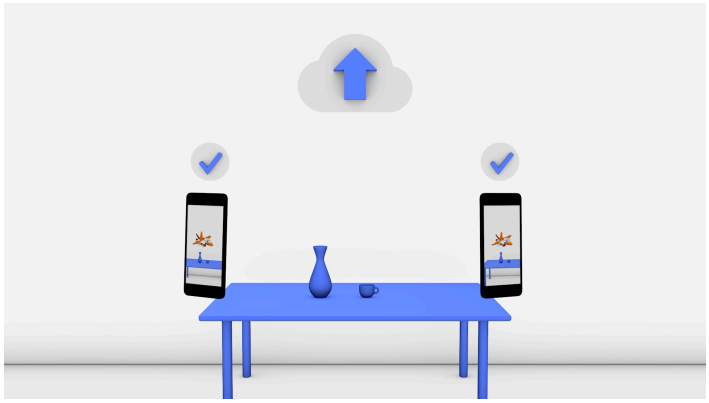


Figura 9 Google Cloud Anchors

ARCore tiene, además, compatibilidad con Unity y dispositivos Android. Otras posibles plataformas son ARKit y Vuforia. La primera de ellas si que posee compartición de experiencia de realidad aumentada mientras que Vuforia no. Sin embargo, ARKit únicamente es compatible con sistemas iOS, por lo que no puede ser adoptada. Por otra parte, dado que Vuforia no tiene un sistema similar a Cloud Anchors, también sería descartada.

5.1.3. Visión por computador

Una biblioteca de visión por computador debe seleccionarse para capturar el movimiento de los jugadores pues no hay todavía una implementación de serie en la biblioteca de ARCore. Se utiliza para poder conocer qué jugador toma un determinado elemento gastronómico de un plato/vaso y quién es. Se deberá obtener el *stream* de la cámara y procesarlo para desencadenar eventos.

La detección de movimiento y de objetos es una de las características principales de la visión por computador. Existen diferentes alternativas que pueden ser adoptadas para el desarrollo del proyecto.

En primer lugar, dado que necesitamos seguir el movimiento de las manos, podemos optar por un *framework* de reconocimiento de gestos, como ManoMotion [1]. ManoMotion ofrece una solución para poder analizar en tiempo real gestos y movimientos de nuestras manos a partir de una imagen RGB que proporciona la cámara de nuestro *smartphone*.

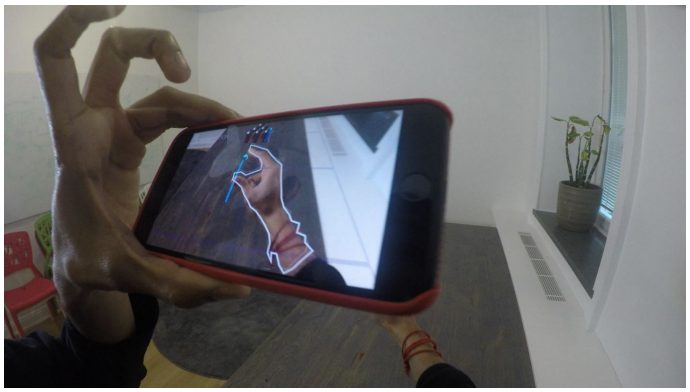


Figura 10 ManoMotion

Esta plataforma presenta diferentes planes, según el ámbito de uso, personal, profesional, empresarial, etc. Tanto el plan personal como el profesional únicamente detectan una mano, izquierda o derecha, el contorno, la punta de los dedos y los gestos realizados. El plan empresarial permite reconocer dos manos y definir gestos propios. El mayor inconveniente de esta solución para nuestro sistema es que únicamente permite el seguimiento de una mano y en realidad, no existe una enorme preferencia por los gestos realizados que es el punto fuerte de esta plataforma. Por otra parte, se trata de una biblioteca cuyo SDK debe ser solicitado y, tiempo después, los desarrolladores lo envían a los interesados. Es decir, resulta difícil obtener la biblioteca inmediatamente por lo que no es adecuado adoptar esta tecnología en nuestro sistema.

Otra posible alternativa es TensorFlow, la librería *open source* de *machine learning* de Google. Esta librería permite la detección y clasificación de múltiples objetos en una

[1] https://www.manomotion.com/

escena. Esto significa que sería capaz de detectar diferentes manos, pero no es capaz de distinguir qué mano de qué jugador está observando si necesitamos diferenciar entre jugadores para desencadenar eventos diferentes en el juego. Esto limita la adopción de esta librería en nuestro proyecto. No es necesario, dados los requisitos, utilizar la inteligencia artificial, sino que únicamente basta con emplear algoritmos de una biblioteca de visión por computador.

Un buen candidato es OpenCV. Se trata de una biblioteca que posee diversos algoritmos de visión por computador para alcanzar nuestro propósito. Además, existe compatibilidad con el sistema operativo y motor de juego escogido, ya que existe un *asset* llamado OpenCV plus Unity, gratuito, que adapta el *wrapper* OpenCVSharp al entorno Unity.

Esta biblioteca permite utilizar detección de objetos por color, con lo cual, dado que en nuestro juego tendremos diferentes usuarios, podemos detectar mediante elementos de color que los usuarios tendrán en las puntas de los dedos quién realiza qué acción. Esto soluciona el problema de tener que detectar diferentes manos en la escena ya que, si cada jugador tiene un color asignado y un elemento de color en su mano, se puede conocer quién es y detectar la acción.

Además, el juego se plantea de forma que existen diferentes puntos sobre una mesa sobre los que se deposita comida y según qué comida se coja, repercute o se desencadena un evento sobre el jugador. Esto significa que se debe conocer las diferentes áreas, y el sistema debe reconocer a qué plato o elemento está apuntando el jugador. En OpenCV existe una forma de conseguir este objetivo y es utilizando los marcadores de realidad aumentada ArUco.

ArUco es una biblioteca para desarrollar aplicaciones de realidad aumentada basada en OpenCV. La gran característica de esta biblioteca son los diccionarios de marcadores de realidad aumentada que ofrecen. Se trata de un conjunto de marcadores de diferentes tamaños, que son reconocidos mediante OpenCV y que permiten, entre otras cosas, determinar qué marcador se está visualizando y qué posición y rotación posee. Cada marcador tiene un identificador asociado, con lo cual se puede distinguir cerca de qué marcador está la mano de un jugador.

Existen diferentes tipos de marcadores ArUco, los marcadores individuales, los cuales se detectan de forma independiente; los ChArUco, basados en una estructura de tablero de ajedrez con diferentes marcadores en las casillas blancas; o *diamond markers*, formados por cuatro marcadores; así como los *fractal markers* [17], marcadores dentro de otros. Una solución a nuestro problema es utilizar marcadores simples que formen un área donde se colocará la comida o elemento de juego y detectar cuando una mano está dentro de esta región.

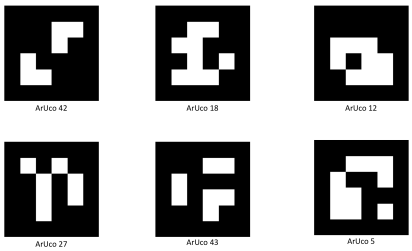


Figura 11 ArUco markers

Cabe destacar que ARCore ofrece detección de imágenes, las cuales deben estar previamente almacenadas en una base de datos. Esta solución, sin embargo, no es adecuada para nuestro sistema pues la detección de la imagen se produce cuando ésta ocupa al menos un 25% del fragmento de la cámara. Los marcadores de ArUco, en cambio, pueden ser detectados, aunque su tamaño sea pequeño o se visualicen a una distancia más prominente. Vuforia también presenta esta característica incluso con oclusión de parte del marcador y sin la limitación de ARCore. Sin embargo, existen dificultades para integrar ambas plataformas en el mismo proyecto y utilizar algunas de las características de ARCore.

Actualmente, existen otros diferentes tipos de marcadores dentro del ámbito de la visión por computador. Podemos destacar los marcadores de ARTag [18], que garantiza identificación rápida de marcadores, ya que no requiere hacer coincidir su imagen de marcador interno con una biblioteca de plantillas o RUNE-Tag [19] que soluciona de forma muy eficaz el problema de la oclusión. El principal inconveniente de empleo de estos marcadores es la integración en Unity ya que son bibliotecas escritas en C++ que todavía no tienen una adaptación para el motor de videojuego y su uso en C# conllevaría el esfuerzo de adecuación a este lenguaje.

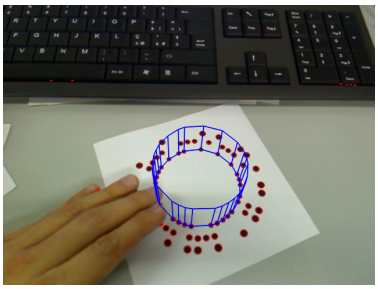


Figura 12 RUNE-Tag

5.1.4. Multijugador

Se trata de un sistema de naturaleza multijugador, con lo que se debe integrar una plataforma que permita crear y gestionar una partida multijugador. Según los requisitos debe acoger como máximo 4 jugadores y estos deben formar equipos. Para llevar a cabo el desarrollo de un sistema multijugador en Unity existen diferentes opciones, entre ellas podemos destacar UNet y Photon Cloud.

Por un lado, UNet es una plataforma desarrollada por Unity, la cual forma parte de los servicios que ofrecen en el desarrollo de un juego. Sin embargo, se trata de un servicio que está deprecado a la espera de una nueva tecnología desarrollada por Unity que lo reemplace, ya que indican que UNet no satisface las necesidades de muchos creadores de juegos multijugador y que, para alcanzar los objetivos de rendimiento, escala y seguridad, se ha tomado la decisión de desarrollar una tecnología completamente nueva, incluida una red ligera y rápida, y un modelo de servidor de juegos dedicado.

Photon Cloud es una plataforma de desarrollo multijugador SaaS[1]. Las operaciones del servidor, alojamiento y escalado están a cargo del proveedor del servicio. Photon Unity Networking (PUN), es una solución para Unity que ofrece facilidades en la creación de partidas, componentes para sincronizar GameObjects o Remote Procedure Calls, llamadas a los clientes en una misma sala de juego.

Photon Cloud está construido con el concepto de partidas en mente, lo que significa que hay un número limitado de jugadores por partida, separados de cualquier otra persona. En una habitación todos reciben lo que los demás envían. Fuera de una sala, los jugadores no pueden comunicarse.

A diferencia de UNet, es un servicio que actualmente está mantenido y actualizado, y además permite la creación de salas de juego donde se limita el número de jugadores que accede a ellas. Además, el mantenimiento y administración del servidor se realiza en la nube, con lo que el desarrollador no debe preocuparse por estas tareas. En resumen, se trata de una tecnología apropiada para las necesidades del sistema.

[1] *Software as a Service*

5.2. Tecnología utilizada

A continuación, se procede a describir de qué forma se integra y configura la tecnología escogida en nuestro proyecto.

5.2.1. Google ARCore

La biblioteca de realidad aumentada desarrollada por Google está disponible para el motor de videojuegos Unity. La versión del SDK utilizada en este proyecto es la 1.11.0[1]. Esta versión permite disfrutar de reconocimiento de imágenes, Google Cloud Anchors, reconocimiento de caras, una versión inicial de visión por computador, y manipulación de objetos tridimensionales. Asimismo, permite activar reflejos y sombras en objetos, en determinadas situaciones de luz, así como activar una tasa de fotogramas de la cámara de 60 fotogramas por segundo.

5.2.2. Google Cloud Anchors

Google Cloud Anchors es una característica de realidad aumentada que permite disfrutar de experiencias de realidad aumentada en diferentes dispositivos móviles. Utiliza Google Cloud Platform como infraestructura en la nube para almacenar la información sobre los puntos clave y el objeto que sirve de origen de nuestro mundo de realidad aumentada.

Para disfrutar de esta funcionalidad, debemos obtener una API *Key* de esta biblioteca en Google Cloud Platform.

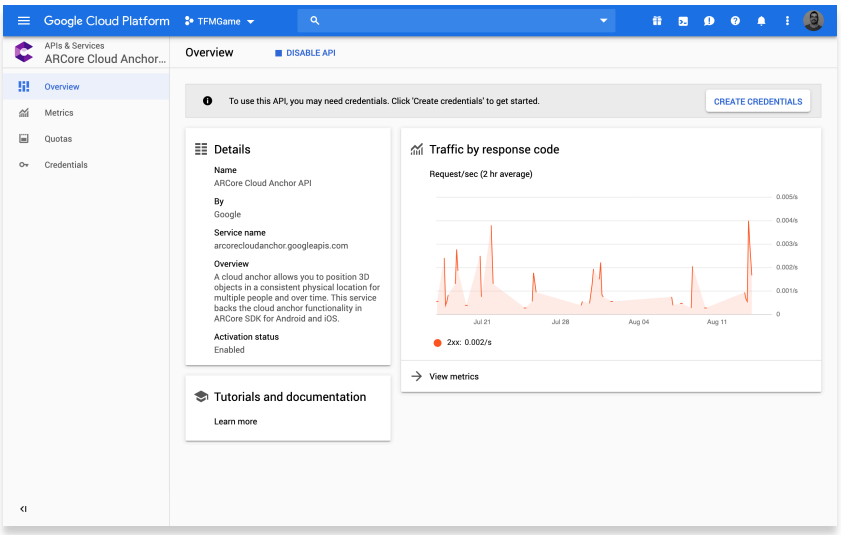


Figura 13 Google Cloud Anchors API Key

[1] https://github.com/google-ar/arcore-unity-sdk/releases/tag/v1.11.0.1

Esta credencial se debe copiar y pegar en nuestro entorno de Unity, así como activar esta característica en el fichero de configuración de Google Cloud ARCore.

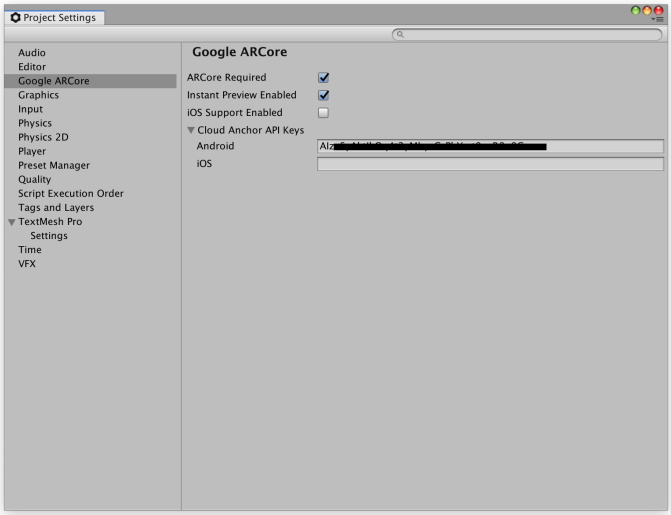


Figura 14 Importar API Key en Unity

5.2.3. Photon Cloud PUN

Photon Cloud es la plataforma escogida para establecer el sistema multijugador del sistema. La ventaja principal es que es un servicio SaaS, es decir, las acciones del servidor, alojamiento y escalado están a cargo del proveedor del servicio y el cliente no debe preocuparse de su diseño de la arquitectura, desarrollo y mantenimiento. PUN es la solución de Photon Cloud para Unity, y ofrece la posibilidad de crear experiencias multijugador, así como una gran escalabilidad e información en tiempo real. Para poder incorporar esta plataforma en nuestro proyecto debemos crear una cuenta en el servicio de Photon y crear una nueva aplicación de tipo Photon PUN.

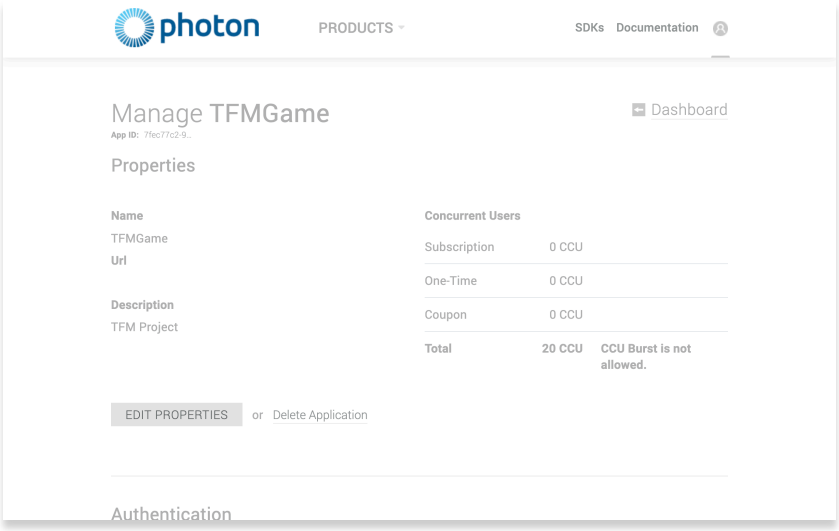


Figura 15 Gestión del proyecto en el Dashboard de Photon Cloud

La App ID generada para nuestro proyecto deberá ser copiada. A continuación, mediante el Asset Store de Unity, importaremos el paquete PUN 2. Entonces nos pedirá que introduzcamos la clave copiada.

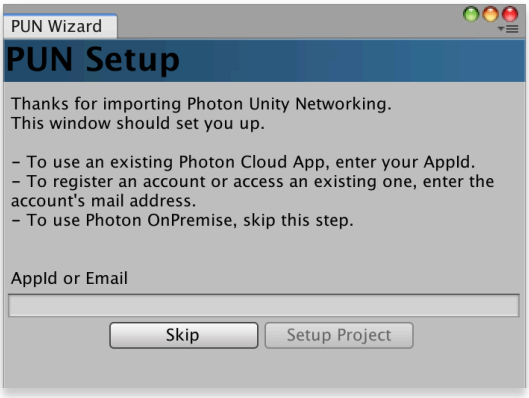


Figura 16 Configuración de PUN en Unity

5.2.4. OpenCV plus Unity

Existen diferentes adaptaciones de la biblioteca OpenCV para Unity. Algunos de estos paquetes no son gratuitos como OpenCV for Unity o Emgu CV. En este proyecto se utiliza OpenCV plus Unity[1], que utiliza el *wrapper* de OpenCVSharp, el cual sí es gratuito. Permite la detección de contornos y objetos, así como de marcadores de realidad aumentada.

[1] https://assetstore.unity.com/packages/tools/integration/opencv-plus-unity-85928

5.3. Diseño e implementación de la lógica del juego

Una vez configurado el entorno de desarrollo y añadidas las dependencias, se puede pasar a la implementación de la lógica del juego.

A continuación, se describen los GameObjects y Prefabs que constituyen el sistema y los detalles importantes de cada uno de ellos.

5.3.1. GameObjects implementados

- **Photon Connection:** Clase encargada de establecer la conexión con el servicio de Photon. Define el número de versión del juego que utiliza Photon y realiza la conexión utilizando el fichero de configuración de Photon.

  Hereda de MonoBehaviourPunCallbacks para atender a algunos eventos que Photon puede invocar.

  Cuando se instancia este objeto se define el número de versión del juego para Photon y se conecta utilizando el fichero de configuración de Photon definido en la carpeta del proyecto.

  Cuando se conecta al servidor de Photon se une entonces al Lobby. El Lobby es la sala de espera para Photon, el usuario se encuentra en el Lobby y puede unirse a una habitación de juego o crear una.

```
public string versionName = "0.1";

private void Awake()
{
    PhotonNetwork.GameVersion = versionName;
    PhotonNetwork.ConnectUsingSettings();
}

public override void OnConnectedToMaster()
{
    PhotonNetwork.JoinLobby(TypedLobby.Default);
}
```

- **PhotonManager:** Clase encargada de atender diferentes operaciones relacionadas con Photon y el sistema multijugador del juego. Este GameObject se instancia en la primera escena del sistema, el menú principal, y no se destruye en la carga de la escena del juego para poder seguir teniendo acceso a sus métodos.

  Algunas de las operaciones que se realizan desde esta clase son lanzar la creación de una habitación de juego, dejar una habitación de juego, cambiar la visibilidad

de una habitación u obtener el número de jugadores y nombre de la habitación a la que se está conectado.

Cuando se accede a una habitación de juego, es la clase encargada de instanciar un jugador utilizando el Prefab de jugador.

- **MenuUIController:** Clase encargada de controlar la visibilidad y las acciones que lanzan los elementos de interfaz gráfica del menú de usuario principal. Cuando un usuario accede al sistema, se encuentra con un menú principal o *lobby*. Puede crear una habitación de juego o unirse a una habitación creada a partir de una lista.

  En esta clase se define el método lanzado cuando se pulsa en los botones para crear una partida o para unirse a una. Asimismo, se controla la disponibilidad de estos botones en función de la conexión de Photon.

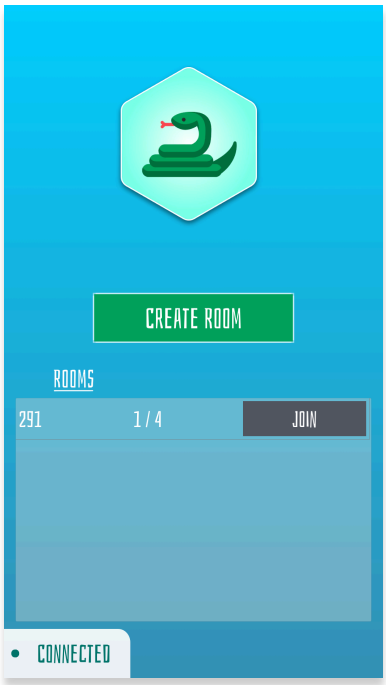


Figura 17 Menú principal

- **GameManager:** La clase GameManager es la responsable de controlar cuántos jugadores hay en cada equipo, conocer el siguiente color que será asignado a un jugador cuando se une a la partida, y controlar en qué estado se encuentra el juego (inicio, jugando, fin). Implementa el patrón de diseño *singleton*, para poder acceder siempre a la misma instancia de este objeto y evitar diferentes valores del estado del juego o número de jugadores.

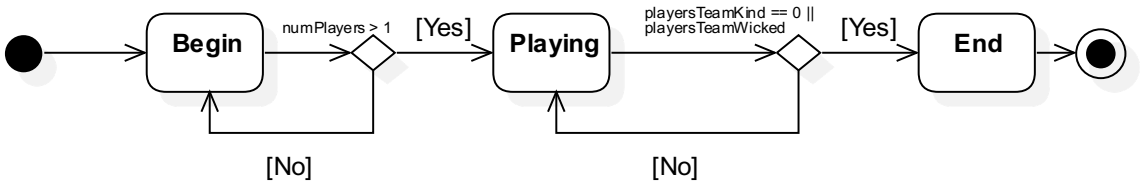


Figura 18 Máquina de estados

Cuando únicamente quedan jugadores de un equipo en el estado jugando se da por finalizada la partida y este equipo habrá ganado.

- **ComputerVisionController:** Clase encargada de controlar todos los aspectos relacionados con la visión por computador del sistema. Se activa el procesamiento de imagen cuando el estado del juego del GameManager pasa a Playing. La imagen procesada proviene del *stream* de la cámara del dispositivo móvil.

  La biblioteca de Google ARCore posee un Prefab para la cámara, que está preconfigurada para ser utilizada como cámara principal de la escena y procesar la imagen para integrar la realidad aumentada. La cámara predeterminada que está presente en la creación de una escena se sustituye por la cámara de Google ARCore. La biblioteca ofrece una clase, TextureReader, que permite obtener la imagen de la cámara para poder ser procesada. Para activar la captura de la imagen desde la cámara, cuando el estado del juego pasa a jugando, se suscribe al evento OnImageAvailableCallback de TextureReader.

  Esta clase está involucrada en los casos de uso de obtener un ítem gastronómico e interceptar un ítem gastronómico.

  Los diferentes ítems gastronómicos se identifican mediante marcadores de ArUco[1]. Es un marcador cuadrado compuesto por un borde negro ancho y una matriz binaria interna que determina su identificador. El tamaño del marcador determina el tamaño de la matriz interna. Por ejemplo, un tamaño de marcador de 4x4 está compuesto por 16 bits. Un diccionario de marcadores es un conjunto de marcadores. Las propiedades principales de un diccionario son el tamaño del diccionario y el tamaño del marcador. Dada una imagen, el proceso de detección devuelve una lista de los marcadores detectados. Para cada marcador, devuelve el identificador y la posición de las cuatro esquinas.

[1] https://docs.opencv.org/3.1.0/d5/dae/tutorial_aruco_detection.html

```
Point2f[][] markers;
int[] ids;
Point2f[][] rejectedImgPoints;

var detectorParameters = DetectorParameters.Create();
var dictionary =
CvAruco.GetPredefinedDictionary(PredefinedDictionaryName.Dict6X6_250);

var mat = OpenCvSharp.Unity.TextureToMat(_texture);
Cv2.Rotate(mat, mat, RotateFlags.Rotate90Clockwise);
Cv2.Flip(mat, mat, FlipMode.Y);

var grayMat = mat.CvtColor(ColorConversionCodes.RGBA2GRAY);
CvAruco.DetectMarkers(grayMat, dictionary, out markers, out ids,
detectorParameters, out rejectedImgPoints);
```

La textura del *stream* de cámara se transforma a Mat[1]. Se trata de la clase con la que trabaja OpenCV para procesar una imagen. Representa un conjunto numérico denso monocanal o multicanal de n dimensiones. Se puede usar para almacenar vectores y matrices de valores reales o complejos, imágenes en escala de grises o en color, campos de vectores, nubes de puntos o histogramas.

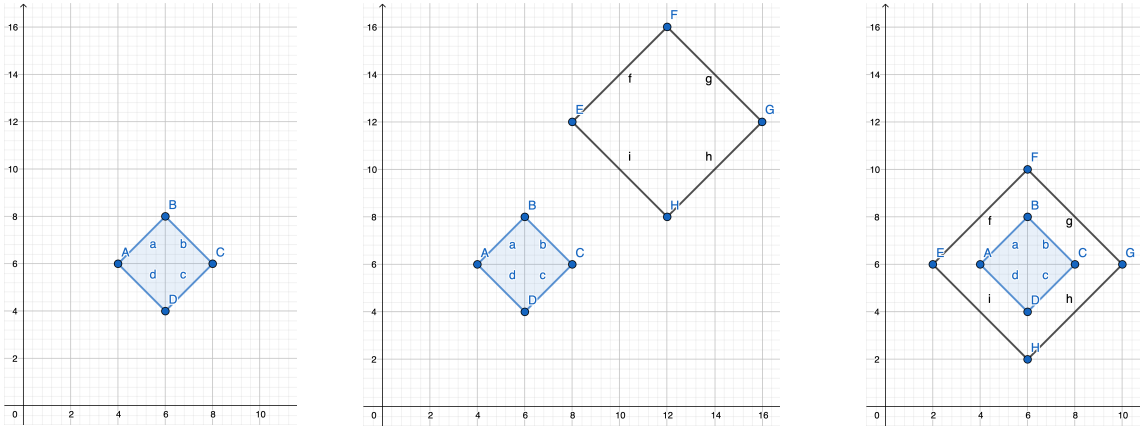
La detección de un marcador se realiza sobre una imagen en escala de grises, con lo que se convierte a este formato de color y se lanza la detección de marcadores, devolviendo la lista de identificadores y la de las posiciones de las esquinas de cada marcador.

Esta operación permite conocer en qué posición se encuentra un marcador, pero nos interesa saber que un jugador obtiene un ítem real, con lo que debemos detectar el momento en que la mano de un jugador se encuentra en el área donde está un elemento de comida o bebida. La limitación de esta biblioteca es que no permite que el marcador pueda ser ocluido o tapado parcialmente, con lo que se debe colocar los elementos de juego alrededor del marcador o cerca de este y ampliar el área de detección de cada marcador para detectar cuando la mano de un jugador está dentro.

Para cada marcador detectado se debe calcular un área alrededor de detección de la mano de un jugador. Para poder realizar esta operación se necesita la posición de las cuatro esquinas de cada marcador. Las cuatro esquinas de un marcador delimitan un polígono de cuatro lados que debe ser ampliado y constituirá el área de detección en la cual se encontrarán los elementos de juego.

[1] https://docs.opencv.org/3.4.3/d3/d63/classcv_1_1Mat.html

La redimensión del polígono de un marcador se realiza multiplicando la posición de cada esquina de un marcador por un factor de ampliación. Al hacer esta operación, el polígono es desplazado también ya que el sistema de coordenadas en un dispositivo móvil tiene el origen de coordenadas en la esquina superior izquierda. Para corregir esto se debe calcular el centro del marcador y de la nueva área calculada y aplicar una corrección en la posición de ésta para que ambos polígonos compartan el mismo centro.




```
for (int j = 0; j < markers.Length; j++)
{
    var markerPoints = markers[j];
    if (markerPoints.Length == 4)
    {
        var points = new Point[1][];

        points[0] = new Point[]{markerPoints[0], markerPoints[1],
                    markerPoints[2], markerPoints[3]};
        var m = Cv2.Moments(points[0]);
        Point centerA = new Point(m.M10 / m.M00, m.M01 / m.M00);

        const int factor = 4;
        points[0] = new Point[]
        {
            markerPoints[0] * factor, markerPoints[1] * factor, markerPoints[2] *
            factor, markerPoints[3] * factor
        };
        m = Cv2.Moments(points[0]);
        Point centerB = new Point(m.M10 / m.M00, m.M01 / m.M00);

        int distanceX = Math.Abs(centerA.X - centerB.X);
        int distanceY = Math.Abs(centerA.Y - centerB.Y);

        for (int i = 0; i < points[0].Length; i++)
        {
            points[0][i].X -= distanceX;
            points[0][i].Y -= distanceY;
        }

        Cv2.Polylines(mat, points, true, new Scalar(255, 255, 50), 2);
```

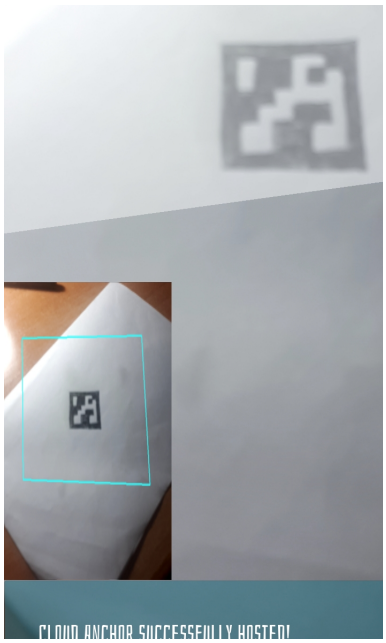


Figura 19 Detección área marcador

Cada una de estas regiones creadas será analizada en búsqueda de una mano que coja un elemento del plato. Para poder distinguir qué jugador ha cogido qué elemento se pueden utilizar regiones de colores en las manos de los jugadores. Estas regiones de color corresponden al color asignado a cada jugador y se pueden llevar en las puntas de los dedos.

OpenCV permite realizar segmentación de imágenes basado en color. Esta técnica sirve para poder localizar objetos, curvas y límites. Sin embargo, esto no garantiza que siempre que se detecte un color corresponda a una región de color en la mano de un jugador. Esto se puede solucionar utilizando los blobs de OpenCV.

Un blob (Binary Large OBject) consiste en un conjunto de píxeles conectados en una imagen binaria. Permite seleccionar solo aquellos segmentos de color de una determinada área y obviar objetos pequeños que normalmente son ruido. Los blobs son extraídos y clasificados según características como el área, la circularidad, el perímetro, la convexidad, etc.

OpenCV ofrece un detector de blobs, el SimpleBlobDetector, que permite filtrar en base a diferentes propiedades[1]:

- Color: Se utiliza un valor de 0 para extraer blobs oscuros y un valor de 255 para blobs claros.

[1] https://docs.opencv.org/2.4/modules/features2d/doc/common_interfaces_of_feature_detectors.html#SimpleBlobDetector%20:%20public%20FeatureDetector

- Área: Extrae blobs que posean un área entre minArea y maxArea.
- Circularidad: Extrae blobs que tengan una circularidad entre un valor mínimo y máximo.
- Convexidad: Extrae blobs que tengan un valor de convexidad, área entre área de la envolvente convexa entre un rango de valores.

Nos interesa extraer blobs en base a un área, con lo que definiremos el filtro de área activado y un valor mínimo y máximo.

La detección se realiza sobre una imagen binaria, con lo que se debe convertir el tipo de formato del objeto Mat. El formato de imagen para convertirla en binaria es HSV (Hue Saturation Value). HSV[1] es un modelo de color alternativo a RGB creado por investigadores de gráficos por computador para acercarse más a la forma en que la visión humana percibe los atributos de creación de color. Normalmente se representa como una figura tridimensional cilíndrica donde la región de corte radial es el tono, y el corte longitudinal define en dos ejes la saturación y el brillo. La representación en tres canales del modelo RGB, en cambio, conlleva más dificultad para segmentar una imagen.

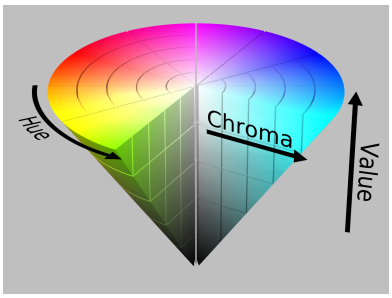


Figura 20 Modelo de color HSV

OpenCV permite convertir el modelo de color a HSV y aplicar un rango de valores de color para obtener una imagen binaria.

```
var hsv = outputMat.CvtColor(ColorConversionCodes.BGR2HSV);
var bin = hsv.InRange(GameColorHSV.GetHSVLowerBound(blobColor),
                      GameColorHSV.GetHSVUpperBound(blobColor));
```

La imagen en HSV se convierte en una imagen binaria mediante la aplicación de un espacio de color dentro del cual debe estar el objeto.

Se ha creado una clase llamada ColorUtil que posee métodos estáticos para obtener valores de colores predefinidos en distintos modelos de color. Se han

[1] https://docs.opencv.org/3.4/da/d97/tutorial_threshold_inRange.html

definido los valores predeterminados inferiores y superiores para cada rango de color.

La representación gráfica del rango HSV del color amarillo es la siguiente:

$$\left.\begin{matrix} H = 40° \\ S = 50\% \\ V = 50\% \end{matrix}\right\} Límite\ inferior$$

$$\left.\begin{matrix} H = 60° \\ S = 100\% \\ V = 100\% \end{matrix}\right\} Límite\ superior$$

Se definen los parámetros de detección del detector, especificando el rango de valores del área y a continuación se procede a la detección. Los puntos clave son dibujados sobre la imagen en color.

```
var detectorParams = new SimpleBlobDetector.Params
{
    FilterByCircularity = false,
    FilterByInertia = false,
    FilterByArea = true,
    MinArea = 30f,
    MaxArea = 1000f,
    FilterByColor = false,
    FilterByConvexity = false
};
var simpleBlobDetector = SimpleBlobDetector.Create(detectorParams);

var keypoints = simpleBlobDetector.Detect(bin);
if (keypoints.Length <= 0) return false;
Cv2.DrawKeypoints(
    outputMat, //input image
    keypoints: keypoints,
    outImage: outputMat, //output image
    color: Scalar.Red,
    flags: DrawMatchesFlags.DrawRichKeypoints
);
return true;
```

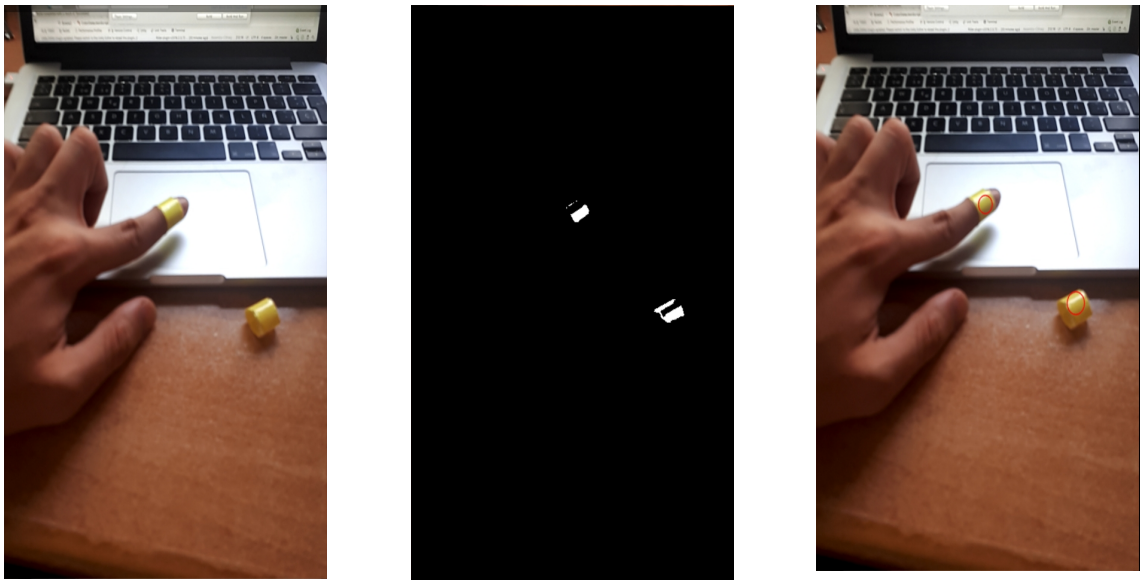


Figura 21 Imagen original – Imagen binaria con blobs – Imagen con blobs dibujados

Para cada área, se buscan blobs de los cuatro colores posibles, rojo, amarillo, azul y verde. En caso de detectar un blob, significa que la mano de un jugador está dentro de un área de marcador con lo que se debe lanzar un evento.

En caso de que el jugador sea quien obtiene un ítem gastronómico, si el color del blob es el mismo que el suyo se comprueba si pertenece al equipo Kind o Wicked. Si el marcador hace referencia a un ítem saludable y es del equipo Kind, la puntuación del jugador aumenta, así como la energía de la serpiente. Si se trata de un ítem no saludable y pertenece al equipo Wicked, su puntuación aumenta y la energía de la serpiente decrece.

En caso de interceptar alguien cogiendo un ítem, el color del blob no pertenece al del jugador local, se lanza un evento donde se detalla la intercepción, el color del jugador y el tipo ítem cogido. Los eventos son lanzados mediante Photon ya que es necesario que el evento sea capturado por todos los clientes del sistema. Los eventos en Photon se identifican mediante un número de evento, entre 0 y 199, y permiten definir el objetivo del evento y enviar contenido.

Para limitar el número de veces que se lanza una llamada de intercepción u observación de un ítem gastronómico, se introducen dos variables que son temporizadores. El usuario debe esperar a poder observar o interceptar un ítem, de esta forma se evita que constantemente esté interceptando al mismo jugador sobre el mismo marcador u observando muchísimas veces la misma acción de obtener un ítem.

- **GameSceneController:** Clase que controla la configuración de la escena de juego, y lleva a cabo la detección de planos, instanciación del plano de juego y control de operaciones que se deben de realizar en caso de entrar como jugador *master* o jugador cliente.

  Si se accede como cliente *master* de la partida, se entra en modo *hosting*. Esto significa que debe detectar planos donde puede ir el tablero de juego mostrado a los demás jugadores e instanciarlo. Si se accede como cliente, se entra en modo *resolving*. Cuando el tablero esté instanciado, podrá ser resuelta su posición y orientación.

  El cliente *master* puede pulsar sobre uno de los planos encontrados e instanciar un tablero de seguimiento o provisional. Cuando pulsa el botón de anclar el tablero, instancia el Prefab del tablero que constituye un Google Cloud Anchor que será registrado y resuelto por los demás jugadores.

  Posee, además, métodos para controlar la salida de la sala o lanzar mensajes de error cuando la biblioteca de ARCore no posee permisos.

- **ARCoreWorldOriginHelper:** Se trata de una clase Helper que proporciona la biblioteca de Google ARCore. Proporciona operaciones para definir el origen de la habitación de juego y transformar la posición y rotación local a la del mundo de juego.

  Dada la posición y rotación del objeto que se utiliza de ancla o anchor del sistema de juego, transforma una posición y rotación dada.

  Se utiliza para especificar el origen de la sala de juego una vez instanciado el tablero y para poderlo resolver en el resto de los clientes.

```
private Pose _WorldToAnchorPose(Pose pose)
{
    if (!m_IsOriginPlaced)
    {
        return pose;
    }

    Matrix4x4 anchorTWorld = Matrix4x4.TRS(m_AnchorTransform.position,
                    m_AnchorTransform.rotation, Vector3.one).inverse;

    Vector3 position = anchorTWorld.MultiplyPoint(pose.position);
    Quaternion rotation = pose.rotation * Quaternion.LookRotation(
                    anchorTWorld.GetColumn(2), anchorTWorld.GetColumn(1));

    return new Pose(position, rotation);
}
```

- **GameUIController:** Clase responsable de la interfaz de usuario del juego. Se suscribe a eventos del jugador local y de la serpiente para mostrar al usuario cambios en las propiedades, como el equipo seleccionado, el color asignado o la energía de la serpiente.

  Muestra las propiedades de la habitación de juego, número de jugadores y número de habitación.

  La interfaz posee un *snackbar* o barra inferior que muestra al usuario información dependiendo de si entra en modo *hosting* o modo *resolving*. Asimismo, muestra detalles del momento en que se ha registrado el *anchor* de la partida o se ha obtenido.

  Se encarga de cambiar la visibilidad de objetos visuales y define el cuerpo de los métodos lanzados cuando se pulsa en los botones de anclar o jugar.

  En el momento que la partida se da por terminada, muestra al usuario un mensaje emergente que especifica quien ha ganado y un botón para abandonar la sala.

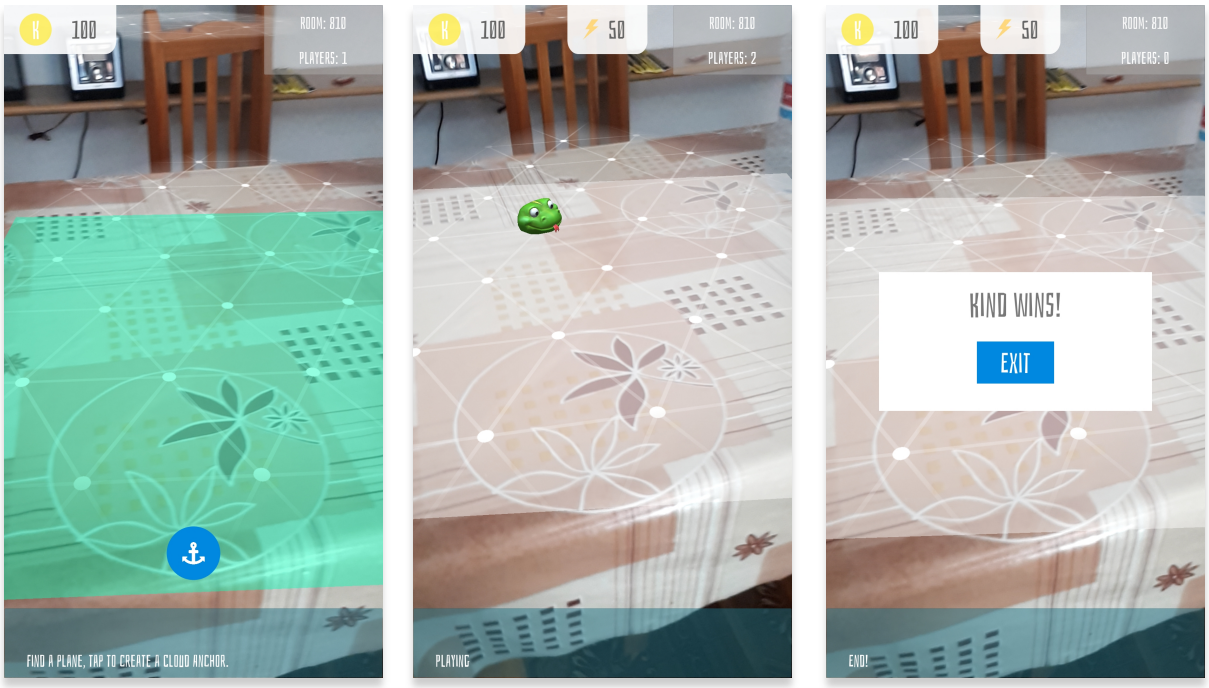


Figura 22 Escena de juego

5.3.2. Prefabs implementados

Los Prefab son los GameObjects que se almacenan con una configuración predefinida, con todos sus componentes, propiedades y GameObjects hijos como recursos reutilizables. En este sistema se han creado una serie de Prefabs para poder representar diferentes elementos que intervienen en el juego.

- **Player:** Es el Prefab que contiene la lógica de un jugador. Entre sus componentes cuenta con un PhotonView y la clase desarrollada LocalPlayerController. El componente PhotonView lo identifica a través de la red (viewID) y configura cómo el cliente que controla esta instancia envía información a las instancias remotas. Se encuentra en un directorio específico, 'Resources', para poder ser utilizado por Photon.

  La clase LocalPlayerController implementa la lógica de nuestro jugador. Cada jugador tiene un color asignado, pertenece a un equipo y posee una puntuación.

```
private void Start()
{
    photonView = GetComponent<PhotonView>();
    if (photonView.IsMine)
    {
        photonView.RPC("RPC_GetColor", RpcTarget.MasterClient);
    }
// …
```

  Para saber si un PhotonView pertenece a un cliente y puede ser controlarlo, se utiliza el método IsMine(). Si le pertenece localmente al cliente, lanza una Remote Procedure Call. Se trata de una llamada que transmite al cliente *master* que debe obtener su color.

```
[PunRPC]
private void RPC_GetColor()
{
    myColor = GameManager.Instance.nextPlayersColor;
    GameManager.Instance.UpdateColor();
    photonView.RPC("RPC_SentColor", RpcTarget.OthersBuffered, myColor);
}


[PunRPC]
private void RPC_SentColor(GameColor whichColor)
{
    myColor = whichColor;
}
```

  El *master* actualiza el color siguiente que será asignado y envía al jugador que había hecho su petición su color. Una vez éste recibe el color invoca el evento de color recibido para que el GameObject GameUIController pueda mostrar cuál es su color.

En el caso de la puntuación de un jugador, se utilizan las Custom Properties de Photon. Se trata de una Hashtable que almacena los valores para distintas propiedades del jugador local.

```
// Start method
    playerCustomProperties.Add("Score", 100);
    PhotonNetwork.LocalPlayer.SetCustomProperties(playerCustomProperties);
}
```

Para aumentar o disminuir la puntuación de un jugador se definen dos métodos que modifican el valor de la puntuación en las propiedades del jugador.

```
public void IncreaseScore(int amount)
{
    int score = (int) PhotonNetwork.LocalPlayer.CustomProperties["Score"] +
                      amount;
    PhotonNetwork.LocalPlayer.CustomProperties["Score"] = score;
}


public void ReduceScore(int amount)
{
    int score = (int) PhotonNetwork.LocalPlayer.CustomProperties["Score"] -
                      amount;
    PhotonNetwork.LocalPlayer.CustomProperties["Score"] = score;
}
```

Estos métodos son llamados cuando obtenemos un ítem gastronómico o alguien intercepta nuestra mano cogiendo un ítem. Todos los jugadores reciben los eventos que invoca un jugador cuando lleva a cabo alguna de estas acciones y si su color es igual al del Blob interceptado le afecta a su puntuación.

El equipo es seleccionado por el jugador con lo cual se debe pasar al cliente *master* qué equipo ha escogido para que éste aumente el número de jugadores asignados a ese equipo en el gestor del juego GameManager. La petición se realiza de la misma forma que la asignación de color, se pasa la selección de equipo al *master* y éste devuelve la respuesta cuando ha sido asignado. Cuando pertenece a un equipo invoca al evento de selección de equipo para mostrarlo a través de la interfaz.

Cuando un jugador deja la partida, debe notificarlo al cliente *master* para que actualice el número de jugadores actuales en cada equipo. Además, la instancia del jugador será destruida.

```
public void LeaveMatch()
{
    if (photonView.IsMine)
    {
        photonView.RPC("RPC_PlayerLeaveTeam", RpcTarget.MasterClient, myTeam);
        PhotonNetwork.Destroy(gameObject);
```

```
    }
}
```

- **Snake:** Prefab que representa a la serpiente que se mueve sobre el tablero. La serpiente es instanciada por el cliente *master* en el momento en que da inicio a la partida. Se mueve de forma aleatoria sin salir de los márgenes del tablero y modifica su velocidad en función de su energía asociada.

  El modelo tridimensional se ha obtenido a través de los recursos gratuitos que ofrece Google en su página Poly[1].

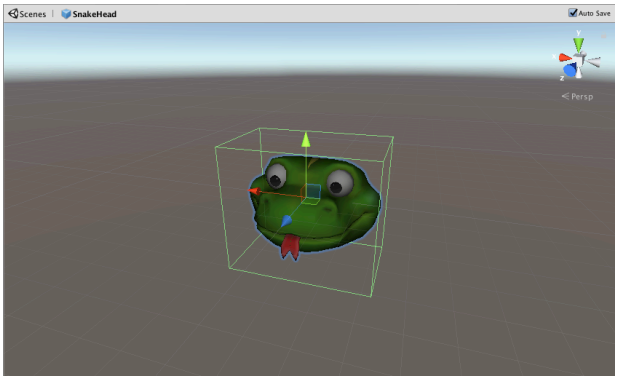


Figura 23 Modelo tridimensional de la serpiente

  Sus componentes son un PhotonView, que permite identificar la instancia durante la partida a través de la red. Además, consta de un PhotonTransformView. Este componente permite automatizar la sincronización de la rotación y posición de nuestra serpiente para que todos los jugadores tenga los mismos valores.

  El movimiento de la serpiente se controla a través de la clase SnakeTransform. Esta clase puede recibir la posición mínima y máxima de los ejes X y Z para limitar el área de movimiento. La posición objetivo se calcula de forma aleatoria y el movimiento de la serpiente es un MRU, movimiento rectilíneo uniforme, ya que su velocidad es constante.

  En el método Update(), que se ejecuta a cada *frame*, se lleva a cabo este movimiento.

```
private void Update()
{
    if (_photonView.IsMine)
    {
        float step =  speed * Time.deltaTime; // calculate distance to move
        transform.position = Vector3.MoveTowards(transform.position,
                                _targetPosition, step);
        transform.LookAt(_targetPosition);
        if (Vector3.Distance(transform.position, _targetPosition) < 0.001f)
```

[1] https://poly.google.com/

```
            {
                GetRandomPosition();
            }
        }
        else
        {
            SmoothNetMovement();
        }
    }
```

La función MoveTowards() mueve la serpiente desde la posición en la que se encuentra a una posición objetivo realizando un paso que viene dado por la velocidad por la diferencia de tiempo entre cada *frame*. El método LookAt() se utiliza para encarar el modelo tridimensional hacia la posición objetivo.

Cuando la distancia entre la posición actual y la posición objetivo es suficientemente pequeña, se calcula una posición aleatoria nueva sobre el tablero.

El método SmoothNetMovement() es ejecutado por los clientes con el objetivo de evitar posibles problemas de lentitud en la percepción del movimiento del personaje.

SnakeTransform implementa la interfaz IPunObservable que posee un método OnPhotonSerializeView() para sincronizar datos como la velocidad o la posición objetivo, los cuales se utilizan en el método SmoothNetMovement().

```
public void OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)
{
    if (stream.IsWriting)
    {
        stream.SendNext(transform.position);
        stream.SendNext(speed);
    }
    else
    {
        _targetPosition = (Vector3) stream.ReceiveNext();
        speed = (float) stream.ReceiveNext();
    }
}
```

En nuestra escena tenemos un GameObject, SnakeController, que se encarga de instanciar la serpiente cuando recibe el evento de inicio de juego, y de destruirla cuando se alcanza el estado final. Posee una variable que controla la energía de la serpiente que se mostrará a los usuarios a través del GameUIController. Cuando se produce un evento relacionado con un ítem gastronómico, se recibirá un aumento o disminución de la energía y al modificar esta variable, se modificará la variable de velocidad de la instancia de nuestra serpiente a través de la clase SnakeTransform.

La energía de la serpiente se actualiza mediante una Remote Procedure Call en todos los clientes para poder mostrar el nuevo valor en la interfaz de usuario.

- **Anchor:** Este Prefab corresponde al tablero de juego. Sobre el tablero de juego se mueve la serpiente. Es el objeto Cloud Anchor, que sirve de anclaje para representar un punto de origen para que todos los jugadores puedan observar los objetos de realidad aumentada desde su perspectiva de forma correcta. El modelo tridimensional de este Prefab es un plano blanco.

  Como se ha comentado anteriormente después de instanciar el tablero provisional, al pulsar el botón de crear ancla, se instancia este Prefab. Una vez instanciado, debe crearse el Cloud Anchor. Este tiene un identificador asociado que será enviado a todos los clientes para que puedan resolver el objeto. A continuación, se detalla parte del código de creación del Cloud Anchor en la clase AnchorController.

```
XPSession.CreateCloudAnchor(anchor).ThenAction(result =>
{
    if (result.Response != CloudServiceResponse.Success)
    {
            _mGameSceneController.OnAnchorHosted(false,
                     result.Response.ToString());
            return;
    }

    photonView.RPC("PunRpcSetCloudAnchorId", RpcTarget.AllBufferedViaServer,
    result.Anchor.CloudId);

  _mGameSceneController.OnAnchorHosted(true, result.Response.ToString());
});
```

  XPSession[1] es una clase que representa una sesión ARCore multiplataforma. Se pueden llevar a cabo las operaciones de crear un Cloud Anchor o resolverlo. Cuando se crea el ancla de nuestra habitación se envía el identificador a todos los demás jugadores y se notifica al cliente *master* que ha creado correctamente el tablero.

```
XPSession.ResolveCloudAnchor(cloudAnchorId).ThenAction(result =>
{
    if (result.Response != CloudServiceResponse.Success)
    {
        _mGameSceneController.OnAnchorResolved(false,
                     result.Response.ToString());
        _mShouldResolve = true;
        return;
    }

    _mGameSceneController.OnAnchorResolved(true, result.Response.ToString());
    _OnResolved(result.Anchor.transform);
});
```

[1] https://developers.google.com/ar/reference/unity/class/GoogleARCore/CrossPlatform/XPSession

Cuando los jugadores reciben el identificador pueden llevar a cabo la resolución del tablero. El método _OnResolved() utiliza la clase ARCoreWorldOriginHelper antes mencionada para establecer la rotación y posición de referencia de nuestro sistema mediante el método SetWorldOrigin(). Este método realizará cambios sobre la posición y rotación del GameObject de dispositivo ARCore de nuestra sesión para adaptarlo a la partida.

Cuando al menos dos jugadores hayan instanciado el tablero se podrá dar inicio a la partida.

6. Evaluación y resultados

Una vez desarrollado el sistema, se puede proceder a la evaluación de este con el objetivo de extraer resultados que deberán ser analizados.

Para realizar la evaluación se lleva a cabo un proceso experimental. Según Wohlin et al. [20], consta de una serie de fases, la definición de la que se extrae el objetivo, la planificación que lleva al diseño del experimento, la operación que obtiene unos datos que son analizados en el análisis e interpretación, el cual aporta unas conclusiones que pueden ser llevadas a una presentación y difusión.

6.1. Objetivo

Para definir el objetivo se puede utilizar la plantilla *goal question metric* (GQM) [21]. El objetivo es analizar un sistema multijugador gastrolúdico con el propósito de evaluar la experiencia de uso y viabilidad de este tipo de sistemas desde el punto de vista del usuario e investigador en el contexto de un trabajo de fin de máster.

6.2. Diseño del experimento y operación

En cuanto al diseño del experimento se deben definir diferentes aspectos de la prueba, que se detallan a continuación.

6.2.1. Participantes

En el experimento participaron un conjunto de 5 individuos en un rango de edad entre los 22 y los 25 años (Media (M) = 22.8, Desviación estándar (DE) = 1.166). Los individuos formaron grupos de parejas, formando 3 grupos.

6.2.2. Equipo

Este sistema se basa en cuatro tecnologías principales, Unity como motor de desarrollo de videojuegos, Photon Unity Networking (PUN), como plataforma de desarrollo de la infraestructura multijugador, la biblioteca de visión por computador OpenCV, y la biblioteca de realidad aumentada Google ARCore. Los dispositivos utilizados en el experimento son *smartphones* Android compatibles con el servicio de realidad aumentada de Google.

6.2.3. Procedimiento

Para cada grupo de parejas de jugadores, se realizó una partida en la que cada jugador pertenecía a un equipo diferente, uno pertenecía al equipo Kind y otro al equipo Wicked.

En una mesa se disponen los diferentes marcadores e ítems gastronómicos de diferentes tipos (ver Figura 23). Se han escogido dos tipos de comida diferentes: una saludable, fruta; y otra no saludable, una chocolatina.

Figura 24 Preparación del experimento

A cada jugador se le dan dos cintas que irán en el dedo índice y corazón del color que tienen asignado.

El cliente que crea la habitación escanea la mesa para colocar el tablero de juego. Una vez anclado y cuando el segundo jugador haya accedido a la habitación, podrá empezar la partida.

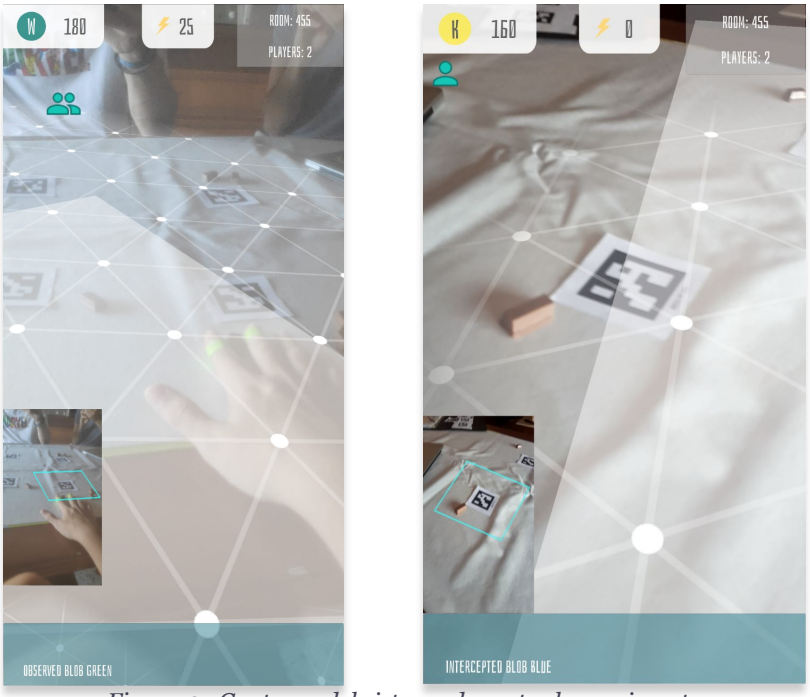


Figura 25 Capturas del sistema durante el experimento

Finalizada la partida, se pasa a cada jugador una encuesta para poder obtener sus opiniones acerca de la experiencia de juego durante el experimento. El cuestionario (ver Anexo 1) utilizado esta diseñado por IJsselsteijn et al. [22]. Se trata de un cuestionario de experiencia de juego que tiene una estructura dividida en tres módulos y respuestas a las cuestiones en base a una escala *Likert*. El primer módulo es la parte principal del cuestionario y trata la experiencia de juego desde siete componentes: Inmersión, flujo, competitividad, afecto positivo y negativo, tensión y desafío. La segunda parte del cuestionario está relacionada con el contexto social, bien por la interacción con otros jugadores que jueguen conjuntamente o en línea o bien con personajes del juego. La tercera y última parte evalúa cómo se sintieron los jugadores después de haber dejado de jugar.

6.3. Análisis e interpretación de resultados

El análisis de los resultados se puede realizar tratando los aspectos fundamentales de cada módulo del cuestionario. De este modo, cada sección corresponde a un módulo de la encuesta y en cada una se muestran representaciones gráficas de las respuestas. Para agrupar las preguntas por competencias o aspectos de la experiencia de juego en los diferentes módulos se ha utilizado la guía especificada en el último punto del documento de evaluación.

6.3.1. Módulo principal

El módulo principal trata siete componentes de experiencia de juego y cada uno corresponde a diferentes preguntas de este apartado.

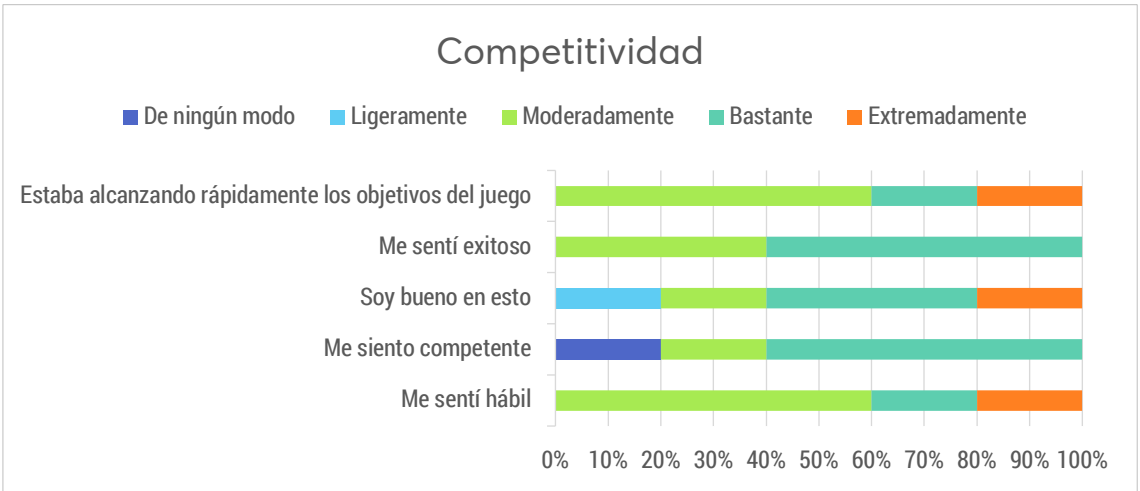


Gráfico 1 Módulo principal: Competitividad

La competitividad presenta tres preguntas que tienen una mayor puntuación, relacionadas con los objetivos del juego, el éxito y la habilidad. La única pregunta que ha recibido una peor puntuación ha sido "Me siento competente". En general, refleja que el juego les ha transmitido una sensación de competitividad o de nivel de habilidad.

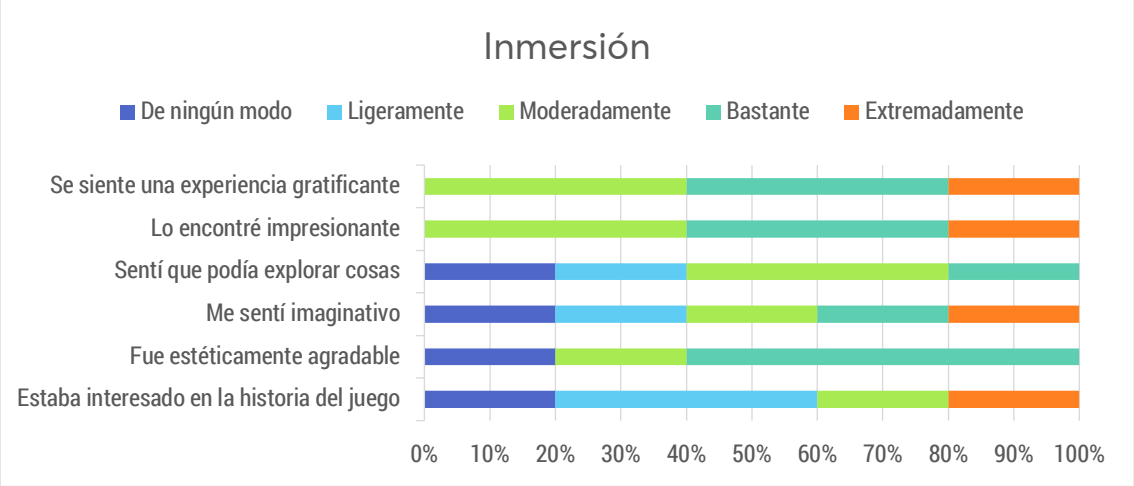


Gráfico 2 Módulo principal: Inmersión

La pregunta de "Estaba interesado en la historia del juego" es la pregunta que menor puntuación ha recibido, aunque esto puede ser debido a que este sistema no está ideado como un juego con una historia como elemento relevante. Las preguntas más importantes para analizar nuestro sistema pueden ser "Se siente una experiencia gratificante" o "Sentí que podía explorar cosas". Ambas preguntas reciben una valoración positiva.

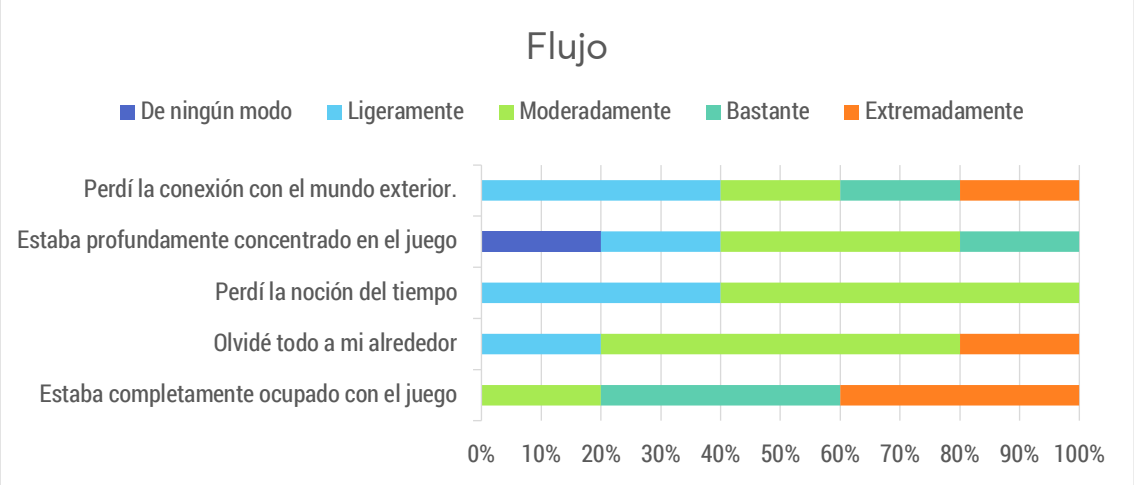


Gráfico 3 Módulo principal: Flujo

En cuanto al flujo, la cuestión que obtiene una mayor puntuación es "Estaba completamente ocupado con el juego" y la que obtiene una peor calificación es "Estaba profundamente concentrado en el juego". Esto parece contradictorio, aunque puede estar causado porque se sintieron ocupados mientras jugaban, pero no pudieron alcanzar un determinado grado de concentración debido a algún estorbo del contexto

de juego. Perder la noción del tiempo o olvidar que hay alrededor son elementos importantes de esta competencia.

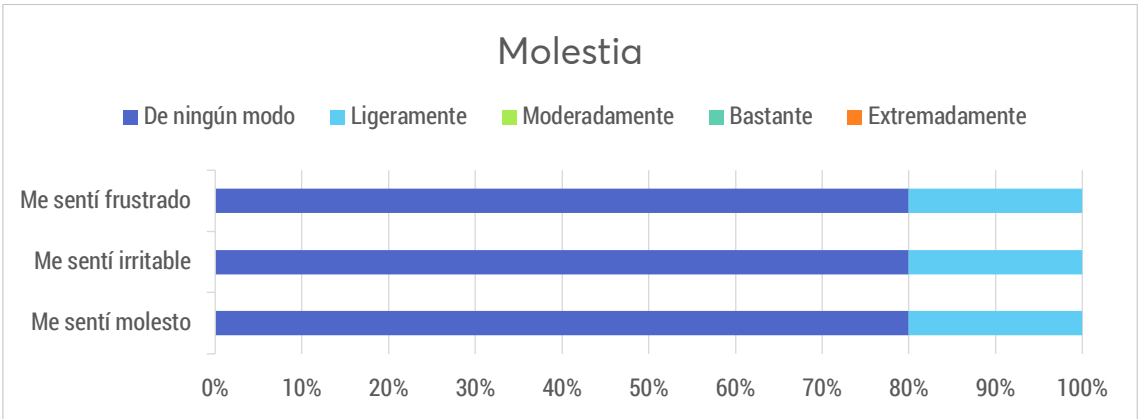


Gráfico 4 Módulo principal: Molestia

La información que transmiten las respuestas al apartado de molestia o enfado es que no provoca frustración, irritabilidad o molestia en el jugador.

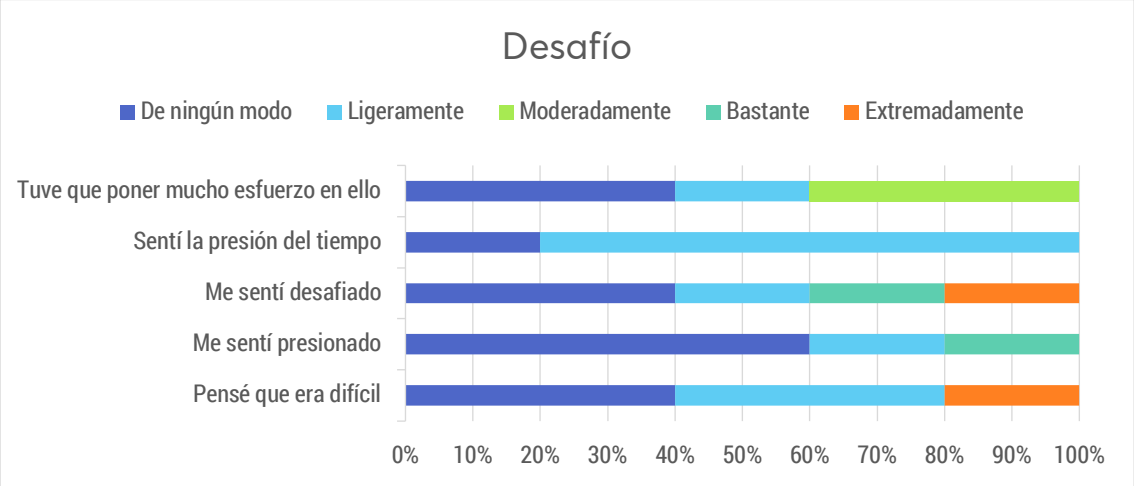


Gráfico 5 Módulo principal: Desafío

La cuestión "Pensé que era difícil" traslada la sensación de que no es un sistema difícil de utilizar. No se sintieron presionados ni demasiado desafiados. Una nota negativa sería que no sintieron la presión del tiempo, aunque no se trata de un elemento principal en el sistema por lo que sería común esta valoración. Por último, no tuvieron que poner mucho esfuerzo en el juego y esto podría corregirse más posibilidades de juego.

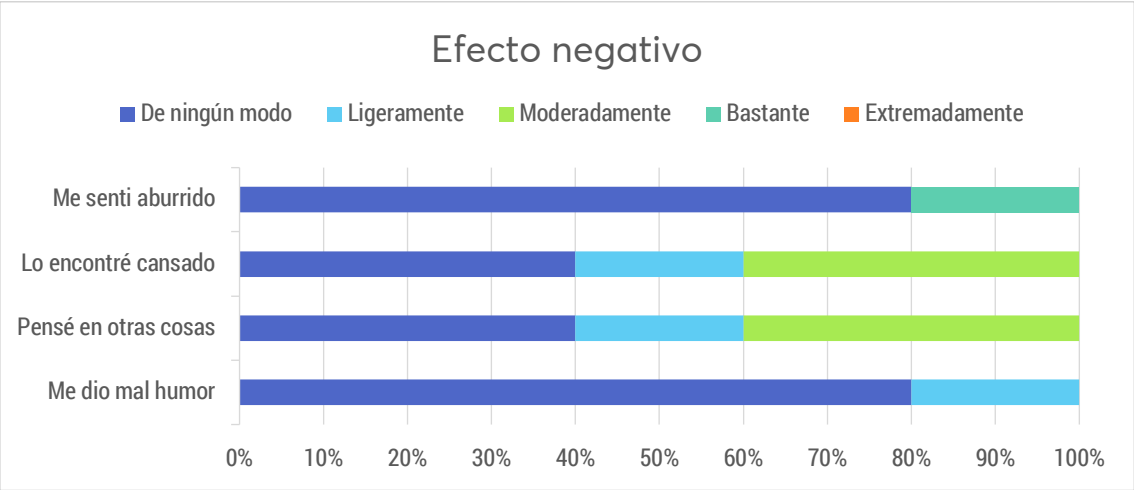


Gráfico 6 Módulo principal: Efecto negativo

El efecto negativo nos permite conocer si pensaron en otra cosa, lo pudieron encontrar cansado o les dio mal humor. De estas respuestas cabe destacar la respuesta "Bastante" al "Me sentí aburrido".

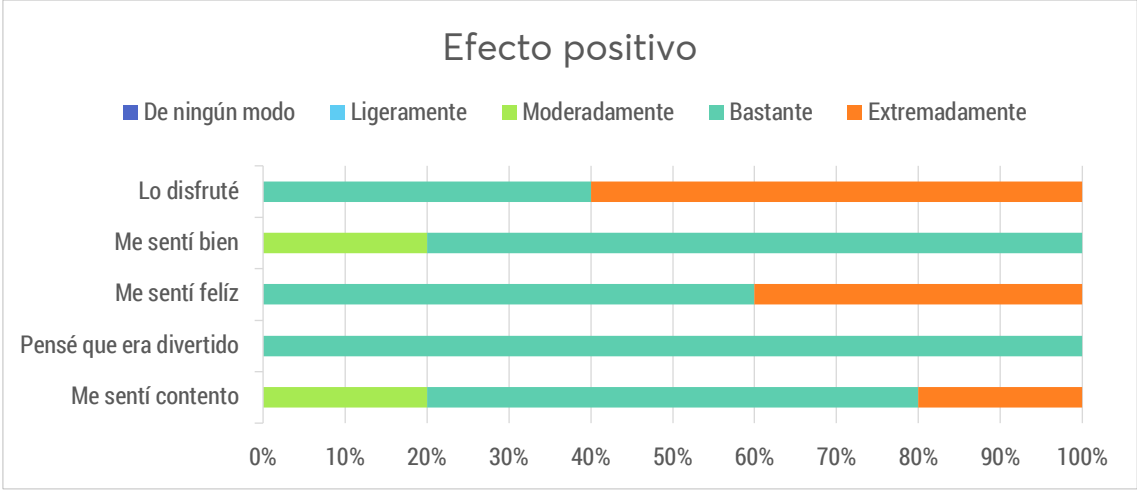


Gráfico 7 Módulo principal: Efecto positivo

El efecto positivo, en cambio, se refleja en si disfrutaron el juego, se sintieron bien o felices. En general, los jugadores valoraron de forma positiva estas cuestiones.

6.3.2. Módulo experiencia durante el juego

El módulo de experiencia durante el juego trata de explorar como las competencias tratadas en el módulo anterior están presentes durante la experiencia de juego.

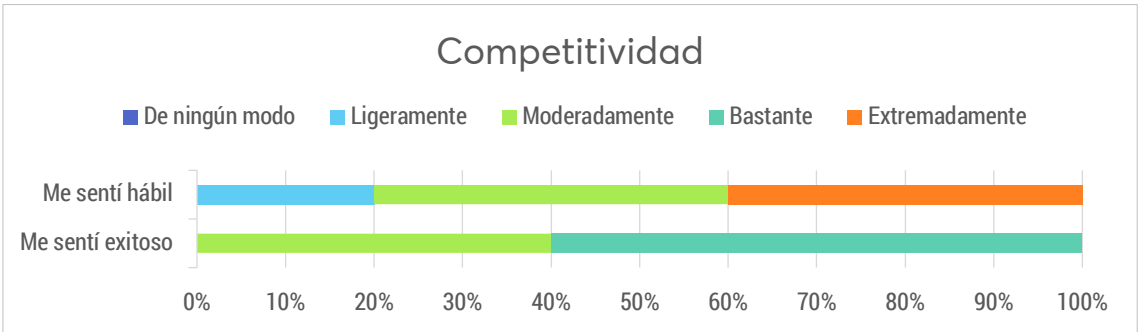


Gráfico 8 Módulo experiencia durante el juego: Competitividad

Durante el juego, la competitividad hace referencia a si se sintieron hábiles o exitosos. En este caso, ambos aspectos se valoran positivamente, lo cual da una visión de que pudieron sentir algún tipo de éxito o aprendizaje de una habilidad de juego.

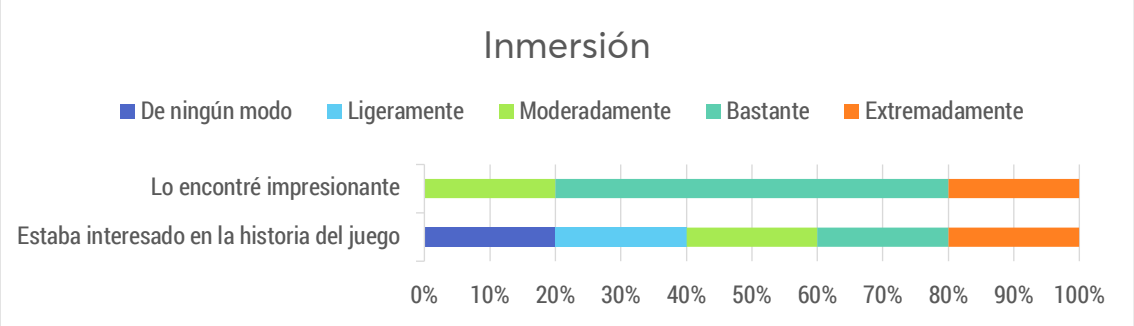


Gráfico 9 Módulo experiencia durante el juego: Inmersión

En cuanto a la inmersión durante el juego, la pregunta de "Estaba interesado en la historia del juego" tiene múltiples y distintas valoraciones que hacen pensar que la pregunta no sea adecuada para este sistema y no sea significativa.

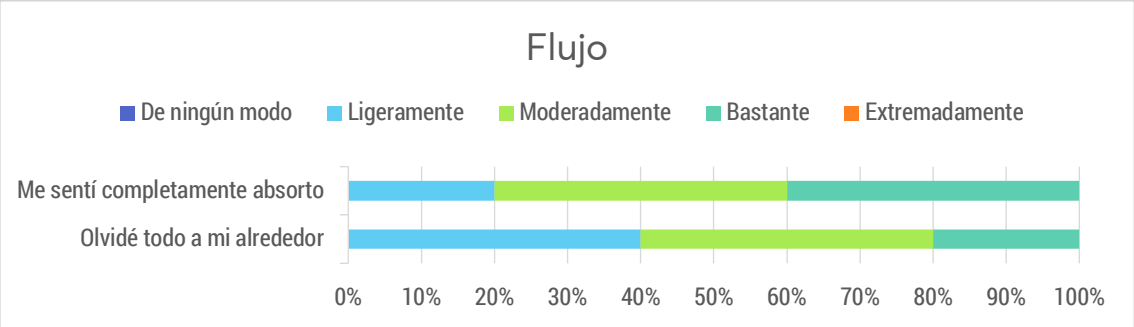


Gráfico 10 Módulo experiencia durante el juego: Flujo

El flujo es uno de los componentes principales y parece que detalles como sentirse completamente absorto u olvidar todo alrededor se dieron en algunos jugadores.

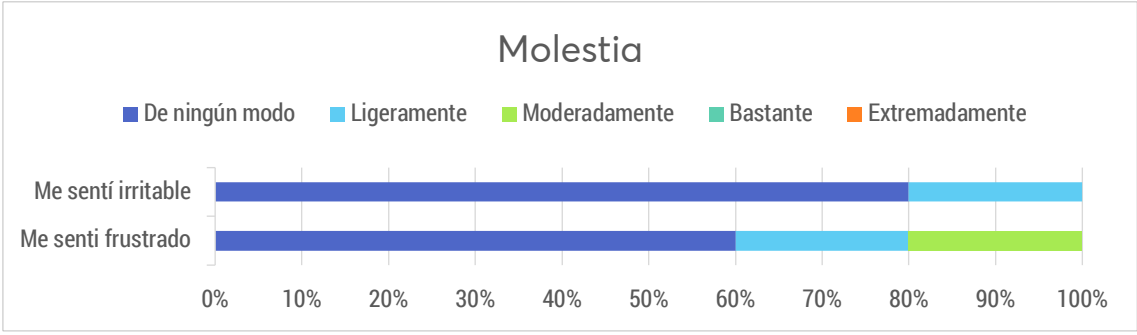


Gráfico 11 Módulo experiencia durante el juego: Molestia

De estas respuestas a la molestia sentida durante la experiencia de juego, el sentirse irritable o frustrado no se presentó de forma notable.

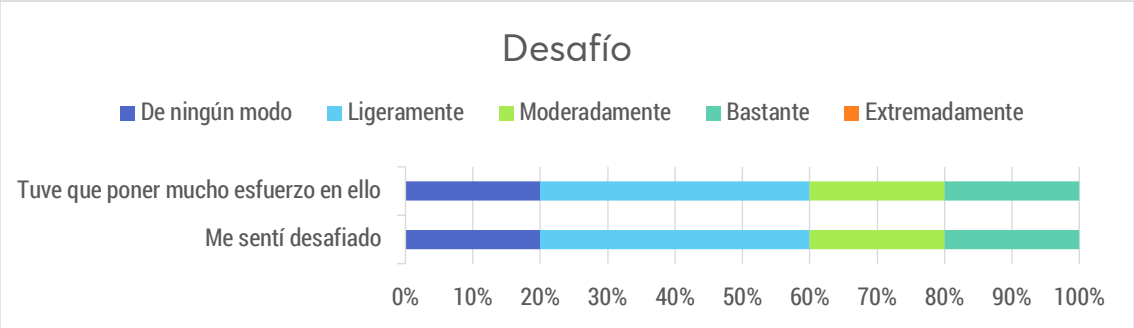


Gráfico 12 Módulo experiencia durante el juego: Desafío

Dadas estas respuestas parece ser que en algunos casos no se sintieron desafiados realmente o no tuvieron que poner mucho esfuerzo en ello. Como se ha comentado anteriormente, esto se podría solucionar introduciendo más reglas y elementos al juego.

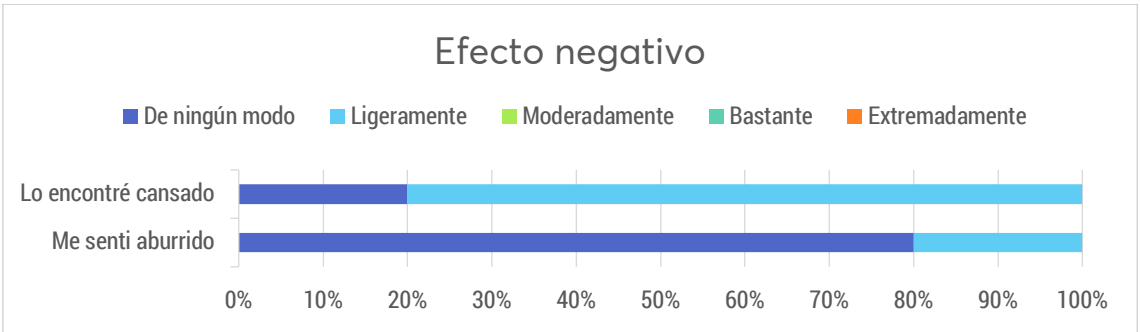


Gráfico 13 Módulo experiencia durante el juego: Efecto negativo

Se puede extraer una valoración positiva ya que en la mayoría de los casos no se sintieron aburridos o lo encontraron cansado.

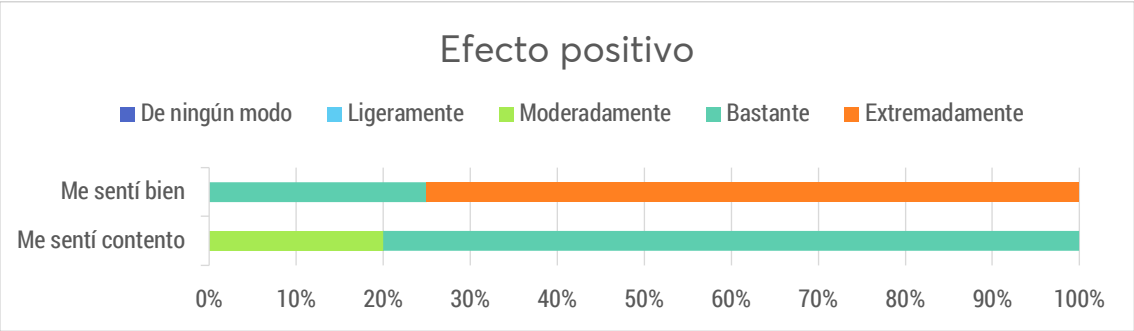


Gráfico 14 Módulo experiencia durante el juego: Efecto positivo

Además, los jugadores coinciden en que la mayoría de las veces, la experiencia durante el juego les hizo sentir bien o sentirse contentos.

6.3.3. Módulo de contexto social

Al tratarse de un sistema multijugador, se deben evaluar ciertos aspectos relacionados con el contexto social de juego. Uno de estos aspectos es la empatía. Las cuestiones ponen de relevancia que fue agradable estar con otros o que los jugadores transmitieron su felicidad. En cambio, se da menor valoración a que se sintieron conectados o que empatizaron con los otros. Esto puede deberse a una mayor necesidad de elementos multijugador para que sientan que están más conectados.

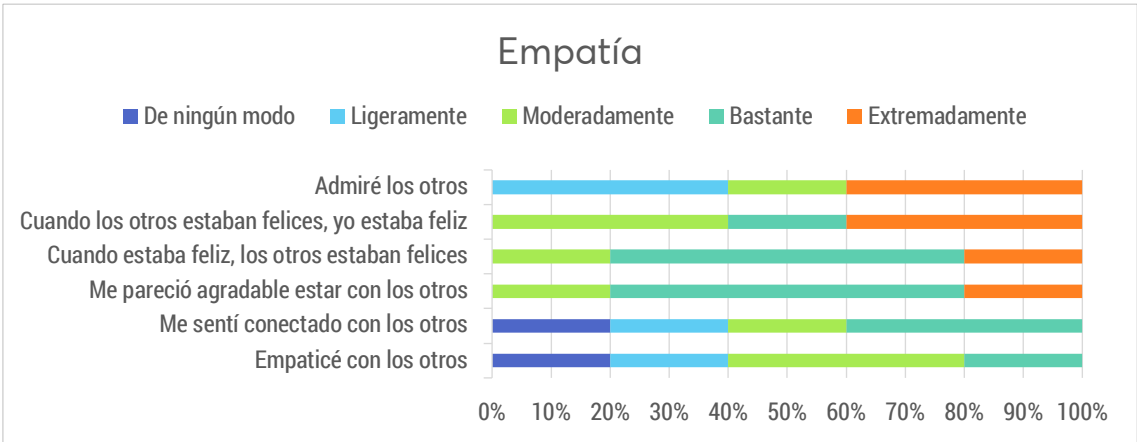


Gráfico 15 Módulo contexto social: Empatía

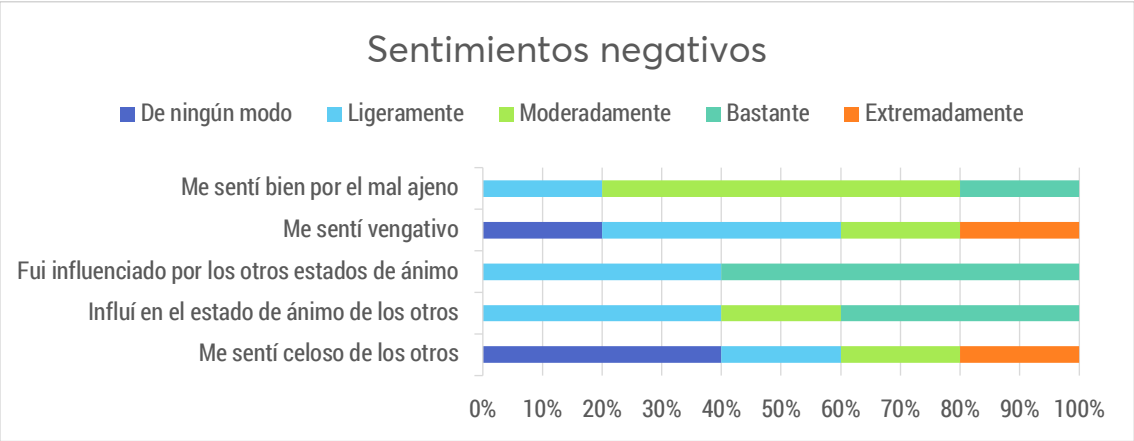


Gráfico 16 Módulo contexto social: Sentimientos negativos

En cuanto a los sentimientos negativos despertados hacía otros jugadores, se da menor importancia al sentimiento de venganza o de celos y se la da más a la influencia que tuvieron en el estado de ánimo de los otros o que ellos tuvieron en el jugador.

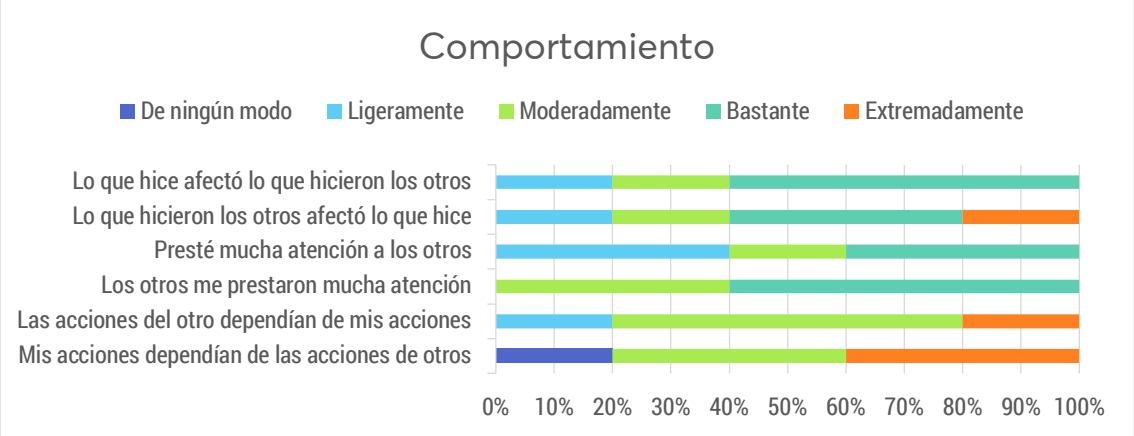


Gráfico 17 Módulo contexto social: Comportamiento

De este gráfico se puede extraer la conclusión de que las acciones o comportamientos de un jugador influyeron o afectaron a lo que hicieron otros y que, en general, se prestó atención a las acciones de los otros. La única pregunta que posee unas valoraciones más desiguales es la de "Mis acciones dependían de las acciones de otros" y esto puede tener una mejora si se introducen más componentes de interacción o que afecten al comportamiento de otros.

6.3.4. Módulo de sensación después del juego

En este módulo, se ponen de manifiesto aspectos que están relacionados con la sensación después del juego, es decir, aquello que sienten los jugadores una vez abandonan el juego.

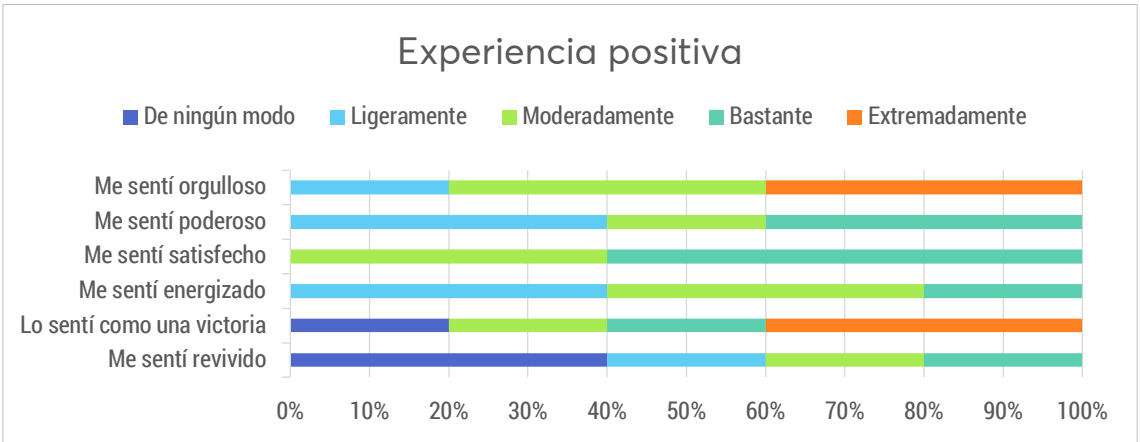


Gráfico 18 Módulo después de juego: Experiencia positiva

En general, la valoración que se realiza sobre la experiencia positiva destaca que sintieron satisfacción o que se sintieron orgullosos, lo cual es positivo. Aparecen algunas cuestiones cuyas respuestas son más sesgadas como el hecho de sentirlo como una victoria o sentirse revividos.

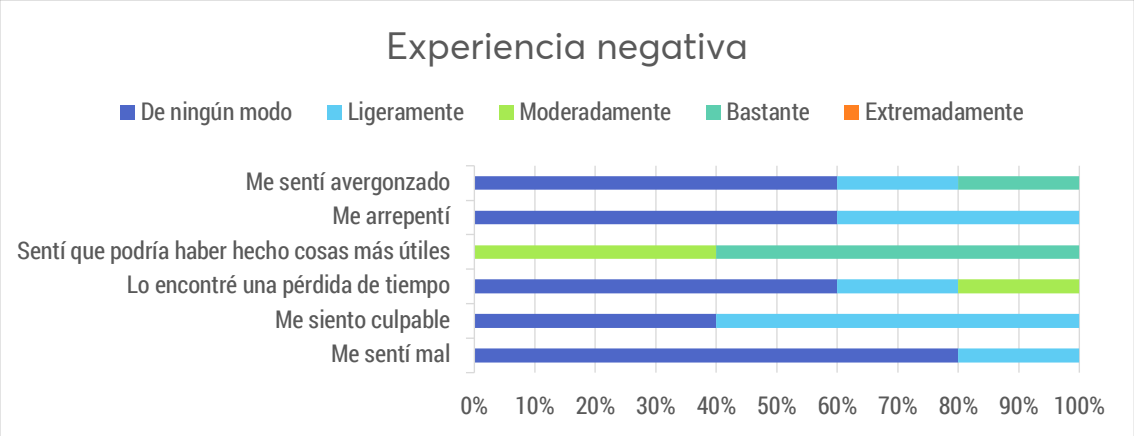


Gráfico 19 Módulo después de juego: Experiencia negativa

De esta gráfica debemos extraer que no se resaltaron demasiados aspectos negativos después del juego, aunque hay que centrarse en la cuestión de "Sentí que podría haber hecho cosas más útiles" ya que es bastante diferente a las demás y pone de relieve el hecho de que quizás en un futuro una persona no sintiese la voluntad de volver a jugar a un tipo de juego de estas características.

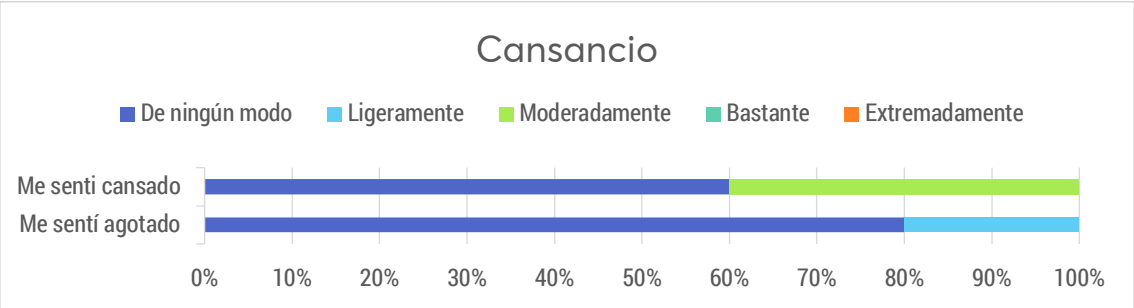


Gráfico 20 Módulo después de juego: Cansancio

De estas dos cuestiones se extrae que los jugadores no se sintieron cansados o agotados. La actividad, aunque implique movimiento por parte del jugador ya que tiene que conseguir elementos, se produce durante un periodo corto de tiempo y se reduce a estar sentado normalmente.

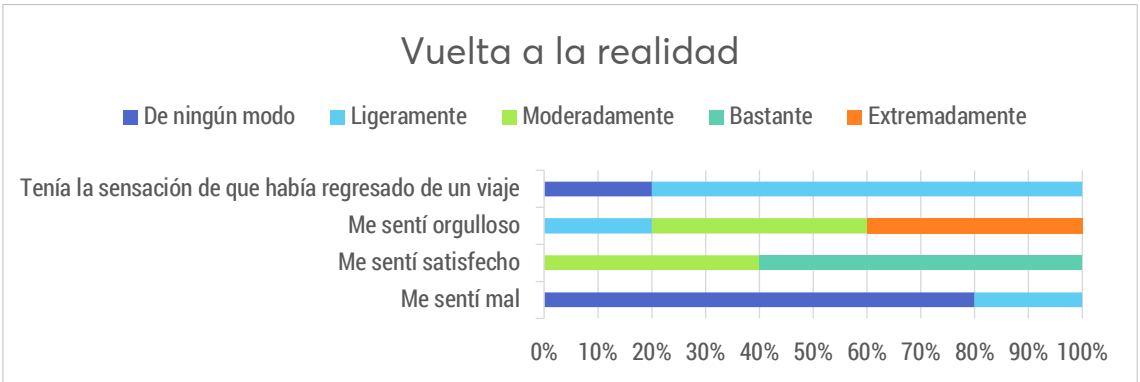


Gráfico 21 Módulo después de juego: Vuelta a la realidad

Por último, la vuelta a la realidad refleja como fue el cambio entre el momento de juego y el hecho de dejar de jugar. Los jugadores no indicaron que se sintieron mal y, en general, se sintieron satisfechos. La pregunta "Tenía la sensación de que había regresado de un viaje" sería más apropiada para experiencias de juego más duraderas o que impliquen una tecnología como la realidad virtual y no es tan relevante en este estudio.

6.4. Observaciones experimentales

Durante el experimento se percibió un detalle que debe ser tenido en cuenta. La biblioteca de visión por computador se utilizó entre otras cosas para identificar las manos de los jugadores que participaban en el experimento. Aunque el rango de área de detección de los elementos de colores que tienen los jugadores esté limitado, se daban casos de falsos positivos al detectar elementos de comida cuyo color estuviese en el rango de colores prefijado. Para poder solucionar este problema durante el experimento se optó por utilizar comida cuyo color no llevase a errores de detección.

7. Conclusiones y trabajo futuro

En este último capítulo se desarrollan las conclusiones a raíz del desarrollo y posterior evaluación del sistema. Se describen, además, posibles mejoras o alternativas sobre tecnologías utilizadas, así como nuevos elementos que podrían introducirse en futuras implementaciones y que constituyen el trabajo futuro a realizar sobre el proyecto.

7.1. Conclusiones

Este desarrollo ha constituido un acercamiento a un sistema que se encuentra dentro del marco del área de investigación de la gastroludología. Más específicamente, se ha llevado a cabo el desarrollo de un sistema gastrolúdico de experiencia mixta al utilizar tanto elementos reales como virtuales, de realidad aumentada.

Ha permitido comprobar la viabilidad del tipo de sistema al realizar una evaluación experimental y extraer resultados acerca de cuestiones planteadas a jugadores que probaron el sistema en un entorno de juego informal. De los gráficos extraídos y su análisis, se puede llegar a la conclusión que un sistema de este tipo puede ser viable y usable, en gran medida, pues hay ciertos aspectos derivados del análisis que deberían ser mejorados, como conseguir una mejor inmersión si existen más acciones que un usuario puede realizar.

Además, se han utilizado diferentes tipos de tecnologías como la realidad aumentada o la visión por computador que han permitido extraer conocimiento sobre si representan tecnologías de interacción adecuadas para este tipo de sistemas. La realidad aumentada se ha utilizado por la gran popularidad que ha adquirido últimamente en dispositivos móviles, y por la posibilidad de implementar un sistema en la que elementos reales sean visibles por el usuario y constituya un ejemplo de sistema de realidad mixto. Se trata de una tecnología apropiada para esta clase de sistemas, aunque utilizada en dispositivos móviles limita o puede generar molestia en el usuario tener que interactuar con elementos reales con una única mano y quizás sea más adecuado apostar por otros tipos de tecnologías o herramientas, a saber, realidad virtual o soportes para dispositivos móviles. La realidad virtual podría tener una buena acogida en este ámbito ya que soluciona el problema de la libertad de movimiento y favorece el incremento de la inmersión y flujo de los jugadores.

En cuanto a la visión por computador, el uso en diferentes aspectos del sistema como la detección de marcadores o la segmentación de imágenes da una visión sobre lo potente de esta tecnología y que es útil para otros desarrollos de sistemas de esta área

de investigación. Sin embargo, se han detectado limitaciones, para este sistema, como la detección de elementos de color que no deberían ser detectados y que podría solucionarse utilizando otro tipo de tecnología.

La principal conclusión que puede extraerse de este desarrollo y del análisis de resultados es que un sistema de estas características puede ser interesante pero aún necesita de una mejora en aspectos como la inmersión durante el juego, la exploración de otras tecnologías, la apuesta por otros sistemas que no sean obligatoriamente de juego y evaluaciones diseñadas a propósito para este ámbito. Se trata de un área de investigación en un estado prematuro que podrá crecer con la exploración y evaluación de nuevas soluciones.

7.2. Trabajo futuro

El desarrollo de este sistema ha constituido un primer acercamiento de un trabajo centrado en esta área de investigación. Existen diferentes aspectos y elementos que pueden ser modificados o añadidos en próximas implementaciones que mejoren este sistema.

La interacción con los elementos gastronómicos se realiza mediante una biblioteca de visión por computador. Esta biblioteca puede llegar a tener limitaciones debido a las condiciones de luz ya que se utiliza la segmentación por color para detectar objetos, y en este caso en particular, detectar las manos de los jugadores cuando cogen un elemento. Esto podría ser sustituido por una biblioteca de detección de manos o una biblioteca de aprendizaje automático que no dependa de que existan ciertas condiciones de luz para detectar una mano.

Otro cambio podría aplicarse a la biblioteca de marcadores aplicada. La principal ventaja de la biblioteca utilizada es que esta integrada en la plataforma de visión por computador empleada, y además permite identificar de forma sencilla diferentes tipos de marcadores. La principal desventaja es que no tolera la oclusión parcial del marcador, mientras que otras bibliotecas como RUNE-Tag o Vuforia sí que lo permiten. Resultaría interesante utilizar estas herramientas en un futuro cuando tengan una adaptación a Unity, que no existe en el caso de RUNE-Tag; o puedan coexistir con la biblioteca de realidad aumentada utilizada, cosa que Vuforia no acepta todavía.

Además, la biblioteca de realidad aumentada utilizada, ARCore, todavía está en una etapa muy temprana de desarrollo, y su característica de visión por computador necesita de una mejora muy considerable. En el momento en el que posea diversas características como segmentación de color o de marcadores, podría llevar a dejar de depender de una biblioteca de visión por computador a propósito.

Algunas mejoras sobre la jugabilidad incluyen utilizar objetos tridimensionales que interactúen también con el personaje de juego o introducir gestos con las manos de los jugadores que llevasen a cambios en el entorno de juego y en los eventos que están ocurriendo.

Dadas las observaciones experimentales comentadas, para solucionar el problema de la detección de falsos positivos cuando se detectan las manos de los jugadores, se podría optar por una biblioteca de aprendizaje automático o modificar la forma de detección mediante otra técnica de segmentación de imágenes.

8. Bibliografía

[1] Y. Chisik, P. Pons, and J. Jaen, "Gastronomy Meets Ludology," 2018.

[2] R. Comber, J. H. Choi, J. Hoonhout, and K. O'Hara, "Designing for human–food interaction: An introduction to the special issue on 'food and interaction design,'" *Int. J. Hum. Comput. Stud.*, vol. 72, no. 2, pp. 181–184, Feb. 2014.

[3] N. Ranasinghe and E. Y.-L. Do, "Digital Lollipop," *ACM Trans. Multimed. Comput. Commun. Appl.*, 2016.

[4] P. Arnold, R. A. Khot, and F. "Floyd" Mueller, "'You Better Eat to Survive': Exploring Cooperative Eating in Virtual Reality Games," *Proc. Twelfth Int. Conf. Tangible, Embed. Embodied Interact. - TEI '18*, 2018.

[5] R. Khot, F. Mueller, and D. Young, "Human-Food Interaction," *Found. Trends® Human-Computer Interact.*, vol. 12, pp. 238–415, 2019.

[6] D. Schmalstieg and T. Hollerer, "Augmented reality: Principles and practice," in *Proceedings - IEEE Virtual Reality*, 2017.

[7] T. P. Caudell and D. W. Mizell, "Augmented reality: an application of heads-up display technology to manual manufacturing processes," in *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*, 1992.

[8] R. T. Azuma, "A survey of augmented reality," *Presence: Teleoperators and Virtual Environments*. 1997.

[9] C. S. C. Dalim, T. Piumsomboon, A. Dey, M. Billinghurst, and S. Sunar, "TeachAR: An Interactive Augmented Reality Tool for Teaching Basic English to Non-native Children," in *Adjunct Proceedings of the 2016 IEEE International Symposium on Mixed and Augmented Reality, ISMAR-Adjunct 2016*, 2017.

[10] R. Perla, G. Gupta, R. Hebbalaguppe, and E. Hassan, "InspectAR: An Augmented Reality Inspection Framework for Industry," in *Adjunct Proceedings of the 2016 IEEE International Symposium on Mixed and Augmented Reality, ISMAR-Adjunct 2016*, 2017.

[11] W. Si, X. Liao, Q. Wang, and P. A. Heng, "Augmented Reality-Based Personalized Virtual Operative Anatomy for Neurosurgical Guidance and Training," in *25th IEEE Conference on Virtual Reality and 3D User Interfaces, VR 2018 - Proceedings*, 2018.

[12] M. A. Sánchez-Acevedo, B. A. Sabino-Moxo, and J. A. Márquez-Domínguez, "Mobile Augmented Reality," in *Virtual and Augmented Reality*, 2018.

[13] L. Abdi, F. Ben Abdallah, and A. Meddeb, "In-Vehicle Augmented Reality Traffic Information System: A New Type of Communication between Driver and Vehicle," in *Procedia Computer Science*, 2015.

[14] T. S. Huang, "Computer Vision: Evolution and Promise," *Report*, 1997.

[15] A. Dadhich, *Practical Computer Vision*. Packt Publishing, 2018.

[16] P. H. Mirvis and M. Csikszentmihalyi, "Flow: The Psychology of Optimal Experience," *Acad. Manag. Rev.*, 1991.

[17] F. Romero Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Fractal

Markers: a new approach for long-range marker pose estimation under occlusion." 2019.

[18] A. Sagitov, K. Shabalina, L. Sabirova, H. Li, and E. Magid, "ARTag, AprilTag and CALTag fiducial marker systems: Comparison in a presence of partial marker occlusion and rotation," in *ICINCO 2017 - Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotics*, 2017.

[19] F. Bergamasco, A. Albarelli, E. Rodolà, and A. Torsello, "RUNE-Tag: A high accuracy fiducial marker with strong occlusion resilience," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2011.

[20] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. 2012.

[21] R. van Solingen, V. Basili, G. Caldiera, and H. D. Rombach, "Goal Question Metric (GQM) Approach," in *Encyclopedia of Software Engineering*, 2002.

[22] W. a. IJsselsteijn, Y. a. W. de Kort, and K. Poels, "Game Experience Questionnaire," *FUGA - Fun of Gaming*. 2013.

Anexo 1: Encuesta de experiencia de juego

Please indicate how you felt while playing the game for each of the items, on the following scale:

not at all	slightly	moderately	fairly	extremely
0	1	2	3	4

2. Game Experience Questionnaire – Core Module

1 I felt content
2 I felt skilful
3 I was interested in the game's story
4 I thought it was fun
5 I was fully occupied with the game
6 I felt happy
7 It gave me a bad mood
8 I thought about other things
9 I found it tiresome
10 I felt competent
11 I thought it was hard
12 It was aesthetically pleasing
13 I forgot everything around me
14 I felt good
15 Iwasgoodatit
16 I felt bored
17 I felt successful
18 I felt imaginative
19 I felt that I could explore things
20 I enjoyed it
21 I was fast at reaching the game's targets
22 I felt annoyed
23 I felt pressured
24 I felt irritable
25 I lost track of time
26 I felt challenged
27 I found it impressive
28 I was deeply concentrated in the game
29 I felt frustrated
30 It felt like a rich experience
31 I lost connection with the outside world

32 I felt time pressure

33 I had to put a lot of effort in to it

3. In-game GEQ

1 I was interested in the game's story

2 I felt successful

3 I felt bored

4 I found it impressive

5 I forgot everything around me

6 I felt frustrated

7 I found it tiresome

8 I felt irritable

9 I felt skilful

10 I felt completely absorbed

11 I felt content

12 I felt challenged

13 I had to put a lot of effort in to it

14 I felt good

4. GEQ - Social Presence Module

1 I empathized with the other(s)

2 My actions depended on the other(s) actions

3 The other's actions were dependent on my actions

4 I felt connected to the other(s)

5 The other(s) paid close attention to me

6 I paid close attention to the other(s)

7 I felt jealous about the other(s)

8 I found it enjoyable to be with the other(s)

9 When I was happy, the other(s) was(were) happy

10 When the other(s) was(were) happy, I was happy

11 I influenced the mood of the other(s)

12 I was influenced by the other(s) moods

13 I admired the other(s)

14 What the other(s) did affected what I did

15 What I did affected what the other(s) did

16 I felt revengeful

17 I felt schadenfreude (malicious delight)

5. GEQ – post-game module

1 I felt revived

2 I felt bad

3 I found it hard to get back to reality

4 I felt guilty

5 It felt like a victory

6 I found it a waste of time

7 I felt energised

8 I felt satisfied

9 I felt disoriented

10 I felt exhausted

11 I felt that I could have done more useful things

12 I felt powerful

13 I felt weary

14 I felt regret

15 I felt ashamed

16 I felt proud

17 I had a sense that I had returned from a journey

6. Scoring guidelines

Scoring guidelines GEQ Core Module

The Core GEQ Module consists of seven components; the items for each are listed below. Component scores are computed as the average value of its items.

Competence: Items 2, 10, 15, 17, and 21.

Sensory and Imaginative Immersion: Items 3, 12, 18, 19, 27, and 30.

Flow: Items 5, 13, 25, 28, and 31.

Tension/Annoyance: Items 22, 24, and 29.

Challenge: Items 11, 23, 26, 32, and 33.

Negative affect: Items 7, 8, 9, and 16.

Positive affect: Items 1, 4, 6, 14, and 20.

Scoring guidelines GEQ In-Game version

The In-game Module consists of seven components, identical to the core Module. However, only two items are used for every component. The items for each are listed below.

Component scores are computed as the average value of its items.

Competence: Items 2 and 9.

Sensory and Imaginative Immersion: Items 1 and 4.

Flow: Items 5 and 10.

Tension: Items 6 and 8.

Challenge: Items 12 and 13.

Negative affect: Items 3 and 7.

Positive affect: Items 11 and 14.

Scoring guidelines GEQ Social Presence Module

The Social Presence Module consists of three components; the items for each are listed below. Component scores are computed as the average value of its items.

Psychological Involvement – Empathy: Items 1, 4, 8, 9, 10, and 13.

Psychological Involvement – Negative Feelings: Items 7, 11, 12, 16, and 17.

Behavioural Involvement: Items 2, 3, 5, 6, 14, and 15.

Scoring guidelines GEQ Post-game Module

The post-game Module consists of four components; the items for each are listed below. Component scores are computed as the average value of its items.

Positive Experience: Items 1, 5, 7, 8, 12, 16.

Negative experience: Items 2, 4, 6, 11, 14, 15.

Tiredness: Items 10, 13.

Returning to Reality: Items 3, 9, and 17.