Componente de ajuste de dificultad en un videojuego utilizando tablas de decisión adaptativas (9 de octubre de 2015)

K. Barrios y C. Zapata, *Member IEEE*

Abstract— While a lot of dynamic difficulty components in videogames act based upon the results of players actions and not the actions themselves, in this paper we try a different approach using information from every player action in every different situation. This component learns the player's reaction to every possible situation of: environment, enemy placement, enemy actions, and other variables, and use that to predict their next action so the enemies can act according to it. This is not a switch of difficulty settings, but a definition of the best enemy behavior that changes in real time. We propose using Adaptive Decision Tables to control the enemies' actions so their behavior can change according to the situation.

Keywords — adaptive table, videogames.

I. INTRODUCCIÓN

Durante la creación de videojuegos (definidos por Ralph Baer[1] como "aparatos [...] con el propósito de jugar, entrenar con simulaciones y participar de otras actividades") se considera que los jugadores son diferentes y, por lo tanto, no pueden disfrutarlos de la misma forma unos y otros. Según Csikszentmihalyi [2], debe haber un balance entre dificultad y habilidad del jugador para que éste pueda divertirse con el juego; mientras que Hunnicke [3] dice que los videojuegos son aburridos cuando son muy fáciles y frustrantes cuando son muy difíciles, casos en los que desaparece su fin: entretener.

Es por esto que existen diferentes formas para ajustar el reto a cada jugador, como mostrar el puntaje para que el jugador tenga un objetivo al tratar de mejorarlo, o dar la libertad de elegir la dificultad, que cambia algunas características predefinidas que aumentan o disminuyen el reto.

Una forma usada últimamente para que cada jugador tenga un reto adecuado es la modificación automática de la dificultad según la forma en que se juega. Muhammad Iftekher Chowdhury [4] menciona que "si el nivel de dificultad ajustado dinámicamente en un videojuego se iguala apropiadamente a la habilidad del jugador, entonces no solo atraerá a jugadores de diferentes demografías, sino que también hará que el jugador juegue el videojuego repetidas veces sin aburrirse".

Se identifica como un problema cambiar diferentes características de un videojuego de manera eficiente al detectar las habilidades del jugador. Y con la particularidad de

que lo que se cambie sea el comportamiento de cada enemigo o personaje que no maneje el jugador (NPC o "non-player character").

Policarpo y Urbano[5] analizan diferentes reglas para el comportamiento de los NPC y concluyeron que sí es posible que una IA dinámica aprenda tácticas que se aprovechen de las debilidades en oponentes que usen IA estática. Huang [6] demuestra que, para un jugador con una determinada habilidad, el ajuste de dificultad dinámica en Inteligencia Computacional tanto en tiempo real como en ANN (Redes Neuronales) puede generar un oponente con un reto apropiado para satisfacer al jugador.

El cambiar el comportamiento de cada NPC es el problema que se usará para este proyecto. Esto implica tomar la información de las acciones o habilidades del jugador y de qué tan bueno es cumpliendo el objetivo del juego, para usar dicha información en la modificación de comportamiento. No importará si el jugador es bueno o es malo, el componente desarrollado identificará su forma de jugar, sus decisiones en cada situación y las acciones involuntarias que realice, aunque sea por razones imperceptibles debido al *factor humano*, y modificará el comportamiento de los oponentes para que se adecúen a dicho jugador, siendo completamente diferentes para cada persona.

A. Problema Seleccionado:

¿Cómo se puede hacer un algoritmo que modifique el comportamiento de los oponentes basándose en información de cada acción del jugador?

El problema se desarrollaría definiendo una forma para tomar la información del jugador y aplicarla al comportamiento de los enemigos.

Se buscaría analizar de forma profunda cada acción de los jugadores y modificar el comportamiento de los NPC según sus costumbres, habilidades y fallos. Cada movimiento que el jugador hace debería afectar directamente el comportamiento de los NPC.

II. OBJETIVOS DEL TRABAJO

El objetivo de este trabajo es implementar un componente de ajuste de dificultad que altere el comportamiento de los

oponentes en respuesta a los patrones de comportamiento del jugador utilizando tablas de decisión adaptativa.

A. RESULTADOS ESPERADOS

Se busca diseñar un componente de aprendizaje de las acciones del jugador lo que permitirá obtener:

1. Componente que identifica situación actual: estado actual del personaje (sobre el piso, saltando, cayendo), la ubicación respecto al entorno (cercanía de paredes, techo, piso, abismos respecto al jugador) y la situación de los enemigos que rodean al jugador (cantidad, posiciones y movimientos).
2. Componente que actualiza la lista de posibilidades de cada acción (saltar, correr, atacar, etc.) que el jugador haría en cada situación. Se debe tener también una lista de todas las situaciones posibles.

Una vez identificadas las acciones del jugador se diseñará un componente que controle el comportamiento de los NPC con tablas de decisión adaptativas para que realicen ataques contra el jugador según las probabilidades de su próximo movimiento. En consecuencia se obtendrá:

3. Reglas para la tabla de decisión adaptativa (reglas adaptativas y no adaptativas).
4. Componente de priorización de reglas (para evitar sobrecarga de proceso).

Para validar que lo obtenido logra el objetivo buscado se implementará el componente de aprendizaje y los patrones de comportamiento en un videojuego 2D de acción de vista lateral, para lo cual se realizarán las siguientes acciones:

5. Desarrollo de un videojuego de acción de vista lateral. Para implementar los componentes.
6. Implementación del componente de aprendizaje y de la Tabla de Decisión adaptativa. Se implementan los primeros cuatro objetivos específicos en el videojuego desarrollado en el quinto resultado específico.

III. MARCO CONCEPTUAL

A. Videojuego

Según la real academia española es “un dispositivo electrónico que permite, mediante mandos apropiados, simular juegos en las pantallas de un televisor o de un ordenador” [7]. Hunicke [3] dice que los videojuegos son aburridos cuando son muy fáciles y frustrantes cuando son muy difíciles. Cuando una o más personas controlan parámetros o personajes en un videojuego, se les considera jugadores. El jugador es quien “disfrutará” del videojuego al jugarlo.

El género de videojuego en el que se implementará la alternativa de solución es un juego de acción en 2D. Lo que vendría a ser un personaje que se mueve en un plano de dos dimensiones donde puede caminar, saltar y atacar, donde se encontrará con oponentes (NPC) con las mismas características y se dará en un escenario con superficies a diferentes niveles de altura

B. NPC

Van Horn [8] indica que NPC son las siglas de “non-player character” o “personaje que no es jugador”. Es un personaje de un videojuego con acciones que no dependen del control directo del jugador. Policarpo y Urbano [5] crean

comportamientos basados en reglas para los NPC, estos muchas veces siguen acciones predefinidas o se usa algoritmos de inteligencia artificial para determinar su proceder en las diferentes situaciones del juego. Chin Hyong Tan [9] dice que los videojuegos involucran interacción del jugador tanto con el entorno como con los NPC.

El término NPC, usualmente, no se limita a oponentes, pero en este proyecto de fin de carrera así se les denominará.

C. Dificultad

La definición de la real academia española dice que dificultad es “embarazo, inconveniente, oposición o contrariedad que impide conseguir, ejecutar o entender bien algo y pronto” [7]. Csikszentmihalyi [2] dice que debe haber un balance entre dificultad y habilidad del jugador para que éste pueda disfrutar del juego.

Muhammad Iftekhher Chowdhury [4] indica que la Dificultad dinámica automática es la técnica automática por la cual se cambia la elevación de dificultad para un videojuego en tiempo real (sin dejar de jugar) para que iguale las habilidades del jugador. Agrega también que implementarla es caro y consume mucho tiempo, por lo que su utilidad está limitada. Pederassi Lomba de Araujo y Deijó [10] indican que el uso de Dificultad dinámica mantiene al jugador constantemente motivado.

D. Unity

Unity es un Game Engine (de uso similar a un framework) que se caracteriza por tener muchas herramientas de fácil uso para la creación de videojuegos, además de que se puede usar en una gran cantidad de plataformas [11].

E. Probabilidad de acción futura.

Se determina la acción futura del jugador según probabilidades que se alteran con cada nueva acción que este efectúa en cada situación diferente.

F. Tablas de Decisión

Definen un comportamiento a tomar según una lista de reglas. Cada regla tiene ciertas condiciones que dan un resultado diferente

G. Tablas de Decisión Adaptativas

Las Tablas de Decisión Adaptativas tienen reglas adicionales en la capa adaptativa, que indican el cambio de los resultados de las reglas de la capa anterior. Neto dice que esta es la mejor alternativa entre las tecnologías adaptativas para usar en el problema propuesto[12].

En este proyecto desarrollará un método de ajuste del comportamiento de los NPC de forma dinámica en un videojuego de acción con vista lateral. Videojuego y dificultad son los conceptos básicos que se usan para el entendimiento del componente. NPC son los personajes de los cuales el componente modificará su comportamiento. Dificultad ajustada automáticamente es el concepto que refleja de qué forma el algoritmo modificará el comportamiento de los NPC.

IV. ESTADO DEL ARTE

Desde hace años se han definido diversas formas de aplicar el ajuste de dificultad en los videojuegos. Desde formas simples

como alterar el desempeño de los NPC según los logros del jugador, hasta formas más complejas como delimitar el rango de habilidad del jugador según varios factores y con eso cambiar parámetros del juego.

La revisión del estado del arte busca analizar cómo son en la actualidad las diferentes propuestas de ajustes de dificultad en los videojuegos, tanto las usadas en juegos comerciales, como las que no han sido aplicadas.

A. Método usado en la revisión del estado del arte

Se ha usado el método tradicional para la revisión del estado del arte. Se realizaron búsquedas en las bases de datos de artículos IEEE y Research Gate, para analizar diferentes alternativas de solución hay para el mismo problema. La búsqueda se realizó la primera semana de abril del 2015.

B. Formas de usar ajustes de dificultad:

1) Personajes que aprenden la mejor estrategia para cada situación gracias a comportamientos de reglas.

Policarpo y Urbano [5] definieron comportamientos definidos por reglas que logran que la IA de personajes aprenda la mejor estrategia para cada situación. Cada oponente tiene una lista de reglas, de las cuales se coloca las de mayor “peso” en el script de su comportamiento. Con cada interacción con el jugador se le aumenta el peso a las reglas que, al ejecutarse, tuvieron un resultado exitoso.

2) Algoritmos evolucionarios de multiple objetivos.

Van Hoorn [8] definió algoritmos evolucionarios con multiobjetivos relacionados a mejorar el desempeño de juego y lograr mayor similitud con el comportamiento humano. Indica también que hay varios usos para la imitación del jugador en diferentes tipos de videojuegos y que hay la necesidad de encontrar modelos que sean robustos y también creíbles.

3) Graduación de dificultad de IA mediante algoritmos de RL y EC.

Chin Hiong Tan [9] usa Reinforcement Learning y Computación Evolutiva para mejorar la satisfacción del jugador al aumentar la dificultad de la IA del juego mientras se juega. Los efectos de cambiar las escalas de aprendizaje (Learning) y mutación se examinan y se propone una regla para los parámetros. Se presenta dos algoritmos para mejorar la satisfacción del jugador, un Adaptive Uni-chromosome y un Adaptive Duo-chromosome. Se varió la tasa de aprendizaje y de mutación de ambos algoritmos para lograr una regla general. Los algoritmos propuestos demuestran la capacidad de igualar a sus oponentes en puntajes promedio y porcentaje de victorias. Ambos algoritmos pueden generalizar una variedad de oponentes.

4) Modificación masiva del comportamiento de cientos de personajes en simultáneo.

Maggiore [13] define planes (conjunto de acciones consecutivas futuras) y un algoritmo de búsqueda de caminos en grafos (cada nodo es una acción y se busca el mejor conjunto de acciones, o mejor plan), con lo que modifica el comportamiento en tiempo real de cientos de personajes, de forma diferente para cada uno, sin requerir información previa. Se usó una técnica de planificación por capas para crear NPCs que no solo reaccionen a estímulos externos, logrando un mundo más creíble. Con un encadenamiento de

acciones en un árbol de nodos se forman planes diferentes, junto a heurísticas para determinar los planes parciales menos efectivos e ignorarlos, además para memorizar y reutilizar planes pasados que hayan resultado efectivos. También toma en cuenta el tiempo que tomará, para que no empiece muy tarde. Es un planificador adaptativo, ya que las acciones que toma y los escenarios no están "hardcodeados".

5) Ajuste de dificultad dinámica al alterar comportamiento de oponentes con CI, MCTS y UCT.

Huang [9] define que se puede lograr un ajuste en la dificultad dinámica al cambiar el reto que presentan los oponentes mediante Inteligencia Computacional, al incluir el Árbol de Búsqueda de Monte Carlo (MCTS) y la Upper Confidence bound for Trees (UCT).

6) Ajuste de dificultad según desempeño y preguntas.

Holtkamp [14] hace que la dificultad se ajuste según características del jugador definidas por las respuestas a nueve preguntas y por su desempeño al jugar.

El estado del arte muestra el éxito y problemas que han tenido diferentes alternativas de solución similares a la presentada en este proyecto en el que se presenta una alternativa para solucionar los problemas de toma de información y modificación de comportamiento encontrados en la revisión del estado del arte, al enfocarse detalladamente en que cada movimiento del jugador defina un ajuste en el comportamiento de los NPC; ya que, al tomar en cuenta cada una de las acciones del jugador, se obtiene datos de una forma tan detallada que refleja sus costumbres al jugar.

V. DISEÑO E IMPLEMENTACIÓN DE LA SOLUCIÓN

A. Aprendizaje

En esta sección se explica la etapa de aprendizaje del componente, que comprende el levantamiento de la información y la forma en la que se utiliza. Además se define cómo esta será estructurada.

Existen varias formas de tomar los datos del jugador, los cuales son esenciales debido a que es la información que se usará. Lo que se busca es un formato o método de extraer los datos que no afecte el desempeño del juego (por ejemplo, que por sobrecarga de proceso al tomar los datos, el juego no sea fluido), pero que, aun así, obtenga la información necesaria para la metodología de ajuste de dificultad elegida.

Por ejemplo, Holtkamp [14] agregó a la toma de datos de desempeño durante el juego una serie de preguntas al jugador, que definían algunos parámetros de su personalidad; Policarpo y Urbano [5] analizan cuál de las múltiples reglas del comportamiento de cada NPC, en este caso un oponente, es más efectiva al enfrentarse al jugador y le dan prioridades diferentes en el script (archivo de procesamiento por lotes) dinámico que determina su comportamiento.

El problema radica en que estos métodos toman datos superficiales del jugador. No detallan lo que busca hacer el jugador, sino lo que obtienen, reduciendo así la variedad de la información tomada. Esto es porque varios jugadores diferentes podrían tener el mismo resultado jugando diferente y teniendo diferentes objetivos (un jugador diestro y uno que se aprovecha de un error de programación podrían conseguir ambos el puntaje máximo).

El levantamiento de información se hará analizando la forma en la que el jugador juega, identificando primero la situación en la que se encuentra y después la acción que ejecuta en esta. El aprendizaje se ejecuta de forma continua y, tras varias sesiones de juego, se define la acción más usada para cada situación, lo que a su vez son probabilidades para predicciones futuras, siendo este el punto de partida para la modificación del comportamiento de los enemigos.

1) Identificación de la Situación

Tiene que identificarse la situación actual según una serie de características del personaje, los NPC y el entorno. Esto servirá para relacionar cada acción del jugador a una situación particular.

2) Flags:

Para las características se usarán flags (definidos en la Tabla 1), que cambiarán según se está jugando. Cada vez que el jugador haga una acción se revisan los flags, se determina que esa acción la hace el jugador para ese conjunto de flags y se procede a modificar las probabilidades de cada acción para cada situación. La Situación está determinada por los valores del conjunto de flags.

Tabla 1. Flags que determinan situación

#	Flag
1	Personaje está en el piso (no en el aire).
2	Personaje está cerca de pared izquierda.
3	Personaje está cerca de pared derecha.
4	Personaje está caminando a la izquierda.
5	Personaje está caminando a la derecha.
6	El techo está cerca al piso.
7	Hay caída a la izquierda.
8	Hay caída a la derecha.
9	Hay un NPC cercano al personaje a la izquierda.
10	Hay dos o más NPC cercanos al personaje a la izquierda.
11	Hay un NPC cercano al personaje a la derecha.
12	Hay dos o más NPC cercanos al personaje a la derecha.

3) Acciones:

En cada Situación, el jugador puede hacer cualquiera de las acciones que desee del personaje. En la Tabla 6 vemos las acciones que puede realizar.

Tabla 2. Acciones que puede hacer el jugador

#	Acción
1	Caminar a la izquierda
2	Caminar a la derecha
3	Saltar
4	Saltar a la izquierda
5	Saltar a la derecha
6	Ataque 1 (hacia adelante)
7	Ataque 2 (hacia arriba)
8	Ataque 3 (proyectil a distancia)

4) Modificación de probabilidades

Para cada una de las Situaciones, hay una lista ordenada de todas las acciones del jugador. Cada vez que el jugador hace una acción en determinada Situación, dicha acción sube una

posición en la lista correspondiente a la Situación en que se ejecutó. Se toma la acción en primera posición de cada lista como la acción que tiene más probabilidad que ejecute el jugador en cada situación.

Para cada situación, el jugador puede hacer cualquiera de las acciones, pero la que está en primer lugar en la lista de acciones probables para dicha situación es la que tiene mayor probabilidad de realización.

En la Tabla 7 podemos ver cómo cambia el orden de la lista de probabilidad de acciones para una misma situación.

Tabla 3. Cambio de prioridad de acciones para cada Situación

Situación (set de Flags)	Acciones (de mayor a menor prioridad)							
A (flags: 1-3-4)	1	2	3	4	5	6	7	8
Para la situación A hace la acción 3								
A (flags: 1-3-4)	1	3	2	4	5	6	7	8
Para la situación A hace la acción 4								
A (flags: 1-3-4)	1	3	4	2	5	6	7	8
Para la situación A hace la acción 3								
A (flags: 1-3-4)	3	1	4	2	5	6	7	8

B. Tabla de Decisión Adaptativa

Los métodos de ajuste de dificultad usan información para definir el comportamiento de los enemigos. La información puede obtenerse del jugador, de la efectividad de acciones pasadas de los enemigos, deduciendo la mejor elección para la situación actual, etc. Y puede ser procesada de diferentes formas, ya sea usando algún algoritmo de inteligencia artificial, búsqueda de caminos en árbol de nodos, etc.

Chin Hiong Tan [9] usa Computación Evolutiva para, mediante ensayo y error, definir una regla general con la que modifica los parámetros de los algoritmos. Maggiore [13] usa una búsqueda en un árbol de nodos para encontrar la mejor secuencia de acciones de los NPC.

El problema radica en que las diversas formas de procesar la información y aplicarla en el comportamiento de enemigos no tiene una “mejor forma” de hacerlo. Debe hacerse de una forma que sea óptima dependiendo de: el origen de la información usada, cómo se ha procesado y el género del videojuego; considerando en qué características se quiere ser más eficiente como el requerimiento de proceso, la velocidad de aplicación, máximo de enemigos simultáneos, frecuencia de la toma de datos, etc.

Se decidió usar Tablas de Decisión Adaptativas en vez que otras metodologías porque, según Neto[12], permiten definir reglas (ideal para el comportamiento de los NPC) y cambiarlas constantemente (ideal para la modificación de su comportamiento). Se evaluaron varias alternativas pero se optó por Tablas de Decisión Adaptativas.

Con las Tablas de Decisión Adaptativas se logra que el comportamiento de los NPC cambie mientras se juega continuamente.

1) Reglas para la tabla de decisión adaptativa

Hay dos listas de Reglas de la Tabla de Decisión Adaptativa: las reglas de la capa no adaptativa y las reglas de la capa

adaptativa. Las reglas de la capa no adaptativa son una lista de estímulos y sus respectivas respuestas. Las reglas de la capa adaptativa son otra lista de estímulos y sus respuestas, pero estas respuestas son todas las combinaciones de estímulo y respuesta de las reglas de la capa no adaptativa, y los estímulos señalan que estímulo con respuesta *actualizar* en la capa no adaptativa.

Definiremos a la cantidad de Situaciones como N y la cantidad de acciones de los NPC como M. Por esto la cantidad de reglas de la capa no adaptativa es N y la cantidad de reglas de la capa adaptativa es NxM. Además, un Contexto es la suma de una Situación y la Acción del jugador más alta en la lista de posibilidades para dicha Situación.

En la Tabla 8 puede verse un ejemplo de las reglas de la capa adaptativa y de las reglas de la capa no adaptativa.

Tabla 4. Ejemplo de reglas de Tabla de Decisión Adaptativa

Capa no adaptativa		Capa Adaptativa	
TAMAÑO = N		TAMAÑO = M X N	
Condicion A	Accion 1	Condicion A	Accion 1
Condicion B	Accion 3	Condicion A	Accion 2
Condicion C	Accion 2	Condicion A	Accion 3
Condicion D	Accion 3	Condicion B	Accion 1
		Condicion B	Accion 2
		Condicion B	Accion 3
		Condicion C	Accion 1
		Condicion C	Accion 2
		Condicion C	Accion 3
		Condicion D	Accion 1
		Condicion D	Accion 2
		Condicion D	Accion 3

Cada regla de la capa no adaptativa corresponde a un Contexto y la acción correspondiente que debe ejecutar el NPC. Hay un Contexto por cada Situación.

Las reglas de la capa adaptativa son la combinación de todos los contextos con todas las acciones que pueden hacer los NPC. Las reglas de la capa determinan el cambio de acción del NPC correspondiente a cada Contexto en las reglas de la capa no adaptativa.

2) *Componente de priorización de reglas*

Debido a la cantidad de Flags, acciones del jugador y de los NPC, se puede ver que tienen un gran tamaño la cantidad de Situaciones, Contextos, Reglas no adaptativas y, en una considerable mayor medida, Reglas adaptativas. Esto podría hacer que el rendimiento de la alternativa de solución para este problema disminuya por el alto y constante requerimiento de proceso.

Para disminuir esta carga de proceso, se ha tomado en cuenta, únicamente, a las reglas correspondientes a una cantidad reducida de Contextos. Los Contextos elegidos también pueden cambiar, según los que se hayan dado más veces en las

últimas situaciones. Pero, este cambio de contextos elegidos no mantendrá la frecuencia de uso de la Tabla de Decisiones Adaptativa, porque sino la carga de proceso no disminuirá. Por ejemplo, si la Tabla de Decisiones Adaptativa es consultada con cada acción del personaje o de un NPC, el cambio de Contextos elegidos se podría hacer con un hilo separado cada medio minuto, o cada vez que se gana o pierde en un nivel.

VI. CONCLUSIONES

Mantener el reto en el jugador es una de las formas de hacer interesante un juego. Este factor se puede obtener de diferentes maneras: una historia atractiva, objetivos complejos, distintos caminos para lograr los mismos objetivos, entre otros.

Este trabajo no sólo gira alrededor de a IA que se debe desarrollar para contar con enemigos que reacciones a las diversas acciones del jugador. Es muy importante contar con una adecuada toma de información sobre las acciones del jugador y estructurarlas de tal forma que afecten mínimamente el desempeño del juego.

El uso de tablas de decisión adaptativas se ajustan muy bien al proyecto pero se deben tomar ciertas decisiones sobre cómo aplicarlas para reducir el impacto en el jugador y el juego.

Dado que este proyecto aún no ha concluido no podemos dar cifras objetivas sobre el trabajo.

VII. TRABAJOS FUTUROS

Existen varios trabajos futuros que se pueden generar de este proyecto:

1) *Otros géneros*

Adaptar la alternativa de solución para videojuegos de otros géneros, siendo los más fáciles de adaptar los videojuegos de lucha y los videojuegos de acción 3D.

2) *Estandarizar dificultad*

Analizar la probabilidad de victoria para cada acción del jugador en cada situación, y que se modifiquen diversas características del videojuego para mantener constante esta probabilidad en todas las situaciones, así se lograría mantener una dificultad estándar que depende del grado de habilidad del jugador.

3) *Machine Learning en predicciones*

Utilizar Machine Learning para las predicciones. En vez de que cada nueva acción del jugador cambie la lista de posibilidades, se acumularía la información y se utilizaría Machine Learning para la predicción del comportamiento futuro del jugador, debido a que son grandes cantidades de información las que se habrán tomado de todas las acciones previas, esa es la utilidad de Machine Learning según Kohavi [15]. La metodología a usar sería Feature Selection, donde se descartan las características que menor importancia tienen; se usaría para procesar solo la información relevante [16].

4) *Reemplazos a autómatas adaptativos*

Usar metodologías diferentes para la modificación de comportamiento de NPC: Autómatas no adaptativos, Tablas adaptativas de decisiones (modificando secciones de la tabla), Redes neuronales (con un set de datos iniciales para que haya una dificultad estándar y sobre eso modificarlos personalmente con cada jugador con métodos de aprendizaje)

o Sistemas Expertos (tomar el juicio de un experto como base para determinar la dificultad ideal para cada jugador).

VIII. REFERENCIAS

- [1] R. H. Baer, W. T. Rusch y W. L. Harrison, «TELEVISION GAMING APPARATUS AND METHOD». USA/NH Patente United States Patent 3659285, 25 4 1972.
- [2] M. Csikszentmihalyi, *Flow : the psychology of optimal experience*, New York: Harper & Row, 1990.
- [3] R. Hunicke y V. Chapman, «AI for Dynamic Difficulty Adjustment in Games,» Northwestern University, Computer Science Department, Evanston, IL, 2004.
- [4] M. I. Chowdhury y M. Katchabaw, «Bringing Auto Dynamic Difficulty to Commercial Games: A Reusable Design Pattern Based Approach,» de *CGAMES 2013, The 18th International Conference on Computer Games*, London, Ontario, Canada, 2013.
- [5] D. Policarpo y P. Urbano, «Dynamic Scripting Applied to a First-Person Shooter,» Laboratório de Modelação de Agentes, FCUL, Lisboa, Portugal, 2010.
- [6] W. Huang, S. He, D. Chang y Y. Hao, «Dynamic Difficulty Adjustment Realization Based on Adaptive Neuro-controlled Game Opponent,» de *Third International Workshop on Advanced Computational Intelligence*, Suzhou, Jiangsu, China, 2010.
- [7] R. A. Española, *Diccionario de la lengua española*, Madrid, 2012.
- [8] N. van Hoorn, J. Togelius, D. Wierstra y J. Schmidhuber, «Robust player imitation using multiobjective evolution,» Dalle Molle Institute for Artificial Intelligence (IDSIA), Manno-Lugano, Switzerland, 2009.
- [9] C. H. Tan, K. C. Tan y A. Tay, «Dynamic Game Difficulty Scaling Using Adaptive Behavior-Based AI,» *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 3, nº 4, pp. 289 - 301, 2011.
- [10] B. B. Pederassi Lomba de Araujo y B. Feijó, «Evaluating dynamic difficulty adaptivity in shoot'em up games,» de *SBC - Proceedings of SBGames 2013*, Río de Janeiro, 2013.
- [11] «Unity - Game Engine,» [En línea]. Available: unity3d.com.
- [12] J. J. Neto, Interviewee, *Uso de Tecnologías Adaptativas para el comportamiento dinámico de oponentes en videojuegos*. [Entrevista]. 7 9 2015.
- [13] G. Maggiore, C. Santos, D. Dini, F. Peters, H. Bouwknecht y P. Spronck, «LGOAP: adaptive layered planning for real-time videogames,» de *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, Niagara Falls, ON, 2013.
- [14] W. Holtkamp, P. Trevino, C. Yun y O. Johnson, «Dynamic Difficulty with Personality Influences,» de *GLS 8.0 (Games+Learning+Society)*, Madison, WI, 2012.
- [15] R. Kohavi y R. Provost, «Glossary of Terms Special Issue on Applications of Machine Learning and the Knowledge Discovery Process,» Kluwer Academic Publishers, Boston, Manufactured in The Netherlands, 1998. [En línea]. Available: <http://ai.stanford.edu/~ronnyk/glossary.html>. [Último acceso: 08 06 2015].
- [16] I. Guyon y A. Elisseeff, «An Introduction to Variable and Feature Selection,» Leslie Pack Kaelbling, 2003.