V Congreso de la Sociedad Española para las Ciencias del Videojuego (V CoSECiVi)

SESIÓN 2

Generación Efectiva de Controladores Difusos Evolutivos para Carreras en Simuladores de Coches

Antonio M. Mora
Depto. de Teoría de la Señal, Telemática y Comunicaciones
ETSIIT-CITIC, Universidad de Granada, España
amorag@ugr.es

Mohammed Salem
Department of Computer Sciences
University of Mascara, Algeria
salem@univ-mascara.dz

Juan J. Merelo
Depto. de Arquitectura y Tecnología de Computadores
ETSIIT-CITIC, Universidad de Granada, España
jmerelo@ugr.es

Pablo García-Sánchez
Departamento de Informática
Universidad de Cádiz, España
pablo.garciasanchez@uca.es

***Resumen*—Los simuladores de carreras de coches han sido utilizados durante mucho tiempo como un entorno para probar algoritmos de control autónomo de vehículos. Constituyen un entorno en el para evaluar todo tipo de algoritmos, incluyendo metaheurísticas, como por ejemplo Algoritmos Evolutivos. Sin embargo, el mayor desafío en este tipo de algoritmos evolutivos es diseñar un proceso de evaluación fiable y eficaz para los individuos, que se traducirá en la generación de buenas soluciones para el problema abordado: encontrar un controlador que sea capaz de ganar en una amplia gama de pistas y con distinto número de oponentes de diversa dificultad. Dicha evaluación no sólo implica el diseño de una función de aptitud (fitness) adecuada, que represente el nivel de calidad de los controladores de automóviles, sino también la selección de la mejor solución para la problema de optimización que se está resolviendo. Este paso puede resultar muy complejo, debido a la incertidumbre/ruido presente en el problema (condiciones meteorológicas y de la pista, comportamiento impredecible de otros conductores, etc). Por tanto, este trabajo presenta una serie de propuestas para realizar la evaluación de controladores difusos optimizados, analizando el rendimiento o calidad de los mismos en comparación con propuestas anteriores. Esto facilitará el diseño automático de un controlador autónomo para el simulador de carreras de coches TORCS. Para ello se parte de los resultados preliminares obtenidos en trabajos anteriores y se rediseña parte del procedimiento de evaluación del fitness. De modo que se estudian dos funciones de fitness diferentes en varios experimentos, junto con un novedoso enfoque basado en la competición para la selección del mejor individuo en cada generación.**

***Index Terms*—simuladores de carreras de coches,TORCS, controladores difusos, conductores autónomos, algoritmos genéticos, optimización, evaluación**

I. Introducción

La conducción autónoma es un problema que surge en muchos entornos, como los aviones, barcos, trenes o vehículos submarinos no tripulados. Recientemente están empezando a aparecer los primeros coches autónomos, en los que no es necesario un conductor humano. Todos estos sistemas se basan, en general, en una serie de entradas de sensores que incluyen velocidad real, obstáculos y otros vehículos. Considerando esas entradas, el controlador tendrá que tomar una decisión sobre la velocidad y dirección óptimas [7]. La prueba de diferentes metodologías de conducción autónoma en el mundo real suele estar reservada a unas pocas empresas importantes, por lo que las pruebas de algoritmos se realizan normalmente en entornos simulados, los cuales ofrecen, a su vez, el incentivo de poder comparar mediante competiciones distintas propuestas.

En este artículo, utilizaremos *The Open Racing Car Simulator (TORCS)* [19], un simulador de carreras muy realista que proporciona un gran entorno de pruebas para la implementación y evaluación de conductores/controladores autónomos. TORCS ha sido utilizado en diversas ocasiones para la celebración de competiciones de Inteligencia Artificial (IA), donde el objetivo es crear los mejores pilotos autónomos de carreras [10]–[12]. Además de evaluar nuestros controladores enfrentándolos contra otros, el simulador puede ser utilizado como un entorno aislado para optimizar la conducción en una carrera en solitario.

Los Algoritmos Evolutivos (AEs) [1] han sido aplicados frecuentemente como un método de optimización de propósito general en este área, los cuales son habitualmente combinados con motores de comportamiento que gobiernan diferentes partes del coche [6], [15], [16] e incluso con aproximaciones basadas en Deep Learning [8]. Dichos motores de comportamiento han sido modelados recientemente mediante controladores difusos [4], [9], [14], los cuales aplican Lógica Difusa (*fuzzy logic* o FL en inglés) [2]. Dicha técnica es bastante adecuada para definir este tipo de agentes autónomos, ya que está inspirada en razonamientos que los humanos aplicamos al conducir. Un controlador difuso funciona con variables lingüísticas, y por ejemplo, girará *ligeramente* a la derecha cuando la siguiente curva esté cerca; si bien estos controladores tienen que ser diseñados para asociar apropiadamente las entradas a las salidas deseadas en situaciones específicas.

Los autores de este trabajo presentamos anteriormente una propuesta que combinaba dos controladores difusos especializados, diseñados a mano, que eran capaces de decidir el ángulo de giro adecuado del coche y la velocidad deseada en cada

punto (o en cada instante de juego, *tick*) durante una carrera [17]. Dicha propuesta fue revisada más tarde [18], optimizando los parámetros de sus funciones de pertenencia por medio de un Algoritmo Genético (AG) [3], mejorando el rendimiento del diseño manual, así como el de otros controladores de la literatura con los que se comparó.

Esto probó que los algoritmos evolutivos eran capaces de definir un conjunto de parámetros para el controlador difuso mucho mejor que los de un diseño hecho a mano, pero al mismo tiempo desveló varios desafíos. En general, los AEs optimizan la función de fitness que se utiliza, de modo que los controladores difusos evolucionados (en lo sucesivo, "FCs", o *Fuzzy Controllers*) serán finalmente tan buenos como lo permita dicha función. El problema es que en este caso no se puede considerar como función de aptitud la posición alcanzada por el FC en todas las carreras posibles en todas las pistas posibles y contra todos los oponentes posibles, así que tenemos que conformarnos con un *sustituto* de la función de fitness en un entorno muy limitado.

De modo que para mejorar el proceso de evaluación primero se optó por eliminar los oponentes, haciendo pruebas en carreras en solitario. Además se estudió qué factores relacionados con la velocidad, el daño y el tiempo por vuelta se deberían tener en cuenta para la evaluación de cada controlador (fueron incluidos en la función de fitness inicial).

Los resultados obtenidos en aquellos trabajos fueron alentadores, pero el modelo propuesto era algo ïmpreciso", ya que había que decidir cuál es la mejor pista para realizar el entrenamiento, así como los factores de carrera en solitario con mayor impacto en el rendimiento en una carrera real. En este sentido, en el presente estudio se ha elegido de forma analítica una pista concreta y representativa para los entrenamientos, la cual tiene una longitud media y combina tramos rectos con zonas de muchas curvas. A su vez se han combinado en la función de fitness, en dos aproximaciones distintas, los términos relacionados con la velocidad durante la carrera (a maximizar) y el daño recibido (a minimizar), que consideramos como los más relevantes. Además, se ha tenido en cuenta que el proceso de evaluación del fitness tiene un cierto grado de incertidumbre, como el hecho de que los daños y algunas condiciones de la pista pueden variar aleatoriamente en función del tipo de circuito. Por ello, en lugar de seleccionar directamente el mejor controlador en base a su valor de fitness como hicimos en los trabajos anteriores, en esta propuesta vamos a llevar a cabo una carrera entre los mejores individuos (controladores) para seleccionar al ganador.

El objetivo de estos tres factores (elección de pista, función de fitness y selección de ganador) es crear un algoritmo para crear controladores que sea más robusto y eficiente.

II. El simulador y los controladores

En esta sección se presentan el entorno de investigación considerado (el simulador de carreas de coches), y se describen los controladores definidos por los autores.

II-A. The Open Racing Car Simulator

TORCS [19] es un simulador de carreras de coches de código abierto, realista, multijugador y modular que permite a los usuarios competir contra otros oponentes controlados por ordenador. Es un entorno de pruebas bastante fiable y muy utilizado en investigación en inteligencia artificial.

Cada coche en TORCS manejará un conjunto de sensores y valores del entorno, por ejemplo distancias a bordes de la pista o a otros vehículos rivales, el combustible restante, la marcha actual, la posición en la carrera, la velocidad, o los daños, entre otros. Estos valores serán considerados por los conductores autónomos o *controladores*, para gestionar el coche utilizando los llamados *actuadores*: giro del volante, acelerador, freno y cambios de marcha.

II-B. Subcontroladores Difusos

El controlador diseñado por los autores se basa en el modelo de sensores y actuadores de la *Simulated Car Racing Competition* [12].

Sin embargo, la velocidad objetivo y el ángulo de giro de la dirección se calculan mediante dos sistemas modulares y especializados [17]. Estos subcontroladores incorporaron lógica difusa y consideran cinco sensores de posición. Partiendo de ellos, se aplicaron AGs para mejorarlos de manera automática.

El *subcontrolador difuso de velocidad objetivo* pretende estimar la velocidad objetivo óptima del coche, tanto en las partes rectas, como en las curvas de la pista. Para ello tiene en cuenta dos criterios: moverse lo más rápido posible y de la manera más segura (con el menor daño posible). Esta estimación se basa en dos casos generales: si el coche encara una línea recta, la velocidad objetivo tomará un valor máximo (*maxSpeed* km/h). Sin embargo, si está cerca de una curva, se disminuirá la velocidad actual a un valor incluido en el intervalo *[minSpeed, maxSpeed]* km/h.

Este controlador difuso tiene una salida, la velocidad, y tres valores de entrada (ver Figura 1):

- `Front` = Sensor_9: Distancia frontal al borde de la pista (ángulo 0°).
- `M5` = max (Sensor_8, Sensor_10): distancia máxima al borde de la pista con un ángulo de +5°y -5°con respecto a `Front`.
- `M10` = max (Sensor_7, Sensor_11): distancia máxima al borde de la pista con un ángulo de +10°y -10°.

Se trata de un sistema difuso de tipo Mamdani [5] con tres funciones de pertenencia (MF) trapezoidales para cada variable de entrada. En [18] se optimizaron con un AG los conjuntos de parámetros que definen las funciones de pertenencia, mejorando en gran medida los resultados obtenidos.

Además, el controlador está basado en un conjunto de reglas difusas, diseñadas para maximizar la velocidad del coche dependiendo de las distancias detectadas al borde de la pista. Dichas reglas pueden verse en [17].

El segundo es el *subcontrolador difuso para el giro del volante*, que pretende determinar el mejor ángulo de giro en base a una estimación de la posición objetivo del coche. Su

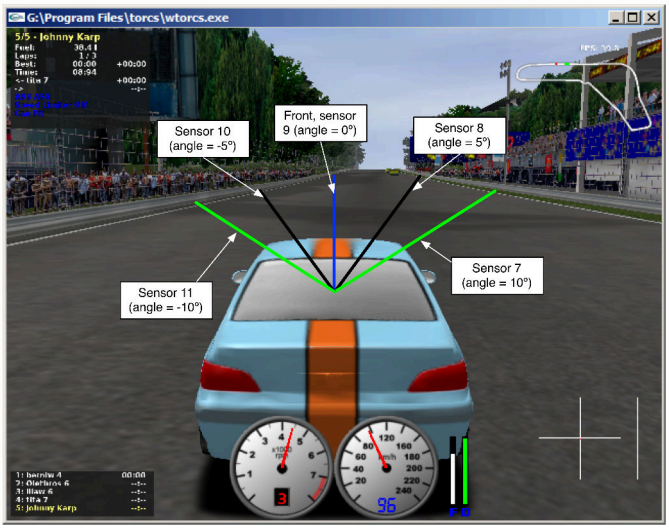


Figura 1. Sensores considerados de los 18 que tiene asociados cada coche en TORCS

estructura es similar a la del controlador anterior, basándose en los mismos sensores, pero considerando el giro como salida del mismo.

De modo que, como reglas generales: si el coche circula en línea recta, se fijará como posición objetivo el centro del carril por el que circula; mientras que, si el coche está cerca de una curva a derecha o izquierda, se acercará a la curva dejando un espacio entre el coche y el borde de la pista para evitar la pérdida de control.

Para detectar las curvas, el controlador revisa los valores de los sensores (`M10`, `M5` y `Front`), de modo que si el valor en el sensor frontal es el mayor, hay un tramo recto; mientras que si los valores de `M5` y `M10` con ángulos positivos (+5 y +10) son los mayores, habrá una curva a la derecha, y viceversa.

El controlador usa un conjunto de reglas que fue definido modelando el comportamiento de un conductor humano [17].

III. Algoritmo Genético

El algoritmo de optimización propuesto tiene como objetivo encontrar los parámetros óptimos de las funciones de pertenencia de los dos subcontroladores previamente introducidos.

El AG comienza creando una población inicial con valores aleatorios (distribución uniforme) para los parámetros en el rango definido $[0, 100]$. La idoneidad de cada solución candidata se calcula inyectando sus valores genéticos a los parámetros de las funciones de membresía de los dos subcontroladores difusos. El controlador autónomo definido se utiliza para conducir un coche en una carrera de 20 vueltas en un circuito sin oponentes en TORCS, y los resultados (velocidad máxima, mínima y media, junto con el daño obtenido) se utilizan para calcular el valor del fitness correspondiente.

Los controladores difusos tienen funciones de pertenencia trapezoidales, que siguen la Ecuación 1. En un controlador de este tipo, las reglas difusas se aplican a términos lingüísicos, que califican las llamadas variables lingüísticas y que se definen mediante funciones de pertenencia que dependen de un conjunto de parámetros que determinan su forma y su 'funcionamento'. De modo que se aplicó un AG para optimizar dichos parámetros y determinar la partición difusa de la variable lingüística [20]. Las variables lingüísticas de entrada en nuestro problema serán `Front`, `M5` y `M10`.

Una función de pertenencia (MF) trapezoidal, se define como:

$$\mu_A(x) = \begin{cases} \frac{x-x_1}{x_2-x_1}, & x_1 \le x \le x_2 \\ 1, & x_2 \le x \le x_3 \\ \frac{x_4-x}{x_4-x_3}, & x_3 \le x \le x_4 \\ 0, & else \end{cases} \quad (1)$$

with:

$$x_1 \le x_2 \le x_3 \le x_4 \quad (2)$$

Como se puede ver, una MF está determinada por cuatro parámetros x_1, x_2, x_3 y x_4, los cuales tienen valores en el intervalo *[a, b]* (Figura 2).

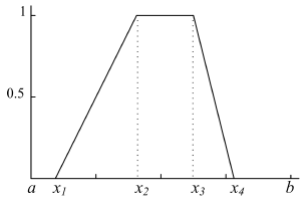


Figura 2. Función de pertenencia trapezoidal

De modo que una partición difusa con *n* MF trapezoidales se define mediante *2n* variables ($a = x_1, x_2, .., x_{2n} = b$) (Ecuación 4). Con:

$$a = x_1 \le x_2 \le ... \le x_{2n-1} \le x_{2n} = b \quad (3)$$

$$\mu_{A1}(x) = \begin{cases} 1, & x_1 \le x \le x_2 \\ \frac{x_3-x}{x_3-x_2}, & x_2 \le x \le x_3 \\ 0, & x > x_3 \end{cases}$$

$$\mu_{Ai}(x) = \begin{cases} 0, & x \le x_{2i-2} \\ \frac{x-x_{2i-2}}{x_{2i-1}-x_{2i-2}}, & x_{2i-2} \le x \le x_{2i-1}, n = 2, ..., i-1 \\ 1, & x_{2i-1} \le x \le x_{2i} \\ \frac{x_{2i+1}-x}{x_{2i+1}-x_{2i}}, & x_{2i} \le x \le x_{2i+1} \\ 0, & x > x_{2i+1} \end{cases}$$

$$\mu_{An}(x) = \begin{cases} 0, & x \le x_{2n-2} \\ \frac{x-x_{2n-2}}{x_{2n-1}-x_{2n-2}}, & x_{2n-2} \le x \le x_{2n-1} \\ 1, & x > x_{2n-1} \end{cases} \quad (4)$$

Cuando el número de parámetros es reducido y sus rangos de variación están bien definidos, un AG con codificación binaria será suficiente para encontrar sus valores óptimos. Sin embargo, dado que en nuestro problema las salidas deseadas requieren precisión y el intervalo de variación de cada

parámetro no esá bien determinado, hemos considerado una codificación real de los parámetros, que hemos dispuesto en un vector que incluye todas las variables a optimizar. De modo que cada individuo de nuestro AG será un vector con esa estructura, que tendrá 18 parámetros, 6 por variable.

La *inicialización de la población* de cromosomas/individuos se realiza asignando valores aleatorios en un rango de variación predefinido [3], a fin de comenzar la optimización desde un conjunto de valores prometedores [17].

Para llevar a cabo la evolución se han considerado: la selección de padres basada en *torneo*, un operador de *cruce aritmético simple en dos puntos* [21] y un operador de *mutación no uniforme* [13], por ser operadores genéticos bien contrastados en la literatura.

El objetivo principal del controlador autónomo en este entorno es ganar tantas carreras como sea posible. Sin embargo, tenemos que optimizar el caso más general mediante la realización de *carreras de entrenamiento en solitario*, que serán menos sensibles a la presencia de ruido/incertidumbre debido a la participación de otros controladores.

En propuestas anteriores nos enfocamos en la minimización del daño recibido ($damage$) y el tiempo de vuelta $LapTime$, a la par que intentábamos maximizar la velocidad máxima alcanzada $TopSpeed$. Sin embargo, en este estudio, nos hemos centrado en un .[en]foque más humano", es decir, tratar de conducir lo más rápido posible en cada una de las partes de la pista evitando los daños. Por lo tanto, hemos considerado:

- Velocidad mínima ($MinSpeed$): para mejorar la conducción en zonas difíciles del circuito, como las zonas de curvas.
- Velocidad Máxima ($MaxSpeed$): para optimizar la conducción en las zonas rectas o sencillas de la pista.
- Velocidad Media ($AVGSpeed$): que modelará el comportamiento general en la pista.
- Daño ($Damage$): con el objetivo de crear controladores seguros, que sean capaces de terminar la carrera en cualquier circunstancia.

De modo que se han combinado estos factores en dos posibles funciones de evaluación/fitness:

GFC-MMS:

$$f_1 = \frac{(MinSpeed * MaxSpeed)}{Damage+1} \quad (5)$$

GFC-AVS:

$$f_2 = \frac{AVGSpeed}{Damage+1} \quad (6)$$

Como se puede observar, en la primera función el objetivo es maximizar las velocidades mínima y máxima, a la par que se minimiza el daño recibido. En la segunda, se intenta maximizar la velocidad media del controlador en el circuito completo.

La evaluación de cada solución candidata (individuo) durante la evolución, haremos que cada uno de ellos compita en una carrera de entrenamiento de 20 vueltas en un circuito de dificultad media sin rivales. Como se ha dicho, hemos omitido la presencia de oponentes para evitar incluir fuentes de incertidumbre adicionales al proceso de optimización. Para obtener controladores de comportamiento general, la pista seleccionada para este proceso tendrá una combinación de zonas de muchas curvas (difíciles) y partes rectas (sencillas).

Una vez que dicha carrera de prueba ha concluido, se tomarán los valores de salida: $Damage$, $MinSpeed$, $MaxSpeed$ y $AVGSpeed$ y se calculará el valor correspondiente del fitness aplicando la fórmula deseada.

IV. Experimentos y Resultados

Después de analizar la mayoría de las pistas disponibles, hemos seleccionado para estos experimentos el circuito **Alpine 2**. Éste es bastante complejo, con múltiples tipos de curvas, aunque también con ciertas partes rectas (Ver Figura 3).

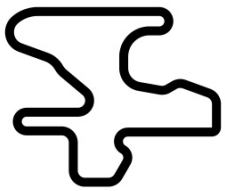


Figura 3. Circuito Alpine 2: Pista lenta de montaña. Longitud: 3773,57m, Anchura: 10m

Como coche para nuestro controlador, hemos utilizado *car1-tbr1*, ya que según experimentos anteriores [17], es la mejor opción debido a su rendimiento moderado y buen control, lo que lo hará adecuado para la gran mayoría de las pistas.

El controlador genético difuso (GFC) ha sido evaluado considerando las dos funciones de fitness propuestas: GFC-MMS (Ecuación 5) y GFC-AVS (Ecuación 6), comparando su rendimiento en carrera. Hemos ejecutado el algoritmo con una población de 50 individuos. El resto de parámetros son: Generaciones=50, Tasa de cruce=0,85, Tasa de mutación=0,1, y 10 ejecuciones diferentes por cada configuración.

Los dos procesos de optimización genética (uno por cada función de fitness) se han llevado a cabo de forma independiente. Sin embargo, a diferencia de trabajos anteriores en los que se seleccionaba el mejor en base a su valor para dicha función, en este estudio hemos buscado implementar una mejor metodología, la cual esperamos que produzca un controlador más competitivo.

Para ello, en ambas implementaciones, una vez finalizado el proceso evolutivo, los cuatro mejores individuos han competido juntos en 5 carreras (de 5 vueltas cada una) en la pista **Alpine 2** (utilizada durante la optimización) y 5 carreras (de 5 vueltas) en la pista **E-Track 5** (nueva para ellos).

Además, con el fin de mejorar la selección de los mejores, otros dos controladores son elegidos al azar para participar en la carrera, de entre una selección de bots de TORCS. Hemos implementado una competición basada puntos, que se basa en el esquema de la Fórmula 1, por lo que las puntuaciones obtenidas dependen de la posición del coche: 1 - 25 puntos, 2 - 18, 3 - 15, 4 - 12, 5 - 10, 6 - 8, 7 - 6, 8 - 4, 9 - 2, 10 - 1. Además, para incluir los términos de fitness en esta

selección, hemos definido una *puntuación extra*, de forma que el controlador que consiga el mejor tiempo o el daño mínimo en cada vuelta recibe 5 puntos extra.

Los resultados de estas ejecuciones se muestran en la Tabla I. Las "Mejores vueltasz "Daño mínimo"son las puntuaciones obtenidas por cada controlador en cada carrera al conseguir el mejor tiempo de vuelta y/o el daño mínimo de todos los contendientes. El símbolo '-' significa que el bot de TORCS no participa en la carrera.

Según la tabla, el primer individuo de $GFC-MMS$ y el segundo de $GFC-AVS$ han ganado el mismo número de carreras, pero $GFC-AVS_2$ ha logrado mejores resultados en las carreras que no ganó. Hay que destacar que los resultados de los bots de TORCS no se han tenido en cuenta ya que sólo sirven para diversificar la selección y no participan en todas las carreras. Esta selección permite, por tanto, elegir el mejor individuo en varias carreras y de forma más robusta y estable, y así se evita la selección clásica por torneo donde se elige el ganador de una sola confrontación.

También se puede señalar que los controladores difusos genéticos obtienen el daño mínimo, incluso cuando no ganan la carrera. Este hecho justifica fuertemente el uso de daño en las funciones de fitness, que es un factor clave a tener en cuenta en las carreras reales (para terminarlas).

Los controladores obtenidos con la primera función de fitness también han sumado los puntos de las mejores vueltas en cinco de las diez carreras. Hay que tener en cuenta que la mejor vuelta es el resultado de un daño mínimo y una alta velocidad ($MaxSpeed$), ambos optimizados por dicha función de evaluación. En la misma línea, el segundo fitness intenta maximizar la velocidad media, pero no necesariamente $MaxSpeed$.

Para probar la efectividad del método, hemos elegido los dos mejores controladores difusos genéticos, $GFC-MMS_1$ y $GFC-AVS_2$, uno por función de fitness, obtenidos en las pruebas anteriores. Éstos han sido evaluados en un conjunto de carreras contra algunos oponentes seleccionados. Además, los dos mejores controladores evolutivos de nuestro anterior trabajo [18], $EVO1$ y $EVO2$, también han sido incluidos en la 'competición'.

Esta evaluación es un tipo de mini-campeonato, que también considera las puntuaciones de Fórmula 1, pero en este caso no hay puntos extra. Se realizaron 10 carreras, cada una de 20 vueltas, y con un total de 10 participantes por carrera: los dos mejores $GFC-MMS_1$ y $GFC-AVG_2$, $EVO1$, $EVO2$, y también 6 bots competitivos de TORCS escogidos de la literatura. Las primeras 5 carreras se realizaron en la pista **Alpine 2** (usada en el entrenamiento); y las otras 5 carreras tuvieron lugar en la pista **E-Track 5** (no utilizada por los nuevos controladores, pero usada en la evolución de los anteriores). La parrilla de salida (posiciones iniciales de los coches) en estas carreras se estableció al azar.

Los resultados se muestran en la Tabla II. Esta tabla muestra cómo uno de los controladores difusos que evolucionan usando los nuevos mecanismos de selección y evaluación, $GFC-MMS_2$, da los mejores resultados, obteniendo muy buenas clasificaciones en las carreras. El controlador Inferno1 también obtuvo muy buenos resultados, alcanzando 3 veces la mejor puntuación por vuelta, pero esos resultados están muy influenciados debido a que usó el coche más rápido. También podemos ver que nuestros dos controladores difusos genéticos sólo han ganado una carrera $GFC-MMS_1$ y dos carreras $GFC-AVS_2$, mientras que los bots $berniw2$ y $inferno1$ han ganado tres carreras cada uno.

Sin embargo, como se puede ver en la tabla, aunque no son capaces de ganar siempre, sí que han terminado en posiciones altas de la clasificación, lo que les ha ayudado a obtener puntos y ganar el campeonato finalmente. De modo que la combinación entre la minimización de daño y la búsqueda de la velocidad media ha sido el mejor enfoque.

V. Conclusiones y Trabajo Futuro

En este trabajo hemos presentado métodos para mejorar la generación efectiva de controladores para el simulador de coches TORCS mediante el uso de algoritmos evolutivos. Se parte de un controlador que hace uso de lógica difusa para calcular la velocidad objetivo (sub-controlador 1) y la dirección del coche (sub-controlador 2).

De modo que se han propuesto dos funciones de fitness más especializadas que las que usamos en trabajos anteriores [17], [18], enfocadas en la minimización de los daños obtenidos durante la carrera, así como la maximización de la velocidad máxima (en tramos sencillos), mínima (en tramos complejos) y media (en toda la pista).

Junto a esto, se ha postulado la selección heurística de pistas para el entrenamiento/evolución de los controladores, que contenga diversidad en sus tramos para obtener conductores autónomos más adaptables.

Además se ha propuesto un mecanismo de selección de los mejores controladores, una vez concluida la evolución, basándose en la realización de carreras y eligiendo al ganador en ellas, en lugar de ceñirnos al valor obtenido en la función de fitness. De este modo, se elegirá a los mejores de forma más justa y robusta.

Los experimentos realizados nos dejan clara la efectividad de las propuestas, puesto que se han enfrentado los controladores obtenidos contra rivales de nivel medio/alto, incluyendo los de trabajos anteriores de los autores, en un campeonato en el que han resultado ganadores los nuevos.

Como trabajo futuro, nos centraremos primeramente en la evaluación de los distintos enfoques aplicados, para identificar aquel que tiene mayor influencia en los resultados. Respecto al algoritmo genético empleado, se podría intentar implementar un enfoque multiobjetivo, que considere los distintos factores de la función de fitness como independientes.

Agradecimientos

Este trabajo ha sido financiado en parte por el Ministerio Español de Economía y Competitividad con los proyectos TIN2014-56494-C4-3-P (UGR-EPHEMECH), TIN2017-85727-C4-2-P (UGR-DeepBio) y TEC2015-68752 (también financiado por FEDER).

Tabla I
PUNTUACIONES OBTENIDAS EN LA SELECCIÓN BASADA EN CARRERAS PARA LAS DOS IMPLEMENTACIONES EN DOS PISTAS DIFERENTES

Controlador	5 carreras en la pista Alpine 2 (5 vueltas cada una)							5 carreras en la pista E-Track 5 (5 vueltas cada una)							
	C1	C2	C3	C4	C5	Mejores vueltas	Daño mínimo	C1	C2	C3	C4	C5	Mejores vueltas	Daño mínimo	Total
$GFC-MMS_1$	**25**	18	8	15	12	15	10	12	**25**	18	**25**	18	10	10	**221**
$GFC-MMS_2$	12	**25**	15	12	15	0	5	8	15	15	4	10	0	0	136
$GFC-MMS_3$	6	6	10	10	8	0	0	15	10	10	18	6	0	5	104
$GFC-MMS_4$	2	8	4	4	6	0	0	10	1	2	1	2	0	0	40
$GFC-AVS_1$	1	4	6	2	4	0	0	4	2	12	10	4	0	0	49
$GFC-AVS_2$	15	10	18	**25**	18	5	10	**25**	18	**25**	15	15	5	10	206
$GFC-AVS_3$	10	2	1	6	2	0	0	2	6	4	6	1	0	0	41
$GFC-AVS_4$	4	1	2	1	1	0	0	1	4	8	2	8	0	0	32
$bt1$	-	-	-	8	-	0	0	-	8	6	8	-	0	0	-
$inferno1$	18	-	12	-	-	0	0	18	12	-	-	-	0	0	-
$berniw2$	8	15	-	18	-	0	0	-	-	-	12	12	5	0	-
$berniw3$	-	12	-	-	**25**	5	0	4	-	-	-	**25**	5	0	-
$damned1$	-	-	**25**	-	10	0	0	-	-	8	-	-	0	0	-

Tabla II
RESULTADOS DEL MINI-CAMPEONATO CON 10 CONTROLADORES Y 10 CARRERAS EN DOS PISTAS DISTINTAS. `tita`, `berniw` E `inferno` SON CONTROLADORES INCLUIDOS CON EL SIMULADOR TORCS [19]

Controlador	Carreras en **Alpine 2** (20 vueltas cada una)						Carreras en **E-Track 5** (20 vueltas cada una)						Puntuación Total
	C1	C2	C3	C4	C5	Puntuación por pista	C6	C7	C8	C9	C10	Puntuación por pista	
$GFC-MMS_1$	25	10	18	12	10	75	18	12	15	15	12	72	147
$GFC-AVS_2$	15	18	25	15	15	**88**	25	18	18	12	18	91	**179**
$tita1$	4	2	1	2	2	11	4	2	1	4	6	17	28
$tita2$	2	1	2	1	1	7	1	1	2	1	2	9	16
$inferno1$	12	15	12	18	18	75	12	15	25	25	15	**92**	167
$inferno2$	10	12	4	10	25	61	10	10	4	2	8	34	95
$berniw1$	18	25	15	8	6	72	8	8	6	10	10	42	114
$berniw2$	8	8	10	25	12	63	15	25	10	8	25	83	146
$EVO1$	6	6	8	4	8	32	2	6	12	8	4	32	64
$EVO2$	1	4	6	6	4	21	6	4	8	6	2	26	47

REFERENCIAS

[1] T. Bäck, *Evolutionary algorithms in theory and practice*. Oxford University Press, 1996.

[2] S. Godil, M. Shamim, S. Enam, and U. Qidwai, "Fuzzy logic: A 'simple' solution for complexities in neurosciences?" *Surg Neurol Int.*, pp. 2 – 24, 2011. [Online]. Available: https://doi.org/10.4103/2152-7806.77177

[3] D. E. Goldberg, *Genetic Algorithms in search, optimization and machine learning*. Addison Wesley, 1989.

[4] S. Guadarrama and R. Vazquez, "Tuning a fuzzy racing car by coevolution," in *Genetic and Evolving Systems, GEFS 2008*, March 2008. [Online]. Available: https://doi.org/10.1109/GEFS.2008.4484568

[5] I. Iancu, *A Mamdani Type Fuzzy Logic Controller*. InTech, 2012, pp. 325–352.

[6] T. S. Kim, J. C. Na, and K. J. Kim, "Optimization of an autonomous car controller using a self-adaptive evolutionary strategy," *International Journal of Advanced Robotic Systems*, vol. 9, no. 3, p. 73, 2012. [Online]. Available: https://doi.org/10.5772/50848

[7] S. Kolski, D. Ferguson, C. Stacniss, and R. Siegwart, "Autonomous driving in dynamic environments," in *In Proceedings of the Workshop on Safe Navigation in Open and Dynamic Environments at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006.

[8] J. Koutník, J. Schmidhuber, and F. Gomez, "Evolving deep unsupervised convolutional networks for vision-based reinforcement learning," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '14, 2014, pp. 541–548.

[9] D. P. Liébana, G. Recio, Y. Sáez, and P. Isasi, "Evolving a fuzzy controller for a car racing competition," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Games, CIG 2009, Milano, Italy, 7-10 September, 2009*, 2009, pp. 263–270. [Online]. Available: https://doi.org/10.1109/CIG.2009.5286467

[10] D. Loiacono, P.-L. Lanzi, J. Togelius, E. Onieva, D. A. Pelta, M. . Butz, T. D. Lonneker, L. Cardamone, D. Perez, Y. Saez, M. Preuss, and J. Quadflieg, "The 2009 simulated car racing championship," *IEEE Trans. Comput. Intell. AI Games*, vol. 2(2), pp. 131–147, 2010.

[11] D. Loiacono, J. Togelius, P. L. Lanzi, L. Kinnaird-Heether, S. M. Lucas, M. Simmerson, D. Perez, R. G. Reynolds, and Y. Saez, "The wcci 2008 simulated car racing competition," in *2008 IEEE Symposium On Computational Intelligence and Games*, Dec 2008, pp. 119–126.

[12] D. Loiacono, L. Cardamone, and P. L. Lanzi, "Simulated car racing championship: Competition software manual," *CoRR*, vol. abs/1304.1672, 2013. [Online]. Available: http://arxiv.org/abs/1304.1672

[13] A. Neubauer, "A theoretical analysis of the non-uniform mutation operator for the modified genetic algorithm," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, 1997. [Online]. Available: https://doi.org/10.1109/ICEC.1997.592275

[14] E. Onieva, J. Alonso, J. Pérez, and V. Milanés, "Autonomous car fuzzy control modeled by iterative genetic algorithms," in *Fuzzy Systems*, 2009, pp. 1615 – 1620.

[15] E. Onieva, D. Pelta, J. Godoy, V. Milanés, and J. Rastelli, "An evolutionary tuned driving system for virtual car racing games: The autopia driver," *International Journal of Intelligent Systems*, vol. 27, pp. 217–241, 2012.

[16] E. Onieva, D. A. Pelta, J. Alonso, V. Milanés, and J. Pérez, "A modular parametric architecture for the torcs racing engine," in *Proceedings of the 5th IEEE Symposium on Computational Intelligence and Games (CIG'09)*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 256–262.

[17] M. Salem, A. M. Mora, J. J. Merelo, and P. García-Sánchez, "Driving in TORCS using modular fuzzy controllers," in *Applications of Evolutionary Computation. EvoApplications 2017, Lecture Notes in Computer Science, vol 10199*, S. K. Squillero G., Ed. Springer, Cham, 2017, pp. 361–376.

[18] ——, "Evolving a TORCS modular fuzzy driver using genetic algorithms," in *Applications of Evolutionary Computation. EvoApplications 2018, LNCS*, K. S. et al., Ed. Springer, 2018, p. To appear.

[19] Sourceforge, "Web torcs," Web, Nov. 2016, http://torcs.sourceforge.net/.

[20] H. D. Thang and J. M. Garibaldi, "A novel fuzzy inferencing methodology for simulated car racing," in *IEEE International Conference on Fuzzy Systems, Hong Kong, China, 1-6 June, 2008, Proceedings*. IEEE, 2008, pp. 1907–1914. [Online]. Available: https://doi.org/10.1109/FUZZY.2008.4630630

[21] S. G. Varun Kumar and R. Panneerselvam, "A study of crossover operators for genetic algorithms to solve VRP and its variants and new sinusoidal motion crossover operator," *International Journal of Computational Intelligence Research*, vol. 13 (7), pp. 1717–1733, 2017.

A Machine Learning Approach to Predict the Winner in StarCraft based on Influence Maps*

*Summary of the paper with the same title published in Entertainment Computing 19 (2017) 29-41

Antonio A. Sánchez-Ruiz
Facultad de Informática
Universidad Complutense de Madrid
Madrid, Spain
antsanch@ucm.es

Maximiliano Miranda
Facultad de Informática
Universidad Complutense de Madrid
Madrid, Spain
m.miranda@ucm.es

***Abstract*—Real-Time Strategy games are very popular test beds for Artificial Intelligence (AI) researchers. In this work we try to predict the winner of StarCraft©[1] games based on *influence maps*. Influence maps are numerical matrices representing the influence of each player's army in the map, and they are useful for different types of spatial reasoning. Our system reaches a level of precision similar to the human judges, although human judges base their predictions on a much more complex and abstract set of game features.**

***Index Terms*—Real-Time Strategy Games, Influence Maps, Machine Learning, Prediction, StarCraft.**

I. Introduction

Real-Time Strategy (RTS) games are very popular test beds for Artificial Intelligence (AI) researchers because they provide complex and controlled environments on which to test different AI techniques [7]. In this work we try to predict the winner as soon as possible based on the events that occur in the game. This role of professional observer arises naturally in most traditional sports like football or even eSports where different experts analyze what is happening during the game and explain their opinions regarding the strategies selected by the players or teams.

We based our predictions on influence maps (IM), a spatial and tactical representation of the game state, and some standard machine learning classifiers. IM were initially used in the context of the game of Go [9] and have been more recently used in RTS games for pathfinding and reactive decision making [2], [4], [5], [8].

In order to test our approach, we use two different datasets of StarCraft games. The first one contains 4-player games in which all the players were controlled by the internal StarCraft AI. The second dataset contains real 2-player games from specialized websites. We have studied the precision of the predictions in different moments of the game, the number of games required to train the learning algorithms and the stability of their predictions over time. Finally, we performed a small experiment to compare our results with the predictions made by some expert players and conclude that our results are comparable, but human experts base their predictions on a much more complex and abstract set of game features.

This paper is a brief summary of our previous work [6]. We encourage the interested reader to consult the original published version to fully understand the problem we try to solve, our approach and the results and conclusions.

This work has been partially supported by the Spanish Committee of Economy and Competitiveness (TIN2014-55006-R, TIN2017-87330-R) and the UCM (Group 921330).

[1] http://us.blizzard.com/en-us/games/sc/

II. Influence Maps

Influence maps (IM) represent the influence of the game units in the map as a numerical matrix or grid. IM are easy to compute and allow different types of spatial reasoning such as identifying boundaries and areas of control. IM can help to visualize and analyze the tactical deployment of troops in the map, a key factor in most strategy games.

The basic idea is that each unit exerts its influence in a nearby area and the numerical influence value in each tile depends on the distance to the unit and the strength (or value) of the unit. Map tiles under the influence of several units accumulate their influences, so that the more units a player concentrates in an area, the greater will be the values of influence in that part of the map.

Although in 2-player games it is possible to use only one IM to represent the strength of both players using positive numbers for one player and negative numbers for the other, in the general case we need a different IM for each player. StarCraft maps are already divided in logical tiles (96x96 is a medium map) but that level of detail would result in very big IM. Fortunately, IM are matrices that can be easily scaled to smaller sizes to represent the state of the game with the desired level of detail. In our experiments we scale each original influence map to a 4x4 matrix so we only need 16 variables to represent the state of each player in the game and we can use the same representation no mater the original size of the map.

Expert	Precision (%)	Stable from (%)	Confident from (%)	Prec. AI DS (%)	Prec. Human DS (%)
Expert's average	72.22 (22.15)	41.43 (07.98)	67.38 (07.95)	76.46 (07.63)	67.99 (16.15)
SVM	70.37 (39.54)	21.67 (18.35)	NA	74.07 (23.13)	66.67 (57.74)
LDA	72.22 (26.06)	36.67 (26.58)	NA	66.67 (11.11)	77.78 (38.49)

TABLE I
RESULTS FROM THE EXPERIMENT WITH EXPERT PLAYERS. THE TABLE SHOWS AVERAGE VALUES AND STANDARD DEVIATIONS.

III. Game Datasets

We used two different datasets of games. The first one contains 100 4-player games in which all the players were controlled by the internal StarCraft AI. These games were quite long, with an average duration of 72.5 minutes. We extracted the state of the game once every 30 seconds obtaining an average of 145 snapshots per game. The second dataset contains 200 2-player human games downloaded from BWreplays[2], a website that collects games from several specialized websites. Humans play very differently from AIs and the average game duration was 12.5 minutes. In fact, in several games one player defeats the other one with basic combat units in a few minutes and there is not enough time to develop technologies or to expand to new locations.

In games with several players, the chances to win depend to a great extent on the interactions among the other players. For example, the first two players to engage in combat will probably lose several units while the other players keep their armies intact. Another example is that any player who suffers successive attacks from different opponents will not have many chances to survive. One peculiarity of the 2-Human player games was that 80% of the games ended early because one player left when he though he had no opportunity to win. In those cases, we assigned the victory to the remaining player.

We obtained the best results using Linear Discriminant Analysis (LDA) [3] and Support Vector Machines (SVM) [1] with a polynomial kernel. In the first dataset (4-AI player games) we obtain an average precision of 66.62% but we need to wait until the last quarter of the game to predict the winner with precision over 80%. In the second dataset (2-Human player games) we obtain an average precision of 62.04% but we need to wait until the last tenth of the game to make a prediction with a precision of 80%. Human games are more challenging to predict (although there are only two players in the game) because of how fast one bad decision can unbalance the game.

IV. Evaluation with Experts

How good is our system when we compare it with expert human judges? We organized a small experiment in which 14 experienced StarCraft players tried to guess who was going to win in 6 previously recorded games (3 from each dataset). The human judges were asked to decide the winner 9 times in each game (at 10%, 20%, ..., 90% of the game duration) and they could move the camera around the map while the game was paused for as long as they wanted (there was no *fog of war* so all the units were visible).

[2]http://bwreplays.com/

Table I summarizes the results of the experiment. We show the results for the average of the 14 human judges and the 2 best classifiers. The columns represent the average precision and standard deviation, the moments of the game in which the predictions became stable and the experts felt confident, and the precision in each dataset.

We observed big differences among the participants in terms of precision (from 57.41% to 87.04%), but in average human judges obtained a precision of 72%, their predictions became stable at 41% of the game, and they feel confident about their prediction at 67% of the game. Our two best classifiers, SVM and LDA, obtained very competitive results and became stable much sooner than human judges.

V. Conclusions

We conclude that it is possible to build machine learning classifiers based on influence maps that can compete with human experienced players in the task of predicting the outcome of StarCraft games.

During the debriefing at the end of the session, we learned that humans base their decisions on a much richer set of features that we are not able to capture with influence maps. In particular, human judges are able to anticipate and recognize high level strategies and they realize when one player makes a fatal decision. On the other hand, human judges found the minimap very useful and minimaps are similar to influence maps.

References

[1] Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Mach. Learn.*, 20(3):273–297, September 1995.

[2] Holger Danielsiek, Raphael Stur, dAndreas Thom, Nicola Beume, Boris Naujoks, and Mike Preuss. Intelligent moving of groups in real-time strategy games. In Philip Hingston and Luigi Barone, editors, *CIG*, pages 71–78. IEEE, 2008.

[3] R. A. Fisher. The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, 7(7):179–188, 1936.

[4] Christopher E. Miles. *Co-evolving Real-time Strategy Game Players*. PhD thesis, Reno, NV, USA, 2007. AAI3274501.

[5] Santiago Ontañón, Gabriel Synnaeve, Alberto Uriarte, Florian Richoux, David Churchill, and Mike Preuss. *RTS AI Problems and Techniques*, pages 1–12. Springer International Publishing, Cham, 2015.

[6] Antonio A. Sánchez-Ruiz and Maximiliano Miranda. A machine learning approach to predict the winner in starcraft based on influence maps. *Entertainment Computing*, 19(Complete):29–41, 2017.

[7] Jonathan Schaeffer. A Gamut of Games. *AI Magazine*, pages 29–46, 2001.

[8] Alberto Uriarte and Santiago Ontañón. Kiting in rts games using influence maps. In *AIIDE*, 2012.

[9] Albert L. Zobrist. A model of visual organization for the game of go. In *Proceedings of the May 14-16, 1969, Spring Joint Computer Conference*, AFIPS '69 (Spring), pages 103–112, New York, NY, USA, 1969. ACM.

Implementación de generador estocástico de partidas mediante la configuración de dimensiones afectivas

José-Manuel Martínez-González
Escuela Superior de Ingeniería
Universidad de Cádiz
Puerto Real, España
jose.martinezgonzalez@alum.uca.es

Pablo García-Sánchez, Manuel Palomo-Duarte, Pablo Delatorre
Departamento de Ingeniería Informática
Universidad de Cádiz
Puerto Real, España
{pablo.garciasanchez,manuel.palomo,pablo.delatorre}@uca.es

***Resumen*—Enmarcados dentro del ocio interactivo, los videojuegos generan diversas emociones a sus jugadores. Este trabajo describe el diseño y desarrollo de un videojuego que construye en cada partida, de manera estocástica, diferentes versiones del universo narrativo en función de valores que el jugador asigna para las tres dimensiones afectivas: valencia, intensidad y control. Dichas dimensiones se han demostrado correlacionadas con el suspense narrativo evocado por las situaciones de la historia. Éste es medido como un factor entre un rango de valores que es empleado para observar la bondad del sistema. Para ello, se generan aleatoriamente mapas, detectando zonas jugables para incluir en ellas los diferentes elementos del entorno. Se tienen en cuenta el tamaño del mapa y las dimensiones referidas para obtener así una experiencia de juego personalizada en cada caso. La consideración de estas dimensiones implican modificaciones en varias características del juego: enemigos, recursos del protagonista, eventos del terreno y luz ambiental.**

Se espera que este desarrollo sea la base para que futuros investigadores puedan realizar experimentos relacionados con la línea de investigación de la computación afectiva.

***Index Terms*—emoción, videojuegos, narrativa, suspense, computación afectiva**

I. Introducción

Enmarcados dentro del ocio interactivo, los videojuegos generan miedo, alegría, odio o tristeza –de la misma forma en la que lo hacen las películas, las novelas y las series de televisión– mediante un discurso particular que permite evocar diferentes emociones a la audiencia [1]. Si bien la literatura científica contempla ya estos efectos emocionales desde el punto de vista de diferentes estrategias narrativas [2], dichos estudios se circunscriben a la percepción de los jugadores y, excepcionalmente, a la intención de los autores. Más allá de sus elementos estáticos en la trama y el entorno, los juegos actuales modulan típicamente la influencia emocional mediante la sencilla técnica de ajustar el nivel de dificultad, bien a criterio explícito o bien de forma automática, lo que afecta al control percibido sobre la situación y, en consecuencia, a la sensación de indefensión. Sin embargo, esto se considera como una simplificación del impacto potencial que tendría si se favoreciera un mayor dinamismo en la intensidad de emociones. A este respecto, se observa en los desarrollos la carencia de una perspectiva cognitiva que clasifique y cuantifique los elementos del universo del juego respecto de su influencia afectiva, no habiéndose encontrado prototipos experimentales que exploten la interpretación del jugador con objeto de modular las emociones evocadas.

El objetivo de esta contribución es describir el diseño y desarrollo de un generador estocástico de partidas para un videojuego *first-person shooter* (FPS) el cual, mediante la parametrización de valores afectivos, generará un universo del juego específico con objeto de provocar un efecto emocional equivalente en el jugador.

Para ello y en primer lugar, en la Sección II se revisa el trabajo existente sobre las dimensiones afectivas y la generación de emociones en los videojuegos; a continuación, la Sección III presenta el diseño y desarrollo la implementación del videojuego y el generador estocástico, ilustrándose el resultado en la Sección IV; finalmente, en la Sección V se exponen las conclusiones y el trabajo futuro.

II. Trabajos relacionados

Lejos de ser únicamente instrumentos de ocio, los videojuegos generan respuestas de diversa naturaleza en el jugador [3] [4]. Éstas se presentan en forma de emociones, evaluaciones cognitivas o motivaciones [5]. Existe una amplia literatura que analiza estos efectos [6], y que van desde la diversión [7] hasta el horror [8]. No obstante, estas propuestas no parecen contemplar el potencial del discurso del videojuego para evocar dinámica y automáticamente dichas respuestas. De esta forma, las técnicas observadas se circunscriben principalmente a la modificación de los niveles de dificultad [9].

Si bien se evidencia que la dificultad de la partida afecta a la capacidad de enfrentar los desafíos y, por tanto, a la emoción por la indefensión del personaje [10], existen otras dimensiones emocionales que apenas han sido tratadas explícitamente. Dichas dimensiones emocionales fueron estudiadas por Bradley et al. (1999) [11] en su estudio sobre normas afectivas asociadas a términos (ANEW), y son la valencia, la intensidad y el control. La valencia emocional V describe en qué grado un elemento causa una emoción placentera [12];

Este trabajo ha sido financiado en parte por el proyecto VISAIGLE (TIN2017-85797-R) financiado por la Agencia Estatal de Investigación (AEI) usando fondos ERDF; y los proyectos IDiLyCo (TIN2015-66655-R) y DeepBIO (TIN2017-85727-C4-2-P), financiados por el Ministerio de Economía, Industria y Competitividad, y los proyectos FEI InVITAR-IA (Universidad Complutense de Madrid) y SOL-201500054211-TRA (Universidad de Cádiz y Junta de Andalucía).

la intensidad A determina en qué grado un estímulo genera excitación [13]; por último, el control D refleja el dominio que un individuo siente, variando desde la falta total de control hasta el control absoluto [14].

En Delatorre et al. (2017) [15], dichas dimensiones emocionales fueron incluidas en una formulación –enunciada en la Ecuación 1– para la cuantificación del suspense narrativo, encontrándose una correlación de $r = 0{,}8196$, $p < 0{,}05$ entre el suspense estimado por la formulación y los valores reportados por el estudio ANEW. Sin embargo, este experimento se circunscribía a la evocación del suspense en sujetos espectadores, no jugadores.

$$\psi = f(V, A, D) = \frac{\Delta^{-1}V + \Delta A + \Delta^{-1}D}{\Delta_V V + \Delta_V A + \Delta_V D} \qquad (1)$$

donde V, A y D representan, respectivamente, valencia, intensidad y control. El operador $\triangle$ obtiene la diferencia entre el valor actual y el más bajo posible en la escala de medida –valor 1, en los experimentos de Bradley et al. (1999)–; $\triangle^{-1}$ obtiene la diferencia entre el valor más alto posible –valor 9– y el valor actual en la dimensión medida; y $\triangle_{\vee}$, obtiene la diferencia entre el valor más alto y más bajo posible. En consecuencia, la Ecuación 1 devuelve un valor más alto cuanto más baja es la valencia, más alta es la intensidad y más bajo es el control.

En este contexto, se pretende emplear el trabajo de Delatorre et al. (2017) como base para el diseño y desarrollo de un generador estocástico de partidas jugables, varias de cuyas características y restricciones dependen del valor parametrizado por el jugador para cada una de las tres dimensiones emocionales. Si bien el generador no está necesariamente orientado a entornos de suspense, se emplea el resultado de aplicar la Ecuación 1 como factor de evaluación, al ser la única valoración cuantitativa actualmente encontrada en la literatura.

III. Implementación

Desarrollado en Unity 3D[1], el videojuego FPS construido para esta contribución propone al jugador la inmersión en un terreno boscoso compuesto por llanos y elevaciones, en el cual debe localizar una serie de cofres. Para ganar la partida, el jugador debe conseguir todos los cofres antes de que su energía se agote a causa de los ataques de los enemigos. Los cofres se distribuyen al azar por el mapa, en el que también se encuentran los enemigos, cuya tipología, predictibilidad y agresividad dependerá de los niveles de afección emocional especificados. Dicha configuración también afecta a otros eventos del universo, por ejemplo, un árbol que cae cuando el personaje pasa cerca de él o un objeto que cambia bruscamente de lugar. Para ganar la partida, el jugador debe conseguir todos los cofres antes de que su energía se agote a causa de los ataques de los enemigos.

Para la creación de cada partida, en primer lugar el sistema genera un mapa compuesto por explanadas y montañas. A continuación, analiza el terreno para detectar las zonas *jugables* del juego, en la que se ubicarán los diferentes elementos. Posteriormente y en función de los valores de afección emocional en las tres dimensiones, se crean instancias de los diferentes elementos y se colocan en posiciones jugables tomadas al azar. Finalmente, se procede a dar el control al jugador para el comienzo de la partida.

A continuación, se destallan estos pasos.

III-A. Generación automática de terrenos

Para la construcción de terrenos se emplea una implementación de la ecuación de Perlin Noise para Unity3D [16]. Dicha aproximación proporciona valores pseudo-aleatorios óptimos cuando se trata de generar orografías realistas [17]. No obstante, la implementación de Unity3D devuelve valores que tienden a 1 y, por tanto, genera mapas como el que se muestra en la Figura 1. De esta manera y a pesar del realismo del terreno, éste carece de las partes lisas necesarias para insertar los elementos con los que interactuará el jugador.

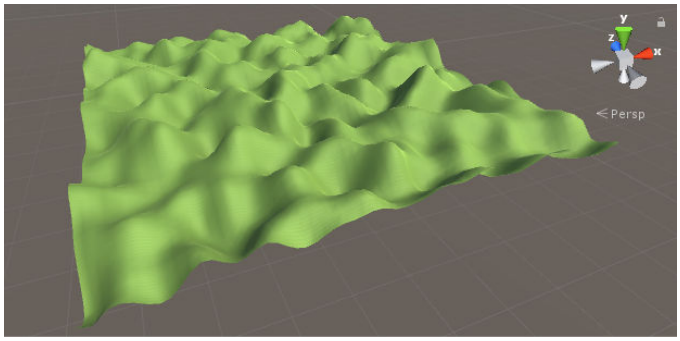


Figura 1: Terreno generado con ecuación de Perlin Noise

Para corregir esto, se modificó el algoritmo para controlar el nivel de ruido [18]. De esta forma se obtienen terrenos con zonas llanas manteniendo suficientemente el realismo deseado, como se ilustra en la Figura 2.

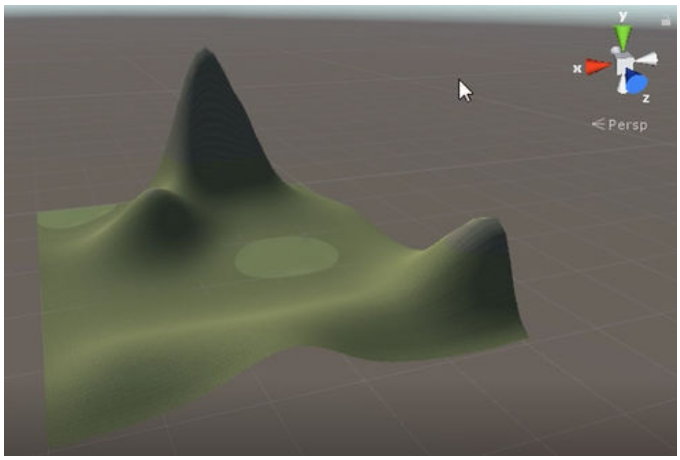


Figura 2: Terreno con ecuación de Perlin Noise modificada

En dicha Figura 2 se aprecia también la aplicación de texturas para permitir la diferenciación visual de las distintas alturas, siendo los niveles más bajos de color verde claro, el

[1] https://unity3d.com/es

verde oscuro para las zonas entre las montañas y, el marrón, sobre las zonas montañosas por las que no es posible transitar.

En relación al tamaño del mapa, se han considerado tres posibilidades: pequeño, mediano y grande. Esto determina el número de árboles y cofres en escena, tal y como se reporta en la Tabla I.

	pequeño	**mediano**	**grande**
tamaño	257	513	1025
árboles	150	300	700
cofres	3	6	10

Tabla I: Número de árboles y cofres según el tamaño de mapa

III-B. Detección de zonas jugables

Una zona se considera jugable si cumple con determinadas restricciones de altura, para lo cual se realiza un recorrido de las distintas zonas del mapa y marcando aquéllas que superan la restricción. A este respecto, se consideró un porcentaje óptimo de terreno jugable aquél que ocupaba el 60 % del mapa. La Figura 3 ilustra el resultado del proceso, mostrando en verde las zonas jugables y, en rojo, las no jugables.

III-C. Inclusión de elementos en la partida

Mediante un menú previo a cada partida, el usuario debe escoger la configuración deseada en cuanto a tamaño de mapa y al valor de cada una de las tres dimensiones afectivas, como se muestra en la Figura 4. En ella se observa también el valor de suspense, resultado de aplicar la fórmula de la Ecuación 1.

Tras pulsar el botón 'GENERAR', da comienzo el proceso de generación del juego.

A partir de la primera población aleatoria de árboles y cofres declarados como elementos estáticos, Unity se encarga de generar una capa sobre el terreno sobre la que pueden desplazarse los personajes teniendo en cuenta los nuevos obstáculos existentes. A su vez, los personajes se conectarán al terreno mediante un componente *NavMesh Agent*. La Figura 5 muestra la capa generada.

Posteriormente se incluyen los personajes no-jugadores (NPC) y eventos, teniendo en cuenta el valor de las dimensiones afectivas. En concreto:

- **Caracterización relativa a la valencia**
  La valencia representa la percepción positiva o negativa del estímulo. Así, las características relativas a la valencia se han relacionado con el tamaño y la gravedad de las posibles amenazas, siendo menores y más débiles cuando mayor es el valor de la dimensión. En el caso de valencia muy alta, los enemigos se sustituyen por animales pacíficos que interactuarán con el jugador
- **Caracterización relativa a la intensidad**
  Se ha considerado que la intensidad emocional del jugador puede corresponderse potencialmente a la capacidad de ataque del personaje. De esta forma, la capacidad de su arma y la cantidad de munición es proporcional a la intensidad escogida.
- **Caracterización relativa al control**
  La dimensión de control se relaciona con la indefensión del personaje, esto es, cuánto dominio percibe el jugador para salir exitoso de la partida. En consecuencia, cuanto mayor control se haya escogido, mayor cantidad de vida tendrá el jugador. Asimismo, se añade un minimapa si el valor de control supera la mitad de su valor. Dicho minimapa mostrará la ubicación de los cofres. En casos en los que el control se aproxime a su máximo, también mostrará la posición de los enemigos. Una imagen del minimapa puede observarse en la Figura 6.
- **Caracterización relativa al cómputo del suspense**
  En esta versión, las interacciones entre las tres dimensiones se han simplificado a través del cálculo de la Ecuación 1. El suspense resultante se relaciona con aspectos del entorno y con la inclusión en la partida de una amenaza constante. Así, la luz ambiental se reduce proporcionalmente a este valor. De igual modo, cuando el nivel es muy elevado -es decir, en las situaciones extremas de muy baja valencia, muy alta intensidad y muy bajo control-, el sistema introduce a un enemigo especial que se moverá por el mapa en busca del personaje, finalizando la partida en caso de colisión con él. Asimismo, se incluyen eventos como árboles que caen u objetos que cambian de lugar.

La Tabla II resume los criterios referidos en este apartado.

IV. Ejemplos de escenarios generados

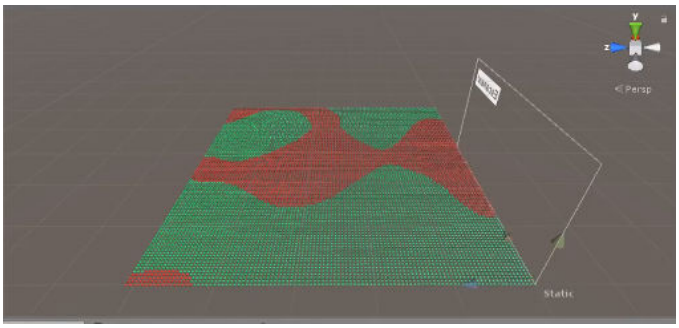
Las Figuras 7a, 7b y 8 ilustran algunos ejemplos generados con la aplicación.

En concreto, las Figuras 7a y 7b muestran una versión donde el nivel de intensidad es bajo, por lo que el jugador no tiene armas. Asimismo, el nivel de control es alto en ambos casos, mostrando el minimapa en la parte inferior. Sin embargo, en conjunto la partida generada en la Figura 7a tiene un mayor valor de suspense, lo que provoca una reducción de luz en el entorno.

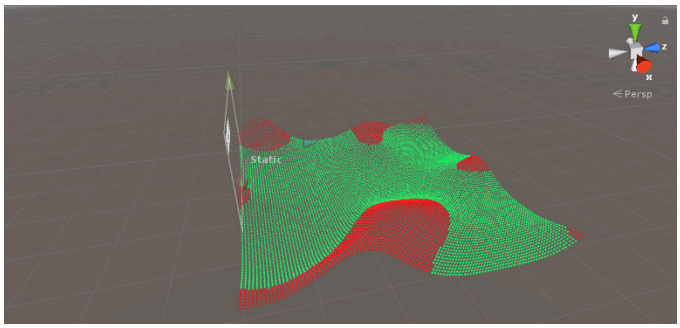
En contraposición, el escenario de la Figura 8 revela una pistola debido a que el nivel de intensidad se parametrizó a un valor entre 0.25 y 0.5. Al ser la dimensión de control menor de 0.5, se observa también la cantidad media de energía y la falta de minimapa. Finalmente, la valencia fue establecida al máximo, por lo que aparecen animales a lo largo del terreno. El valor de suspense, en el rango 0.25 a 0.5, aporta una luz ambiental alta.

V. Conclusiones

Este trabajo presenta el diseño y desarrollo de un videojuego FPS cuyas partidas son construidas de manera estocástica en función del valor de tres dimensiones emocionales: valencia, intensidad y control. Para ello, se ha descrito el proceso de generación aleatoria de los mapas mediante la ecuación modificada de Perlin Noise, la selección de las zonas jugables y la inclusión de los diferentes elementos del entorno, teniendo en cuenta el tamaño del mapa y las dimensiones referidas, y obteniendo así un juego distinto en cada generación. La consideración de estas dimensiones implican modificaciones



(a) Ilustración en dos dimensiones de zonas jugables



(b) Ilustración en tres dimensiones de zonas jugables

Figura 3: Ilustraciones de zonas jugables

	valencia	**intensidad**	**control**	**suspense**
$[0, 0{,}25]$	enemigos duros	sin arma	energía baja	luz máxima, sin eventos
$(0{,}25, 0{,}5]$	enemigos medios	pistola	energía media	luz alta, algunos eventos
$(0{,}5, 0{,}75]$	enemigos débiles	ballesta	energía alta, minimapa con cofres	luz media, varios eventos
$(0{,}75, 1]$	personajes inofensivos	bazoca	intocable, minimapa con cofres y enemigos	luz baja, muchos eventos, enemigo especial

Tabla II: Caracterización de los elementos por valor de las dimensiones emocionales

Figura 4: Menú de configuración de partida

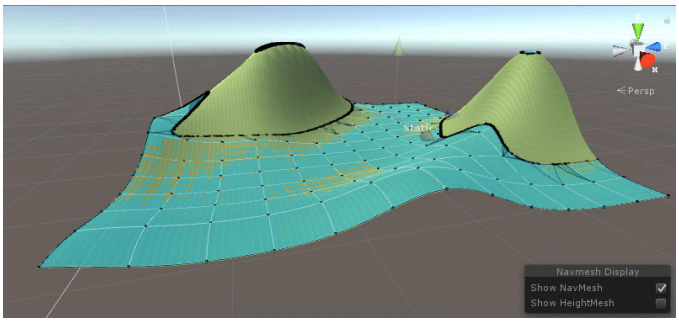


Figura 5: Mapa con zonas jugables calculadas

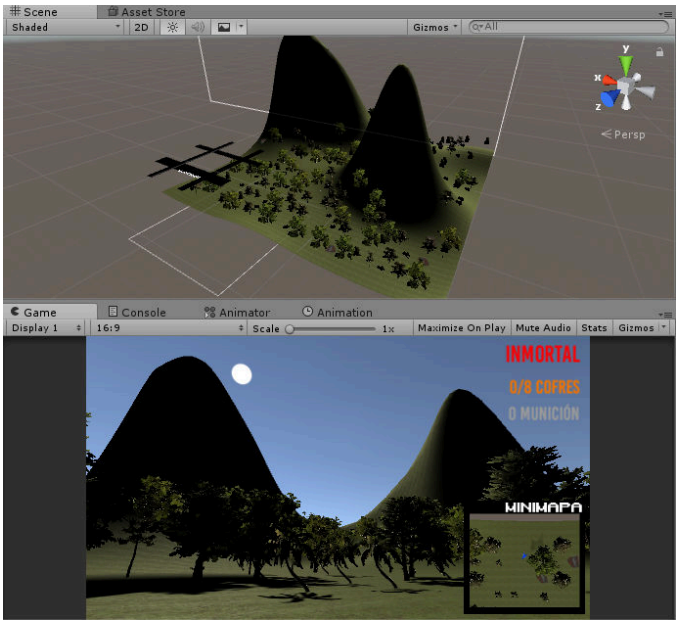
en varias características del juego: enemigos, recursos del protagonista, eventos del terreno y luz ambiental. Se pretende que esta aproximación sirva de base para el diseño de estudios relacionados con la computación afectiva[2].

[2]El código fuente de este software puede descargarse bajo licencia libre en el siguiente enlace: https://github.com/JoseManuelMartinez/GameGenerator

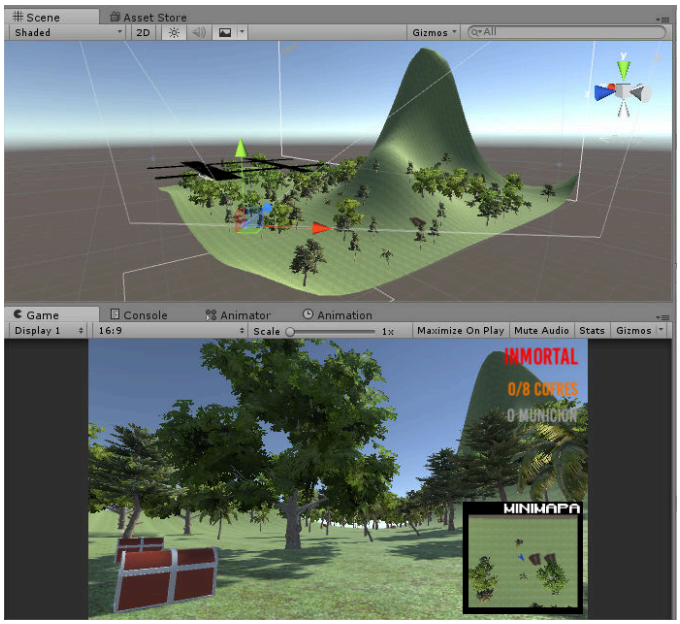
Figura 6: Minimapa

En cuanto al sistema propuesto, algunas decisiones de diseño merecen ser puntualizadas. En primer lugar y dado que el objetivo era presentar un generador estocástico de partidas adaptable a la afección emocional esperada, los criterios por los que se ha asociado cada dimensión a sus respectivas características del universo del juego no están suficientemente sustentadas. De esta forma, por ejemplo, en cierta medida es discutible la consideración de que una mayor intensidad emocional se obtiene cuando el personaje tiene más recursos, o en qué grado el tamaño del enemigo, asociado a la valencia, se solapa con otras dimensiones. No obstante, estas dependencias no son obligatorias, sino escogidas por el diseñador del juego que implemente el generador estocástico.

Por otro lado, aunque la elección de un terreno boscoso parecería limitar el alcance de la propuesta, el sistema no determina éste como el único escenario posible. Al contrario, el proceso de generación estocástica no depende de la elección del entorno ni de la mecánica, sino de las dimensiones emocionales y la formulación en cuanto a la interacción de éstas. Sin embargo, sí puede ser necesario especializar los componentes constructores debido las características del nuevo



(a) Ejemplo de juego generado 1



(b) Ejemplo de juego generado 2

Figura 7: Ejemplos de juegos generados 1 y 2

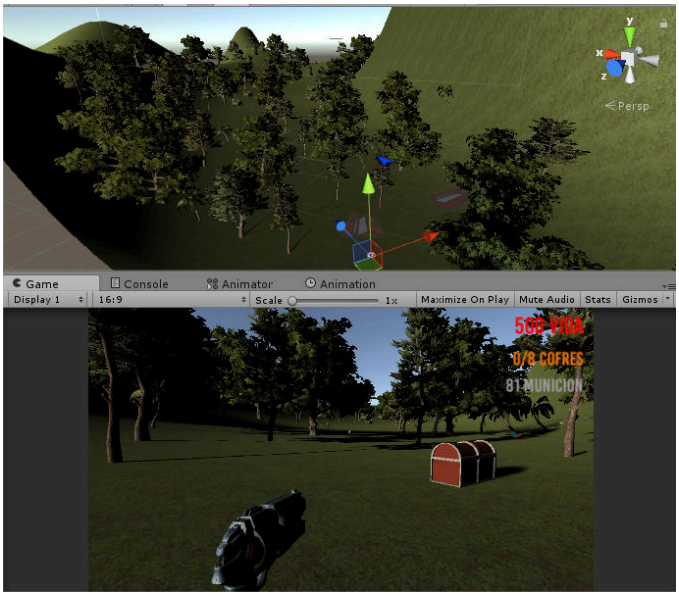


Figura 8: Ejemplo de juego generado 3

universo, lo cual queda a criterio del diseñador.

Asimismo, el entorno inmersivo-afectivo propuesto sugiere la experimentación con periféricos como gafas de realidad virtual o dispositivos de control gestuales[3]. Igualmente, la obtención de las respuestas fisiológicas de los jugadores mediante medidores biométricos o sensores de ondas permitiría evaluar el efecto ponderado de cada una de las dimensiones y, en su caso, la detección de efectos colaterales derivados de la propia tipología del juego. Además, la posibilidad de identificación de las variaciones anímicas del jugador habilitaría nuevas estrategias en la transformación de las partidas durante su ejecución.

Se estima también que un avance sustancial en materia de inclusión dinámica de elementos efectivos vendría de la mano de la combinación del generador propuesto con un sistema de generación automática de historias [19] [20], construyendo la partida mediante la implementación de los NPC como agentes múltiples y teniendo en cuenta la decisión del jugador en momentos críticos de la trama. Esto puede ampliarse a la extensión del mapa conforme el personaje principal avanza, lo cual es aplicable tanto a su tamaño como a su tipología de acuerdo con las dimensiones afectivas y las propias interacciones con el entorno.

No obstante a ello y debido a que el objetivo del proyecto es la evocación de respuestas emocionales específicas, la validez del sistema debe ser probada mediante un estudio con usuarios finales que permita ratificar los resultados obtenidos. Un experimento para la evaluación del videojuego por parte de sujetos humanos se encuentra en fase de diseño y será incluido en posteriores contribuciones.

[3]La versión facilitada ya incluye un módulo para la manipulación de los cofres mediante la tecnología *Leap Motion*.

Referencias

[1] J. Takatalo, J. Häkkinen, J. Kaistinen, and G. Nyman, "Presence, involvement, and flow in digital games," in *Evaluating user experience in games*. Springer, 2010, pp. 23–46.

[2] J. Togelius and G. N. Yannakakis, "Emotion-driven level generation," in *Emotion in Games*. Cham, Switzerland: Springer, 2016, pp. 155–166.

[3] E. Hudlicka, "Affective computing for game design," J. Kienzle, H. Vangheluwe, and C. Verbrugge, Eds.

[4] K. M. Gilleade, A. Dix, and J. Allanson, "Affective videogames and modes of affective gaming: Assist me, challenge me, emote me," in *Conference Proceedings of DiGRA'05, Changing Views: Worlds in Play, Digital Games Research Association*, Vancouver, Canada, 2005.

[5] J. Komulainen, J. Takatalo, M. Lehtonen, and G. Nyman, "Psychologically structured approach to user experience in games," in *Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges*. ACM, 2008, pp. 487–490.

[6] B. Perron, "A cognitive psychological approach to gameplay emotions," in *Conference Proceedings of DiGRA'05, Changing Views: Worlds in Play, Digital Games Research Association*, vol. 3, Vancouver, Canada, 2005, pp. 1–10.

[7] H. Wang, M. Gotsis, M. Jordan-Marsh, D. Spruijt-Metz, and T. Valiente, "Leveling up: Game enjoyment threshold model and player feedback on the design of a serious game," Buffalo, NY, pp. 1–9, 2010.

[8] P. Toprac and A. Abdel-Meguid, "Causing fear, suspense, and anxiety using sound design in computer games," in *Game Sound Technology and Player Interaction: Concepts and Developments*, M. Grimshaw, Ed. IGI Global, 2010, ch. 9, pp. 176–191. [Online]. Available: https://www.igi-global.com/chapter/causing-fear-suspense-anxiety-using/46792

[9] C. Liu, P. Agrawal, N. Sarkar, and S. Chen, "Dynamic difficulty adjustment in computer games through real-time anxiety-based affective feedback," *International Journal of Human-Computer Interaction*, vol. 25, no. 6, pp. 506–529, 2009.

[10] J. Frome and A. Smuts, "Helpless spectators: Generating suspense in videogames and film," *TEXT technology*, vol. 13, pp. 13–34, 2004.

[11] M. M. Bradley and P. J. Lang, "Affective norms for english words (ANEW): Instruction manual and affective ratings," Technical Report C-1, The Center for Research in Psychophysiology, University of Florida, Tech. Rep., 1999.

[12] F. M. Citron, M. A. Gray, H. D. Critchley, B. S. Weekes, and E. C. Ferstl, "Emotional valence and arousal affect reading in an interactive way: neuroimaging evidence for an approach-withdrawal framework," *Neuropsychologia*, vol. 56, pp. 79–89, 2014.

[13] D. E. Berlyne, "Conflict, arousal, and curiosity," *McGraw-Hill Series in Psychology*, vol. XII, 1960.

[14] M. Montefinese, E. Ambrosini, B. Fairfield, and N. Mammarella, "The adaptation of the Affective Norms for English Words (ANEW) for Italian," *Behavior research methods*, vol. 46, no. 3, pp. 887–903, 2014.

[15] P. Delatorre, M. Palomo-Duarte, and P. Gervás, "Formalising suspense from immersive environments," in *Proceedings of 3rd Congreso de la Sociedad Española para las Ciencias del Videojuego (CoSECiVi 2016)*, ser. CEUR Workshop Proceedings, D. Camacho, M. A. Gómez-Martín, and P. A. González-Calero, Eds., vol. 1682, Barcelona, Spain, 2016, pp. 25–36. [Online]. Available: http://ceur-ws.org/Vol-1682/

[16] A. G. Hernández, "Generación aleatoria de terrenos 3d con unity," 2015.

[17] F. K. Musgrave, C. E. Kolb, and R. S. Mace, "The synthesis and rendering of eroded fractal terrains," in *ACM Siggraph Computer Graphics*, vol. 23, no. 3. ACM, 1989, pp. 41–50.

[18] M. Olano, "Modified noise for evaluation on graphics hardware," in *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*. ACM, 2005, pp. 105–110.

[19] C. León and P. Gervás, "A top-down design methodology based on causality and chronology for developing assisted story generation systems," in *Proceedings of the 8th ACM conference on Creativity and cognition*. ACM, 2011, pp. 363–364.

[20] R. H. García-Ortega, P. García-Sánchez, J. J. Merelo, A. San Ginés, and Á. Fernández Cabezas, "The story of their lives: Massive procedural generation of heroes' journeys using evolved agent-based models and logical reasoning," in *Applications of Evolutionary Computation - 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 - April 1, 2016, Proceedings, Part I*, ser. Lecture Notes in Computer Science, G. Squillero and P. Burelli, Eds., vol. 9597. Springer, 2016, pp. 604–619.

How a mobile platform for emotion identification supports designing affective games

Grzegorz J. Nalepa
AGH University of Science and Technology
al. Mickiewicza 30, 30-059 Krakow, Poland
Email: gjn@agh.edu.pl

Barbara Giżycka
AGH University of Science and Technology
al. Mickiewicza 30, 30-059 Krakow, Poland
Email: bgizycka@agh.edu.pl

***Abstract*—Afective computing is a multidisciplinary area of research regarding modeling, identification, and synthesis of emotions using computer-based methods. Affective gaming is dedicated specifically to developing games that use the information regarding player's emotional condition. Such games focus on the emotional dimension of gaming experience, to provide greater player engagement. In this short paper we give an overview of our recent works aimed at developing a mobile software platform for emotion identification using wearable devices. Furthermore, we have been working on the integration of this approach with the design and development of affective games.**

I. Introduction

In this paper we report on recent results presented in the *Future Generation Computing Systems* [1], and relate it to our most recent follow up work on applications to computer games presented at the *IEEE GEM* conference in August 2018 [2].

We believe that information regarding the affective condition of a user can lead to a better understanding of interactions of users with machines. This is one of the applications of Affective Computing (AfC), that was originally introduced in [3]. AfC builds on the assumption that emotions are both physical and cognitive, and can be modeled using low-level physiological data, such as heart rate (HR) or galvanic skin response (GSR). Numerous problems that arise regard specific measurment hardware for these signals, which should be both accessible and unobtrusive, as well as appropriate software – to handle the collected data. Wearable devices, such a wristbands, seem like a promising solution for this matter. In our research we have been working towards developing computational models of emotions, that could be used based on the measurement data from wearable devices. We envisage several uses of such models, including affective gaming.

Emotional layer of gaming experience has brought attention of many research fields [4]–[6]. Data of player's emotional state (derived from her behavior on one or more levels, from physiological to some general behavior metrics) can be used to develop systems comprising of affect models. Affective gaming, stemming directly from a more global account – affective computing (AfC), applies precisely this approach.

In the next two sections we describe our recent works in these two areas, focusing on these two keywork papers.

II. Research on the AfCAI Platform

Our prelimary proposal for combining affective computing with context-aware systems for ambient intelligence applications (so-called *AfCAI* proposal) was presented in the first *AfCAI* workshop (see https://www.affcai.eu). It was based on the use of wearable devices as sources of affective physiological data. This data could be used to exend the notion of context in context-based reasoning [7]. We also developed new software called *Bandreader* for supporting our experimental procedures using a range of wearables [8].

The synthesis of this stage of work can be found in our recent paper [1] in *FGCS*, where we elaborate on detection of affective states, their proper identification and interpretation with use of wearable and mobile devices. We proposes a data acquisition layer based on wearables to gather physiological data, and we integrate it with mobile context-aware framework.

Our solution extends the platform we developed for building mobile context-aware systems [7]. The architecture of this extended platform is presented in Fig. 1. It is an extension of standard Model-View-Controller software architectural pattern that includes context and adaptability as a part of the model. The *adaptable model* layer of the architecture is responsible for discovery and adaptation to user long-term preferences and habits (profiles), but also should provide mechanisms allowing to react on dynamically changing environmental conditions. The *context-based controller* provides mechanisms for context-based mediation between the user and the system that allows the system to resolve vagueness and incompleteness of background data. We extended this basic architecture with two components marked red in Fig. 1: 1) Physiological context provider plugin, responsible for obtaining heart rate, GSR and possibly other data from wearable sensors, and 2) Emotion detection and recognition module, which objective is the discovery of correlations between context and user emotional state. Our solution offers a non-intrusive measurement of affective thanks to the use of wearable devices, such as wristbands. As such, it is suitable for gaming environments.

III. Towards the Design of Affective Games

Our first works in the area of applications of AfC to gaming were described in [9]. We assumed that a change in the affective condition of a player can be detected based of the monitoring of physiological signals following the James-Lange

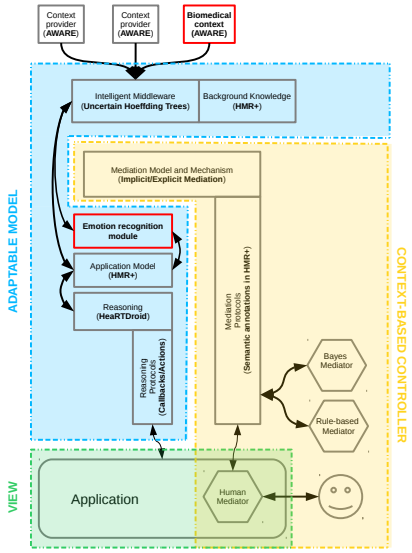


Fig. 1. Our architecture extended with emotion recognition and processing [1]

theory of emotions [10]. We build on the idea of game design patterns [11], which originate in the repetitive nature of game mechanics. We identified a preliminary set of patterns that can be considered affective. Then we demonstrated how these patterns can be used in a design of a scroll-runner game.

Affective games comprise various techniques and tools, which highlight their distinction from the regular kinds of games. We observed that they also require a thoroughly different design approach. To address this challenge, in our most recent work ath *IEEE GEM* [2] we formulated a framework based on game design patterns, more specifically – affective ones. The considered patterns integrate design strategies with measuring player's emotional reactions.

The main contribuition is the proposal of the *Affective Game Design Patterns Framework*. We build on the above mentioned concept of game design patterns introduced by Björk and Holopainen [11]. Patterns form an intricate web of interrelations, and can be abstracted from different levels of interaction with the game. Furthemore, in [12] we built on this idea and discussed the application of our approach to build games with the so-called affective loop.

Our account (Fig. 2) is built on premises: 1) One can infer about their affective state based on physiological data on e.g. HR and GSR. 2) This data may be used to develop both models of affect for in-game agents and player affect, 3) By applying affective game design patterns game developers can use knowledge on the described gaming situation early in the design phase to provide affectively adaptive game world.

To evaluate our approach, four series of experiments have taken place. In [9], where we present our approach and explain our first experimental game designed with affective patterns in mind. The findings and observations from all of conducted studies allowed us to critically evaluate several measurement devices in the light of afffective computing purposes. We also confirmed reliability of our custom mobile application for data acquisition from wearable devices [8].

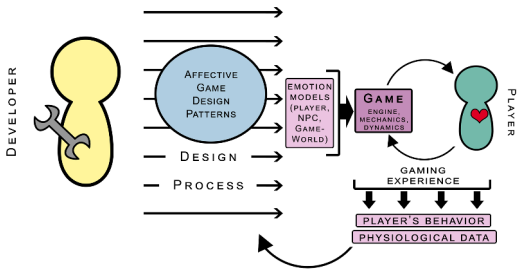


Fig. 2. Affective Game Design Patterns Framework [2]

IV. Conclusion and Future Works

We provided a general overview of our recent works on the intersection of affective computing and video games. Our ultimate goal is to provide a unified affective game design framework. As a foundation, we suggest the use of affective game design patterns, which are a convenient design tool that realizes the affective loop. Currently, we examine the possibilities brought by Unity API delivered for BITalino (http://bitalino.com) we use and experiment with our first affectively adaptive game prototypes. As for our future plans, we will aim at identifying correlations between affective game events, evoked by affective design patterns, and physiological reactions of the player.

References

[1] G. J. Nalepa, K. Kutt, and S. Bobek, "Mobile platform for affective context-aware systems," *Future Generation Computer Systems*, 2018. [Online]. Available: https://doi.org/10.1016/j.future.2018.02.033

[2] B. Giżycka and G. J. Nalepa, "Emotion in models meets emotion in design: building true affective games," 2018, accepted to IEEE GEM 2018.

[3] R. W. Picard, *Affective Computing*. MIT Press, 1997.

[4] K. M. Gilleade and A. Dix, "Using frustration in the design of adaptive videogames," in *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, ser. ACE '04. New York, NY, USA: ACM, 2004, pp. 228–232. [Online]. Available: http://doi.acm.org/10.1145/1067343.1067372

[5] R. El Kaliouby, R. Picard, and S. Baron-Cohen, "Affective computing and autism," *Annals of the New York Academy of Sciences*, vol. 1093, no. 1, pp. 228–248, 2006.

[6] K. Gilleade, A. Dix, and J. Allanson, "Affective videogames and modes of affective gaming: assist me, challenge me, emote me," *DiGRA 2005: Changing Views–Worlds in Play.*, 2005.

[7] S. Bobek and G. J. Nalepa, "Uncertain context data management in dynamic mobile environments," *Future Generation Computer Systems*, vol. 66, no. January, pp. 110–124, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.future.2016.06.007

[8] K. Kutt, G. J. Nalepa, B. Giżycka, and P. Jemioło, "Bandreader – a mobile application for data acquisition from wearable devices in affective computing experiments," accepted to HSI 2018.

[9] G. J. Nalepa, B. Gizycka, K. Kutt, and J. K. Argasinski, "Affective design patterns in computer games. scrollrunner case study," in *Communication Papers of the 2017 Federated Conference on Computer Science and Information Systems, FedCSIS 2017*, 2017, pp. 345–352. [Online]. Available: https://doi.org/10.15439/2017F192

[10] W. James, "What is an emotion?" *Mind*, no. 34, pp. 188–205, 1884.

[11] S. Björk and J. Holopainen, *Patterns in Game Design*. Charles River Media, 2005.

[12] B. Giżycka, G. J. Nalepa, and P. Jemioło, ""AIded with emotions" - a new design approach towards affective computer systems," *ArXiv e-prints*, Jun. 2018.

Desarrollo de un Modelo Estándar de Interacción para Entornos Virtuales

Juan Luis Berenguel Forte*, Francisco Luis Gutiérrez Vela†, Patricia Paderewski Rodríguez‡ y Daniel Pérez Gázquez§
Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada
Granada, España
*Email: juanlubf@ugr.es
†Email: fgutierr@ugr.es
‡Email: patricia@ugr.es
§Email: danielpg@ugr.es

***Resumen*—Dentro del ámbito de los videojuegos de inmersión virtual existe gran variedad de modelos de interacción, ya que los desarrolladores utilizan técnicas muy específicas para sus aplicaciones. Este trabajo trata de recopilar información acerca de las distintas técnicas existentes dentro de dichos entornos virtuales, ya sea bajo el paradigma de la realidad virtual (RV) o el paradigma de la realidad aumentada (RA), para extraer un modelo estándar que satisfaga las necesidades de los desarrolladores y permita la implementación de una librería que de soporte a las interacciones descritas en dicho modelo.**

I. Introducción

Las aplicaciones que se desarrollan para entornos tanto de realidad virtual como de realidad aumentada necesitan usar técnicas de interacción que se adapten a los dispositivos de entrada (mandos, *joysticks*, móviles táctiles ...) y dispositivos de salida (móviles, gafas, proyectores ...) que los sistemas de realidad virtual y aumentada proporcionan a los usuarios. Aunque estos entornos virtuales (EV) llevan varias décadas entre nosotros, la mayor parte de la investigación se ha centrado en el desarrollo de la tecnología que la sustenta [1], por lo que nos encontramos que el uso de técnicas de interacción no está generalizado y cada aplicación usa e implementa técnicas muy específicas. Una gran parte de las aplicaciones que se desarrollan para estas tecnologías son videojuegos, por lo que el uso de técnicas correctas en estos entornos pueden ser la clave para conseguir que un videojuego sea un éxito o un fracaso, ya que influyen en gran medida sobre la experiencia que va a percibir el jugador.

Teniendo en cuenta esta necesidad, en este trabajo proponemos un modelo de abstracción de las técnicas de interacción para ambos paradigmas que permita generalizar las técnicas existentes e implementar un proyecto de prueba de distintas técnicas que de soporte a los desarrolladores a la hora de determinar qué técnicas utilizar.

II. Motivación

Dado que en la actualidad no existe un modelo estándar de interacción para entornos virtuales, se produce una discordancia entre las necesidades de los usuarios de este tipo de software y las implementaciones que se realizan. En ocasiones, este problema puede desembocar en el aumento de la dificultad para el desarrollo de una aplicación efectiva por parte de los desarrolladores, o en la implementación de una técnica no adecuada para el contexto de uso de la aplicación.

En el caso de las aplicaciones de realidad virtual el problema se incrementa, pues implementar un sistema de navegación dentro de este entorno suele resultar dificultoso debido a los problemas relacionados con la sensación de mareo, la pérdida de inmersión y la desorientación, producto de sentir que se está realizando un desplazamiento, pero el cuerpo no se está moviendo físicamente.

II-A. Objetivos propuestos

El propósito de este trabajo consiste en realizar una revisión de las principales técnicas de interacción existentes, atendiendo a su facilidad de uso y a su extensión dentro de los entornos virtuales, así como de otras propuestas anteriores de clasificación de dichas técnicas de interacción, para proponer un modelo abstracto que permita a los desarrolladores tener una guía de implementación para saber cuáles son las mejores técnicas y por qué para las necesidades específicas de su proyecto sin que tengan que recurrir a implementar las suyas propias.

Como trabajo paralelo a este análisis de las técnicas de interacción se implementará un proyecto de ejemplo que permita utilizar las técnicas de mayor relevancia (tanto para realidad virtual como para realidad aumentada) para poner a prueba el modelo y poder evaluar la experiencia de los usuarios. Esto nos permitirá poder evaluar tanto el modelo de interacción propuesto, como las propias técnicas que sean implementadas.

III. Estado del arte

En la literatura científica existen muchos trabajos donde sus autores proponen distintos modelos de clasificación de las interacciones que se dan lugar entre los usuarios y el entorno virtual. En esta sección se citarán algunos de los trabajos que se han encontrado más relevantes para nuestro objetivo.

En [2] se propone una clasificación en 2 grandes categorías principales, que luego se subdividen en varias subcategorías:

- Técnicas egocéntricas: engloba las técnicas que permiten al usuario fundirse con el entorno virtual. Se subdivide en manipulación directa, control físico y control virtual.
- Técnicas exocéntricas: engloba las técnicas que hacen que el usuario no se sumerja en el entorno virtual, sino que se siente un elemento ajeno a este. Se subdividen en manipulación directa, control físico y control virtual.

En [3] se propone una división de las técnicas de interacción en RA en tangibles (se basan en interactuar con un dispositivo físico) e intangibles (se basan en interactuar sin el dispositivo físico). Esta categorización de las técnicas se verá con más detalle en la sección V-B. Adicionalmente, en este trabajo midieron el grado de usabilidad que presentaba cada conjunto de técnicas, resultando el grupo de técnicas tangibles ser el que resultaba más fácil de usar y menos estresante para los usuarios. No obstante, las técnicas intangibles resultaron ser más atractivas y divertidas para los usuarios.

En [4] se propone una categorización de las técnicas en 3 modalidades:

- Interacción directa del usuario: engloba todas las técnicas que requiere el reconocimiento del cuerpo del usuario (seguimiento de manos, reconocimiento de gestos, etc).
- Controles físicos: engloba las técnicas que requieren la utilización de un dispositivo físico, como botones, *joysticks*, volantes, etc.
- Controles virtuales: engloba las técnicas que utilizan un elemento virtual como medio de interacción con el entorno. Dentro de este ámbito, prácticamente cualquier elemento virtual puede ser utilizado para tal propósito.

IV. Técnicas de interacción en entornos virtuales

El campo de la Interacción Persona-Ordenador (IPO) abarca los campos de estudio relativos al diseño, estudio y planificación de la interacción producida entre los humanos y las computadoras [3]. Dentro del campo de la IPO existen técnicas de interacción especialmente dirigidas a los entornos virtuales, las cuales definen el conjunto de interacciones que pueden realizar las personas para relacionarse con el entorno, ya sea mediante el uso de un dispositivo o no.

Con el aumento de la publicación de videojuegos que utilizan estos entornos virtuales como plataforma, la investigación en técnicas de interacción en entornos virtuales cobra una gran importancia, pues son la la clave de una buena inmersión del jugador en dichos entornos, lo cual se traduce en mayor disfrute por parte del jugador y mayor probabilidad de éxito para el videojuego.

Podemos dividir las interacciones en dos grandes grupos: interacciones avanzadas e interacciones básicas.

Según [4], las interacciones básicas en los entornos virtuales se pueden dividir en los siguientes grupos: selección, manipulación, escalamiento, navegación e interacción con menús virtuales y *widgets*. No obstante, las interacciones se pueden clasificar en dos paradigmas distintos, en función de si operamos en realidad virtual (local y a distancia) o en realidad aumentada (tangible e intangible), tal y como se verá en el capítulo V.

Por otro lado, las interacciones avanzadas serían aquellas que están compuestas por un conjunto de interacciones básicas. Como ejemplo, la acción de lanzar una pelota sería una interacción avanzada compuesta por una selección y serie de acciones de manipulación. Debido a esta relación de composición, sólo analizaremos en profundidad las interacciones básicas.

IV-A. Selección

Según [5], la selección de un objeto virtual hace referencia al acto de elegir o especificar un objeto en concreto para un propósito específico. Este acto de selección debe dividirse en dos pasos: primero se ha de identificar el objeto a seleccionar y después se procede a realizar la selección de dicho objeto identificado.

Para realizar la selección de un objeto virtual podemos usar distintas técnicas atendiendo al tipo de entorno virtual en el que estemos. En realidad virtual podemos usar técnicas locales o a distancia, y en realidad aumentada, podemos usar técnicas tangibles e intangibles.

Además, también es necesario que el usuario reciba algún tipo de *feedback* para saber que la selección se ha realizado exitosamente [4].

IV-B. Manipulación

La interacción de manipular un objeto en un entorno virtual hace referencia a la tarea de modificar las propiedades de la posición y la orientación de dicho objeto. Es necesario destacar que para realizar la manipulación de un objeto es necesario haber realizado una selección previa de dicho objeto, aunque para realizar la selección de un objeto no es necesaria la manipulación del mismo. Nuevamente, para realizar estas acciones disponemos de técnicas locales y a distancia (en RV), y de técnicas tangibles e intangibles (en RA).

Dentro de la manipulación hay 3 aspectos esenciales a controlar: el cambio en la posición, el cambio en la orientación y el centro de rotación [4].

El **cambio en la posición** hace referencia al número de unidades de distancia en el mundo virtual en las que dicho objeto es desplazado, además de en qué eje(s) se produce el desplazamiento. La correspondencia entre las unidades de distancia del mundo virtual y las del mundo real puede ser 1:N, es decir, el objeto no tiene por qué desplazarse en el mundo virtual de forma análoga al desplazamiento que nosotros realicemos en el mundo real.

El **cambio en la rotación** hace referencia al número de unidades de rotación en el mundo virtual en las que dicho objeto es rotado, además de en qué eje(s) se produce la rotación. Al igual que con la posición, es posible establecer una correspondencia 1:N entre el mundo virtual y el mundo real.

El **centro de rotación** es el punto en torno al cuál el objeto realizará la rotación. Dicho punto puede ser el centro del objeto virtual u otro punto establecido arbitrariamente por el usuario.

Como ejemplo de una de las técnicas de manipulación en RV más utilizadas se ha observado el uso de los mandos como manos virtuales y el botón de disparo trasero para simular el agarre de un objeto de virtual para luego moverlo y/o rotarlo.

IV-C. Escalamiento

El escalamiento consiste en modificar el tamaño de un objeto dentro del mundo virtual, ya sea para hacerlo más grande o para hacerlo más pequeño. Al igual que la manipulación, necesita que se haya realizado previamente la selección del objeto que se pretende escalar y se puede realizar mediante técnicas locales y a distancia (en RV) y mediante técnicas tangibles e intangibles (en RA).

Para escalar un objeto es necesario definir dos parámetros clave: el centro de escala y el factor de escala [4].

El **centro de escala** es el punto que se utilizará como referencia para realizar el escalamiento: los puntos del objeto se acercarán al centro de escala cuando se disminuya el tamaño del objeto y se alejarán de dicho centro cuando se aumente el tamaño del objeto. Al igual que pasaba con el centro de rotación, este punto puede ser el centro del objeto o cualquier otro punto arbitrario determinado por el usuario.

El **factor de escala** define el número de unidades de tamaño que aumentará o disminuirá el objeto virtual al ser escalado. Al igual que en el desplazamiento y la rotación, no tiene por qué guardar una relación 1:1 con el mundo real.

IV-D. Navegación

Uno de los aspectos fundamentales en el desarrollo de aplicaciones en entornos de realidad virtual es la navegación, ya que tal y como se explica en [6], el proceso de moverse por un entorno de realidad virtual puede producir mareos, e incluso vómitos, a los usuarios, aunque no siempre afecta de la misma manera a todos los usuarios, pues existen caracteres diferenciadores, tales como el género, la edad y la predisposición.

Por otra parte, en las aplicaciones de realidad aumentada generalmente la navegación no supone ningún problema, ya que para que se produzca un movimiento de la cámara, se ha de mover el dispositivo que se está utilizando para ver los objetos virtuales, lo cual implica el propio movimiento del usuario, que es consciente en todo momento de su situación en el mundo real (al contrario que sucede en la realidad virtual).

Según [7], la navegación consiste en seguir un curso mediante movimientos dirigidos. Unido a esta definición de navegación, también define el concepto de *wayfinding* (localización de camino) como la determinación de una estrategia, dirección y curso necesarios para alcanzar un destino deseado.

Para conseguir una navegación efectiva dentro de los entornos de realidad virtual es necesario tener en cuenta un serie de **factores que afectan a la navegación**, que se derivan directamente de 6 características que están relacionados entre sí: entorno, usuario, tarea, estrategia de navegación, ayudas a la navegación y controles de movimiento [8].

Entorno describe atributos que están directamente relacionados con la estructura del mundo virtual como el tamaño, el grado de abstracción a representar o si se trata de un entorno multiusuario o no. Es un factor que depende en gran medida del tipo de aplicación.

Usuario define las características que tiene un usuario, como su edad, su género, sus habilidades y sus debilidades.

Tarea define el objetivo que el usuario pretende conseguir dentro de la aplicación. La navegación deberá ser acorde al tipo de aplicación y al objetivo que tenga el usuario.

Estrategia de navegación describe conceptos que se usan para navegar por un entorno, tales como "buscar", "navegación vectorial" y "navegación posicional", que a su vez se componen de acciones básicas como "alcanzar el objetivo", "seguir el curso" o "seguir la pared" [9].

Las ***ayudas a la navegación*** consisten en un conjunto de ayudas (que pueden ser parte del entorno u objetos externos) que ayudan al usuario a determinar la dirección de la navegación, a determinar su posición o ambas cosas.

Los ***controles de movimiento*** abarcan tanto los dispositivos *hardware* como el *software* necesarios para realizar la navegación dentro del mundo virtual.

Un aspecto importante a tener en cuenta son los **tipos de navegación**, ya que la navegación en entornos de realidad virtual se puede llevar a cabo utilizando distintos paradigmas, de hecho, es normal encontrar aplicaciones de realidad virtual que implementan más de un tipo de navegación. Los paradigmas más usados son la navegación basada en vuelo, navegación de movimiento libre, navegación de movimiento vehicular, navegación basada en nodos, navegación de escalada, teletransportación basada en arco, navegación basada en un avatar fantasma, navegación usando puntos mallados, teletransporte a cámaras prefijadas y navegación basada en parpadeo (*blink*) [6]. Asimismo, también existe la navegación en raíles [10], la navegación basada en la inclinación del usuario [11], la navegación por gestos [12] e, incluso, es posible utilizar elementos *hardware* específicos que nos permiten navegar por el mundo virtual aumentando el grado de inmersión y de simulación de la realidad.

La navegación basada en arco es uno de los tipos de navegación más utilizados, la cual consiste en lanzar desde el mando un rayo virtual parabólico que nos indica en el suelo dónde vamos a ser teletransportados una vez soltemos el botón.

Por último, cabe destacar que también existen aplicaciones de realidad virtual donde el entorno se construye de forma que el usuario no tenga que desplazarse por el mundo virtual, con lo que se evita la navegación.

IV-E. Interacción con menús virtuales y widgets

La interacción con menús virtuales dentro de los entornos virtuales ofrece a los usuarios una nueva forma de poder interactuar con los objetos de dichos entornos, en ocasiones, facilitando tareas que serían complejas de otra manera. Según [4], los menús de interacción pueden tener una, dos o incluso

tres dimensiones, con una relación directa de aumento de posibilidades de interacción y de complejidad de manejar. No obstante, existen ciertos problemas y limitaciones que son independientes de la dimensión del menú, tales como la posición óptima para colocar el menú dentro de la interfaz del usuario o cómo acceder a él (utilizando un botón, un comando de voz, un gesto específico, etc).

Dentro de este ámbito, un ejemplo común de interacción sería la manipulación directa de objetos de interfaz virtuales, tales como pulsar un botón o desplazar una guía.

V. Modelo de abstracción propuesto

Tanto en los entornos de realidad virtual y de realidad aumentada, la gran variedad de formas de interacción entre los usuarios y las aplicaciones genera problemas a los desarrolladores, que no disponen de un modelo estándar y simplificado que les permita decidir qué tipo de interacción es la más adecuada para las necesidades de su aplicación. No obstante, ambos entornos comparten ciertas similitudes que permiten proponer un modelo que pueda aportar una solución genérica para ciertas interacciones comunes.

Para poder establecer un modelo abstracto de las interacciones es necesario establecer una correspondencia entre una necesidad real de interacción en un entorno específico y una o varias técnicas que la satisfagan. Por lo tanto, es imperante conocer las necesidades del desarrollador, que para poder definirlas se pueden dividir en tres pasos: qué se quiere hacer, cómo se quiere hacer y las necesidades específicas (si las hubiere) relacionadas con el contexto específico de uso de la aplicación. Esta división se verá con detalle en la sección V-D.

Asimismo, para realizar el proceso de abstracción que englobe ambos entornos virtuales, es necesario entender en qué tipos de paradigmas se clasifican las interacciones en entornos de realidad virtual y de realidad aumentada para poder abstraerse aún más y proponer un modelo más abstracto que nos permita definir una categorización que abarque tanto las técnicas de realidad virtual como las de realidad aumentada pero manteniendo la posibilidad de profundizar y realizar una clasificación más concreta hasta llegar al nivel de implementación.

Según [4], en un entorno de realidad virtual se pueden diferencias técnicas según si la interacción entre el usuario y el entorno virtual se hace a nivel local o a distancia.

Por otra parte, según [3], dentro del ámbito de la realidad aumentada, podemos clasificar principalmente las interacciones en dos grandes categorías: técnicas tangibles y técnicas intangibles.

V-A. Interacción en realidad virtual

Cuando utilizamos aplicaciones de realidad virtual contamos con un dispositivo visualizador (*Head Mounted Display*, HMD), uno o más controles físicos (mandos) y, en ocasiones, otros dispositivos auxiliares, como sensores, que nos ayudan a posicionar a los usuarios, u otros objetos, en el espacio virtual. El uso de estos dispositivos permite que podamos tener una representación virtual de un elemento del mundo real. Por ejemplo, el mando que usamos en el mundo real, se transforma en una mano dentro del mundo virtual que podemos controlar mediante los botones y el movimiento de dicho mando. No obstante, estos dispositivos también pueden ayudarnos a interacciones con elementos que no estén a nuestro alcance.

Por lo tanto, en el ámbito de la realidad virtual las interacciones se pueden realizar de dos formas principalmente: de forma local y a distancia [4].

En la **interacción local** el usuario, que es posicionado en el mundo virtual mediante el HMD, ha de estar en una posición cercana al objeto virtual y, utilizando uno de los mandos, que es visualizado dentro del entorno virtual (generalmente como una mano virtual) puede interactuar con el objeto deseado. Es decir, en este paradigma de interacción es necesario que el objeto virtual esté al alcance de la mano del usuario dentro del entorno virtual.

En la **interacción a distancia** existe mayor libertad de acción, pues el usuario puede utilizar, nuevamente, un mando para apuntar a distancia a un objeto y seleccionarlo (*raycasting*), seleccionar el objeto que esté mirando, seleccionar un objeto mediante gestos o, incluso, mediante comandos de voz.

V-B. Interacción en realidad aumentada

En la realidad aumentada se utiliza un dispositivo con una cámara que nos permite ver objetos virtuales superpuestos en nuestro entono físico. Por ello, en el ámbito de la realidad aumentada podemos interactuar de manera tangible y de manera intangible.

La **interacción tangible** es cualquier interacción en la que el usuario utiliza un dispositivo físico para llevarla a cabo [3]. Esta puede ser de dos tipos: basada en toque, donde el usuario selecciona un objeto tocando la pantalla de un dispositivo, o basada en dispositivo, donde la interacción se realiza por medio de los datos recopilados de los sensores del dispositivo, como el acelerómetro.

La **interacción intangible** hace referencia al tipo de interacción que se realiza sin una interacción física del usuario con la máquina (gestos en el aire, interacción por voz, etc) [3]. Las interacciones intangibles se pueden dividir en dos grupos: basadas en marcadores y no basadas en marcadores.

V-C. Abstracción de las técnicas de interacción

Tal y como se vio en la sección V-B y según el criterio de clasificación de [3], dentro del ámbito de la realidad aumentada, las interacciones implementadas se pueden clasificar en: tangibles e intangibles.

Adicionalmente, tal y como se vio en la sección V-A, en [4] se clasifican las técnicas de interacción dentro de los entornos de realidad virtual en técnicas locales y a distancia.

Utilizando como fuente estas dos nomenclaturas, se propone una clasificación más abstracta, que divide las interacciones en directas e indirectas, y permita generalizar la clasificación de las técnicas de interacción tanto de realidad virtual como de realidad aumentada, con el fin de exponer las características

comunes a ambos entornos. Utilizando esta clasificación, se puede realizar una correspondencia entre las acciones que podemos realizar en ambos entornos virtuales, a qué necesidad real corresponden y qué posibles implementaciones podemos utilizar para satisfacer dichas necesidades, según el tipo de técnica que queramos usar.

Interacción directa. Este grupo de interacciones engloban todas aquellas interacciones en las que el usuario se relaciona de una forma directa, es decir, sin ningún tipo de intermediario con un objeto virtual. Para ello, puede utilizar tanto interacciones con su propio cuerpo (tales como gestos o comandos de voz), como interacciones que utilicen un dispositivo físico (por ejemplo, coger un objeto virtual con un mando). Los dispositivos físicos no se consideran como intermediarios porque en el entorno virtual actúan como un elemento que es capaz de virtualizar una parte del cuerpo del usuario o una interacción suya. En el apartado de interacción indirecta se explican con más detalle los elementos considerados como intermediarios.

Tal y como se explicó anteriormente, esta categoría estaría a un nivel más alto de abstracción que las interacciones mostradas en la sección, por lo que dentro de ésta podrían incluirse tanto técnicas de interacción local y a distancia, para realidad virtual, y técnicas de interacción tangible e intangible para realidad aumentada.

Interacción indirecta. Este tipo de interacciones están basadas en la utilización de un elemento virtual intermediario que sirve para interaccionar con otro objeto virtual. Tal y como se comentó en el apartado de interacción directa, los mandos utilizados en realidad virtual no se consideran elementos intermediarios debido a que en realidad son dispositivos físicos del mundo real que simplemente permiten tener una representación virtual de ellos mismos.

Por lo tanto, un elemento intermediario será todo aquel que existe únicamente dentro del entorno virtual. Esto también es aplicable a la realidad aumentada, donde el dispositivo utilizado para visualizar los elementos virtuales tampoco es considerado un intermediario. Un ejemplo de este tipo de interacción podría ser utilizar un bate de béisbol virtual para golpear una pelota, también virtual.

Al igual que en las interacciones directas, en esta clasificación estarían incluidas tanto técnicas locales, técnicas a distancia, técnicas tangibles y técnicas intangibles.

V-D. Guía de diseño

Una vez conocido cómo se agrupan las técnicas de interacción, podemos empezar a establecer de qué manera se han de guiar las necesidades de los desarrolladores para que sepan de qué técnicas disponen para sus aplicaciones y qué características tienen, ya que esto será un aspecto fundamental para el contexto de uso de la aplicación. Ya que la categorización de las técnicas realizada en la sección anterior corresponde a un planteamiento a nivel de diseño, las necesidades básicas del desarrollador (qué quiere hacer y cómo quiere hacerlo) se mapearán también a este nivel de abstracción. En último lugar, las necesidades específicas de la aplicación marcarán qué tipo de implementación se usará, una vez establecido el criterio de filtro a nivel de diseño. Para establecer estos criterios de selección se puede hacer con una estructura de pregunta - respuesta.

Qué hacer define la acción que se desea realizar. Puede tratarse tanto a nivel de interacciones básicas como a nivel de interacciones avanzadas (ver sección IV). Esta decisión se encuentra en la capa de diseño.

Ejemplo: “Quiero lanzar una pelota en RV” - “Debes implementar técnicas de selección y manipulación”.

Cómo hacerlo define cómo se quiere llevar a cabo la acción. Es decir, define si se quiere interactuar a distancia, de forma tangible, por voz, por gestos, etc. Esta decisión se encuentra también en la capa de diseño, pero condiciona la implementación, pues si se quiere, por ejemplo, realizar la interacción a distancia, ya no se pueden usar técnicas locales.

Ejemplo: “Quiero realizar el lanzamiento sin desplazarme físicamente ni virtualmente” - “Debes implementar técnicas de selección a distancia (por ejemplo, *raycasting* para interacción directa o menú virtual para interacción indirecta)”.

Preferencias define las necesidades concretas que los desarrolladores pueden tener en el contexto de uso de la aplicación. Por ejemplo, a la hora de implementar un sistema de navegación se puede querer usar el que produzca mayor sensación de inmersión, el que produzca menor sensación de mareo, etc. Esta decisión se encuentra en la capa de implementación, pues estas necesidades se cubrirán dependiendo de la implementación concreta utilizada para un tipo de interacción.

Ejemplo: “Quiero que el lanzamiento sea realista” - “Debes realizar las acciones de manipulación (desplazamiento y rotación de la pelota) de forma local y directa para emular el lanzamiento de la pelota con el mando”.

VI. Validación del modelo propuesto

Con el fin de poner a prueba el uso del modelo propuesto, se ha implementado una aplicación, tanto en realidad virtual como en realidad aumentada, que permite al usuario elegir qué tipo de interacción desea realizar para una acción simple (por ejemplo, seleccionar y rotar un objeto virtual). De esta manera, se pueden obtener dos corrientes de *feedback*:

1. Validar el modelo propuesto. Al poner a prueba el modelo propuesto con usuarios reales (desarrolladores) se puede corroborar la correctitud y la facilidad de uso del modelo propuesto, pues es imperante que los desarrolladores encuentren la clasificación de las técnicas intuitiva y fácil de comprender.
2. La usabilidad de las técnicas. Como resultado paralelo, se puede estudiar la usabilidad de cada tipo de técnica para conocer más detalles y poder ofrecer más información a los desarrolladores acerca de los puntos fuertes y débiles de las mismas. Estos datos se podrán contrastar y complementar con los de otros estudios, como los realizados en [3], tal y como se mencionó en la sección III.

La implementación de ambas aplicaciones se ha hecho utilizando el motor de videojuegos Unity [13]. La elección

de este motor de videojuegos es debido a su facilidad de uso (tanto en curva de aprendizaje como por su licencia gratuita) y su compatibilidad con otros *frameworks* necesarios para el desarrollo de este proyecto.

Para la aplicación de realidad virtual se utiliza el *framework* SteamVR [14], ya que permite compatibilidad con distintos dispositivos *hardware*, es gratuito y, además, ofrece multitud de proyectos y herramientas de ejemplo para ayudar a los desarrolladores a empezar a implementar aplicaciones de realidad virtual con Unity. Se han implementado un total de 7 tipos de navegación (movimiento libre normal y alternativo, teletransporte libre y de nodos, dash, movmiento de brazos y escalada), ya que la navegación en entornos de realidad virtual suele dar más problemas, por lo que resulta de mayor importancia en el estudio.

Para la aplicación de realidad aumentada se ha utilizado el *framework* de Vuforia [15], ya que viene integrado de forma nativa en las últimas versiones de Unity, es compatible con dispositivos móviles Android e iOS, es gratuito y cuenta con tutoriales que permiten a cualquier usuario comenzar a desarrollar sus aplicaciones de forma rápida y sencilla. Se implementaron técnicas de interacción tangibles, directas e indirectas, e intangibles, directas e indirectas.

VII. Conclusiones y trabajo futuro

Tal y como se ha expuesto en la sección II, ha quedado patente el problema existente para los desarrolladores de aplicaciones de realidad virtual y realidad aumentada para implementar correctamente técnicas de interacción que resulten agradables a los usuarios.

Con el objetivo de dar una solución a este problema, se ha propuesto un modelo de interacción más abstracto, que sirve tanto para las interacciones en entornos de realidad virtual como realidad aumentada. Dicho modelo va acompañado de la implementación en ambos entornos de distintas técnicas de interacción para poder evaluar y corroborar la usabilidad del mismo, así como la usabilidad de las distintas técnicas, con el fin de poder brindar a los desarrolladores una guía de qué tipo de interacción deben usar en sus aplicaciones, atendiendo a diferentes criterios.

Las aplicaciones se pusieron a prueba con usuarios reales, donde éstos valoraron que la mejor técnica de navegación en realidad virtual era la escalada, en gran parte debido a su similitud por el movimiento real de los brazos al escalar. Por otra parte, en la aplicación de realidad aumentada, la técnica mejor valorada fue la tangible directa, debido a que es una forma de interacción que está ampliamente normalizada en nuestra sociedad.

Como trabajo futuro, se plantea el proceso de puesta a prueba del modelo con desarrolladores reales para comprobar la facilidad de uso del modelo. Asimismo, se plantea el proceso de ampliar las aplicaciones de prueba para testear más técnicas de interacción con usuarios y poder completar la guía de diseño para desarrolladores.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Economía y Competitividad de España dentro del proyecto DISPERSA (TIN2015-67149-C3-3-R).

Referencias

[1] M. Billinghurst, H. Kato, and S. Myojin, "Advanced Interaction Techniques for Augmented Reality Applications," *LNCS*, vol. 5622, pp. 13–22, 2009.

[2] J. Jung, H. Park, D. Hwang, M. Son, D. Beck, J. Park, and W. Park, "A Review on Interaction Techniques in Virtual Environments," in *2014 International Conference on Industrial Engineering and Operations Management*, 2014, pp. 1582–1590.

[3] L. Ahmed, S. Hamdy, D. Hegazy, and T. El-Arif, "Interaction techniques in mobile augmented reality: State-of-the-art," in *2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)*, Dec 2015, pp. 424–433.

[4] M. R. Mine, "Virtual environment interaction techniques," Chapel Hill, NC, USA, Tech. Rep., 1995.

[5] D. A. Bowman, "Interaction techniques for immersive virtual environments: Design, evaluation, andapplication," *Journal of Visual Languages and Computing*, vol. 10, pp. 37–53, 1998.

[6] M. J. Habgood, D. Wilson, D. Moore, and S. Alapont, "Hci lessons from playstation vr," in *Extended Abstracts Publication of the Annual Symposium on Computer-Human Interaction in Play*, ser. CHI PLAY '17 Extended Abstracts. New York, NY, USA: ACM, 2017, pp. 125–135.

[7] T. T. Elvins, "Wayfinding 2: The lost world," *SIGGRAPH Comput. Graph.*, vol. 31, no. 4, pp. 9–12, Nov. 1997.

[8] S. Volbracht and G. Domik, "Developing Effective Navigation Techniques in Virtual 3D-Environments," *Proceedings of the Eurographics Workshop*, pp. 55–64, 2000.

[9] B. Krieg-Brückner, T. Röfer, H. Carmesin, and R. Müller, "A taxonomy of spatial knowledge for navigation and its application to the bremen autonomous wheelchair," in *Spatial Cognition, An Interdisciplinary Approach to Representing and Processing Spatial Knowledge*, 1998, pp. 373–398.

[10] E. Nevala, "Design guide for room scale vr," https://www.gamedev.net/articles/game-design/game-design-and-theory/design-guide-for-room-scale-vr-r4355/, accedido: junio de 2018.

[11] P. Locomoto, "Lean walk," https://www.realovirtual.com/noticias/5326/project-locomoto-presenta-su-sistema-movimiento-inclinacion, accedido: junio de 2018.

[12] J. C. S. Cardoso, "Comparison of gesture, gamepad, and gaze-based locomotion for vr worlds," in *Proceedings of the 22Nd ACM Conference on Virtual Reality Software and Technology*, ser. VRST '16. New York, NY, USA: ACM, 2016, pp. 319–320.

[13] Unity, https://unity3d.com, accedido: junio de 2018.

[14] Steam, https://steamcommunity.com/steamvr, accedido: junio de 2018.

[15] Vuforia, https://www.vuforia.com/, accedido: junio de 2018.