COMPARACIÓN DE RENDIMIENTO, IMPLEMENTACIÓN Y JUGABILIDAD ENTRE ARQUETIPOS DE INTELIGENCIA ARTIFICIAL DE TOMA DE DECISIÓN PARA VIDEOJUEGOS

JOAN SEBASTIAN DUARTE RODRIGUEZ

KONRAD LORENZ FUNDACIÓN UNIVERSITARIA
FACULTAD DE MATEMÁTICAS E INGENIERÍAS
PROGRAMA DE INGENIERÍA DE SISTEMAS
BOGOTÁ
2017

COMPARACIÓN DE RENDIMIENTO, IMPLEMENTACIÓN Y JUGABILIDAD ENTRE ARQUETIPOS DE INTELIGENCIA ARTIFICIAL DE TOMA DE DECISIÓN PARA VIDEOJUEGOS

JOAN SEBASTIAN DUARTE RODRIGUEZ
Código: 506131049

Trabajo de grado para optar por el título de Ingeniero de Sistemas

Director:
Jorge Eliecer Camargo Mendoza, PhD

KONRAD LORENZ FUNDACIÓN UNIVERSITARIA
FACULTAD DE MATEMÁTICAS E INGENIERÍAS
PROGRAMA DE INGENIERÍA DE SISTEMAS
BOGOTÁ
2017

Nota de aceptación:

_____
_____
_____
_____
_____
_____
_____

_____

Firma del presidente del jurado

_____

Firma del jurado

_____

Firma del jurado

Bogotá D.C, __ de _____ de 2017

IN MEMORIAM

A mi mentor, mi guía, mi maestro, mi abuelo y mi padre:
Ángel Humberto Rodríguez Riveros

DEDICATORIA

Este proyecto está dedicado a mi familia, amigos, compañeros y docentes que me han acompañado y apoyado durante toda la carrera, escuchándome, motivándome y sobre todo ayudándome a crecer como persona cada día más.

Para: Humberto, Blanca, Lucila, Iveth, Mauro, Nelly, Alexander, Angie, Patricia y Javier

AGRADECIMIENTOS

Quisiera agradecer especialmente al profesor Felipe Fagua Barrera por brindarme sus conocimientos en el desarrollo de videojuegos, ya que sin estos no habría logrado muchas de las cosas que me hacen sentir orgulloso de estudiar Ingeniería de Sistemas.

De igual manera deseo agradecerle al profesor Jorge Eliecer Camargo por su acompañamiento, enseñanzas y sobre todo por la pasión y compromiso con la Ingeniería.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE ILUSTRACIONES

1. FORMULACIÓN DEL PROBLEMA

1.1 DESCRIPCIÓN DEL PROBLEMA

El objetivo de los videojuegos es crear experiencias atractivas al usuario que logren inmersión y fomenten la re-jugabilidad. Para lograr esto el equipo desarrollador se enfoca en crear mundos con la cohesión suficiente para lograr que el usuario se sumerja por completo en el juego. En el desarrollo de videojuegos hay 5 factores importantes para lograr este objetivo; música, arte gráfico, narrativa, mecánicas de jugabilidad, y la inteligencia artificial. Estos 5 factores comparten un mismo recurso, el tiempo de procesamiento. Para no afectar la experiencia del usuario de manera negativa hay que implementar los algoritmos que solucionan los problemas de cada área de manera óptima.

La inteligencia artificial es parte fundamental de la experiencia de usuario ya que permite que cada sesión de juego sea única al implementar comportamientos creíbles

De acuerdo con lo anterior, en este trabajo se realizó una comparación entre arquetipos de inteligencia artificial para conocer cuál de ellos es el más apropiado para un jugador.

1.2 ÁREA DEL CONOCIMIENTO

Las áreas de conocimientos en las enfoca este proyecto son la inteligencia artificial y el desarrollo de videojuegos empleando esta última.

1.3 ALCANCES Y DELIMITACIONES

El proyecto se delimita a comparar la implementación de dos de los arquetipos de inteligencia artificial basados en comportamiento planteados por Ian Millington y John Funge en su libro Artificial Intelligence for Games. Los modelos implementados fueron los Árboles de Comportamiento y las Máquinas de Estados.

Cabe resaltar que en cada arquetipo de inteligencia artificial, los agentes que los implementan y el videojuego al que hacen parte, fueron implementados en el lenguaje de programación orientado a objetos C# y fueron desarrollados dentro del motor de juego Unity3D, por lo que los resultados de esta comparativa pueden variar en entornos y lenguajes diferentes usados para el desarrollo de videojuegos.

La muestra de los resultados de este estudio se realizó por medio de una matriz comparativa, la cual se encargó de mostrar los resultados de 4 parámetros de calificación predefinidos sobre un mismo agente utilizando diferentes arquetipos. Estos parámetros son:

el número de líneas de código necesarias para implementar cada arquetipo, el tiempo que toma cada arquetipo en cumplir la acción del agente y un puntaje promedio de percepción del jugador obtenido durante sesiones de juego controladas. Un ejemplo de esta matriz puede ser observado en la Tabla 1.

Nombre del Agente			Arquetipo 1	Arquetipo 2
Parámetros de calificación	1	Porcentaje de aceptación global percepción de usuario		
	2	Tiempo de ejecución calculado en segundos		
	3	Número de líneas de código		

Tabla 1. Plantilla de matriz comparativa de resultados

El método de recolección de datos diseñado dentro del videojuego, al igual que la plantilla de la encuesta que se utilizará para conocer la percepción del usuario, se encontrarán detalladas con sus respectivos anexos en la sección 5. Diseño Metodológico.

2. JUSTIFICACIÓN

Las diferencias de un algoritmo de implementación de inteligencia artificial basado en máquinas de estados varía drásticamente de un modelo estructurado por medio de árboles de comportamiento, por este motivo una comparación detallada que de una perspectiva clara de qué arquetipo de implementación de inteligencia artificial es la más efectiva para determinado proyecto según su rendimiento y facilidad de implementación podrá ayudar a mejorar los tiempos de desarrollo, justificar decisiones de implementación y optimizar el rendimiento general del producto final. De igual manera se busca que la investigación motive a los desarrolladores, pequeños o grandes, a buscar la correcta implementación de modelos de inteligencia artificial en sus juegos.

3. OBJETIVOS GENERALES Y ESPECÍFICOS

3.1 OBJETIVO GENERAL

Proponer una comparación entre arquetipos de inteligencia artificial basadas en comportamientos, que permitan identificar las características y escenarios en la que cada uno es más adecuado para el desarrollo de videojuegos.

3.2 OBJETIVOS ESPECÍFICOS

- Desarrollar un videojuego que contenga los agentes de inteligencia artificial necesarios donde se implementarán los arquetipos que se analizarán.
- Ejecutar pruebas de usuarios que permitan evaluar los agentes de Inteligencia Artificial.
- Realizar un análisis de facilidad de implementación de cada arquetipo utilizando el número de líneas de código, tiempo de ejecución y experiencia del usuario como parámetro de medición.

4. MARCO REFERENCIAL

4.1. MARCO HISTÓRICO

Desde el año 2015 se empezó el desarrollo de un juego bajo la orientación del semillero de videojuegos, este proyecto lleva por nombre hoy en día DronesVR y su principal objetivo en aquel momento era impulsar la participación de los estudiantes dentro del semillero. Fue en este dónde surgió la idea de comparar distintos arquetipos de inteligencia artificial, y de esta manera poder conocer la facilidad de empleo de cada uno, además de esto poder saber en qué escenarios es más factible su implementación

4.2. MARCO TEÓRICO

Este trabajo aplica conceptos de inteligencia artificial, su rol en el desarrollo de videojuegos y metodologías de implementación, como se muestra en la Figura 1. A continuación se realiza una abstracción del contexto y definición de los conceptos más relevantes para lograr una correcta apreciación de lo expuesto en el documento. Esta abstracción se dividirá en dos partes: componentes a desarrollar y conceptos de implementación del videojuego.

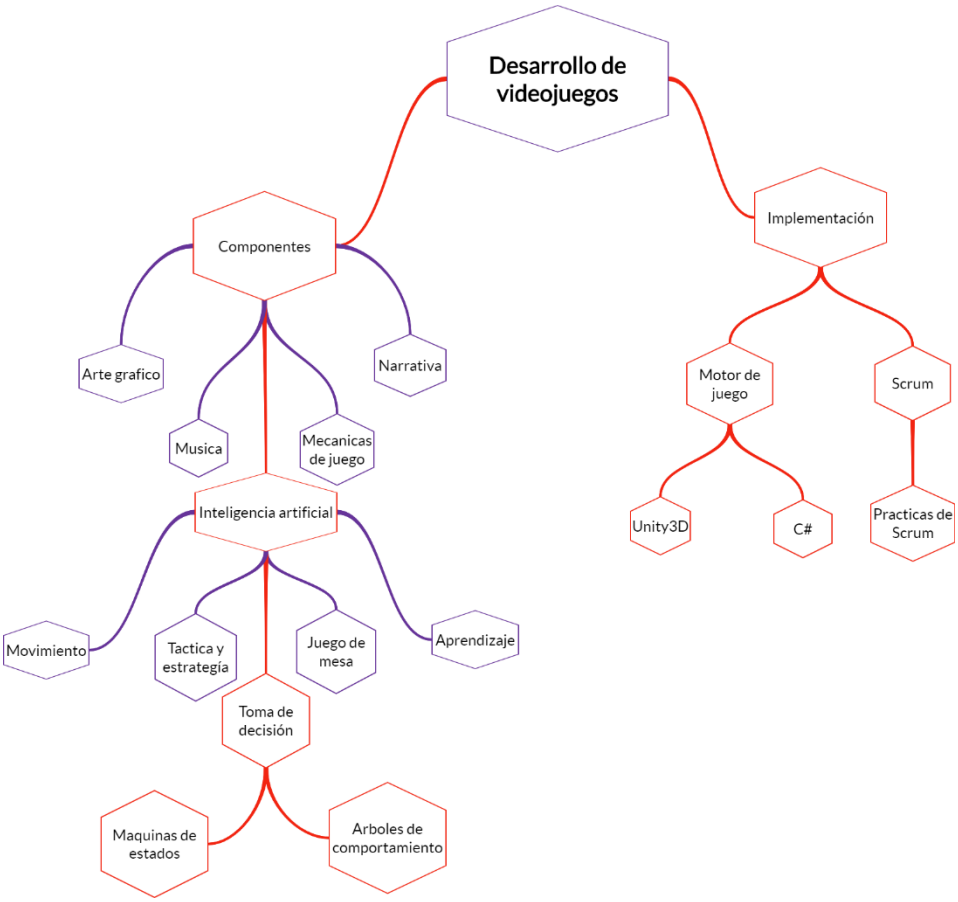

Figura 1. Mapa conceptual de temáticas a tratar

4.2.1. Componentes

Como se mencionó al inicio de este documento y se puede observar en la Figura 2 que los videojuegos están compuestos de 5 elementos principales que, en conjunto, cumplen el objetivo de sumergir a la persona en el juego, estos son: el arte gráfico, la narrativa, la música, las mecánicas de juego y finalmente la inteligencia artificial. A continuación, se explicará de manera detallada el componente que será objeto de estudio en este trabajo: la inteligencia artificial.

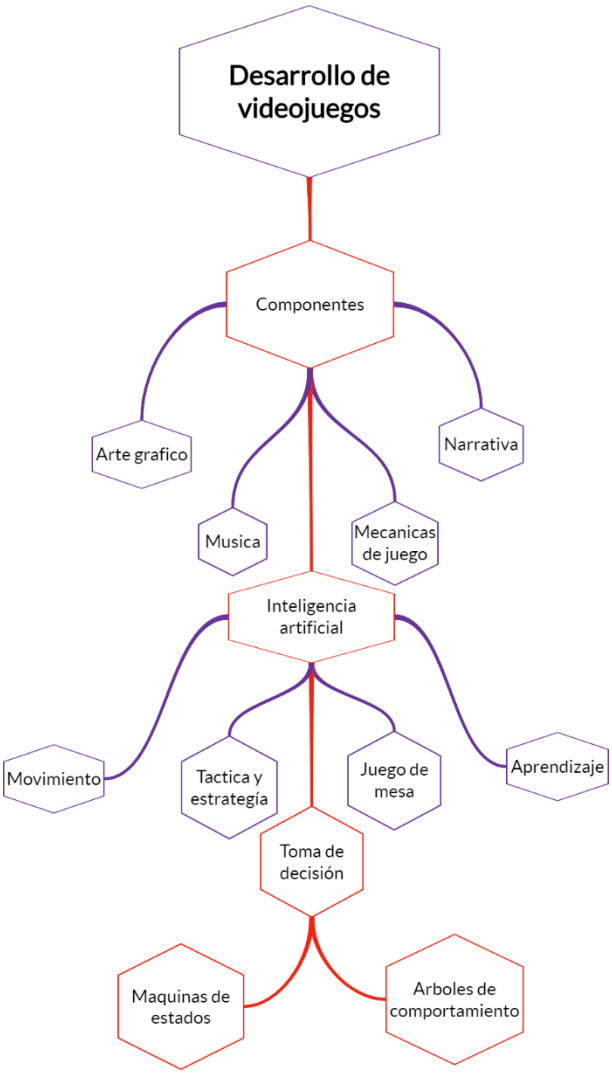

Figura 2. Componentes del desarrollo de videojuegos

4.2.2. Inteligencia Artificial

El concepto de inteligencia artificial estará presente a lo largo de todo el proyecto, por ese motivo es importante entender de manera clara de que trata esta área de la ciencia de la computación.

Para entender un poco más sobre este término, es necesario primero entender el significado de inteligencia, el cual según la real academia de la lengua española es definida como "potencia intelectual: facultad de conocer, de entender o comprender". De esta forma, podríamos entender la inteligencia artificial como la capacidad que tiene una máquina de

realizar estas acciones. La propuesta de Inteligencia Artificial nace con la pregunta ¿Pueden las máquinas pensar? [1]. Aunque esta es un área que ha pasado por varias iteraciones a lo largo de los años, el objetivo de la inteligencia artificial es el de hacer a una computadora o sistema capaz de razonar de la manera en la que lo haría una persona o animal [2], de esta forma podemos pensar en esta como la ciencia que incorpora el conocimiento a los procesos o actividades realizadas por una máquina.

Luego de acercarnos un poco más al significado de lo que se puede interpretar como inteligencia artificial, hay que destacar algunas de las áreas en las que esta se aplica

- *Tratamiento de lenguajes naturales:* En este campo son para aquellas interfaces hombre-máquina donde se nos permita interrogar una base de datos o dar órdenes a un sistema operativo, ejemplo de esto una aplicación que realice traducciones hombre - máquina.
- *Sistemas expertos:* En este campo es la experiencia del desarrollador la cual será la clave para poder realizar un sistema capaz de llegar a deducciones más cercanas a la realidad.
- *Robótica:* La robótica hoy en día es uno de los mayores exponentes que hay de inteligencia artificial ya que destaca en usos como lo son el ensamblaje de piezas, robots de asistencia entre otros que se encuentran en desarrollo.
- *Problemas de percepción:* Ya que como seres humanos no somos perfectos podemos usar máquinas que nos ayuden a detectar defectos en piezas en mal estado o identificar idiomas.
- *Aprendizaje:* Es la modelación de conductas las cuales luego podremos aplicar en computadores.

Partiendo de estas definiciones podemos poner la inteligencia artificial en el contexto del desarrollo de videojuegos, donde existen diversas técnicas para realizar la implementación de estas. Por ejemplo, este proyecto se centrará en estudiar la técnica de toma de decisión. Cada una de estas técnicas contiene arquetipos que varían según su definición, planeación e implementación. Cada implementación de estos arquetipos son llamados agentes de inteligencia artificial [8].

4.2.3. Inteligencia artificial de toma de decisión

La toma de decisión es la técnica de Inteligencia Artificial más utilizado para la creación de enemigos y acompañantes dentro de la industria del desarrollo de videojuegos. Esto debido a que provee autonomía y depende drásticamente de los muchos estados posibles del juego [1], lo que genera una experiencia agradable al usuario ya que le permite acercarse a cada situación de manera diferente; en resumen, le da el poder de decisión sobre su entorno.

La toma de decisión parte de un personaje autónomo, este procesa una serie de información que usa para generar una acción determinada [2]. Hay varios métodos para realizar este proceso de la mejor manera posible, entre estos se encuentran, pero no se limitan a, árboles de decisión y comportamiento, máquinas de estados, lógica difusa, sistemas basados en reglas y sistemas de comportamiento basados en objetivos.

Para representar gráficamente los arquetipos de inteligencia artificial de comportamiento se hace uso de los grafos. Como se puede observar en la Figura 3, un grafo es un conjunto de vértices y una colección de aristas que conectan un par de vértices [3].

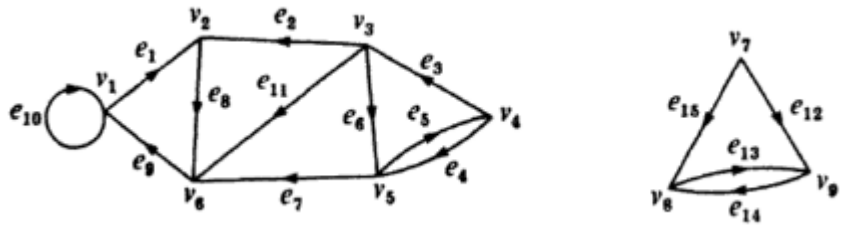

Figura 3. Ejemplos de grafos. Figura recuperada de A. Gibbons, Algorithmic graph theory, Cambridge University Press, 1985.

Cada uno de estos puntos o nodos representan una acción dentro de un agente, mientras que las líneas representan como estos nodos interactúan y conectan entre sí. Estas conexiones entre nodos son las que moldean el comportamiento de cada agente implementado.

A continuación, se realizará una abstracción de conceptos de los arquetipos con lo que se realizará la comparación de este proyecto: Máquinas de estados y árboles de decisión.

4.2.4. Máquina de estados

Las máquinas de estados son utilizadas cuando se requiere que un agente se comporte de una sola forma o de maneras limitadas [2]. Cada estado de la máquina representa una acción que el agente debe estar ejecutando, y cada agente sólo puede ejecutar un estado a la vez. De igual manera el agente sólo podrá realizar un cambio de estado si existe una relación entre ellos.

Por ejemplo, en la Figura 4 se puede observar que cada estado en la máquina está conectado al estado “morir”, lo que significa que el agente que implementa este arquetipo

puede morir en cualquier momento. Sin embargo, en esta misma Figura podemos observar que para que el agente pase del estado “Descansar” a “Esconderse” primero debe pasar por el estado patrullar debido a que no existe una relación directa entre ellos.

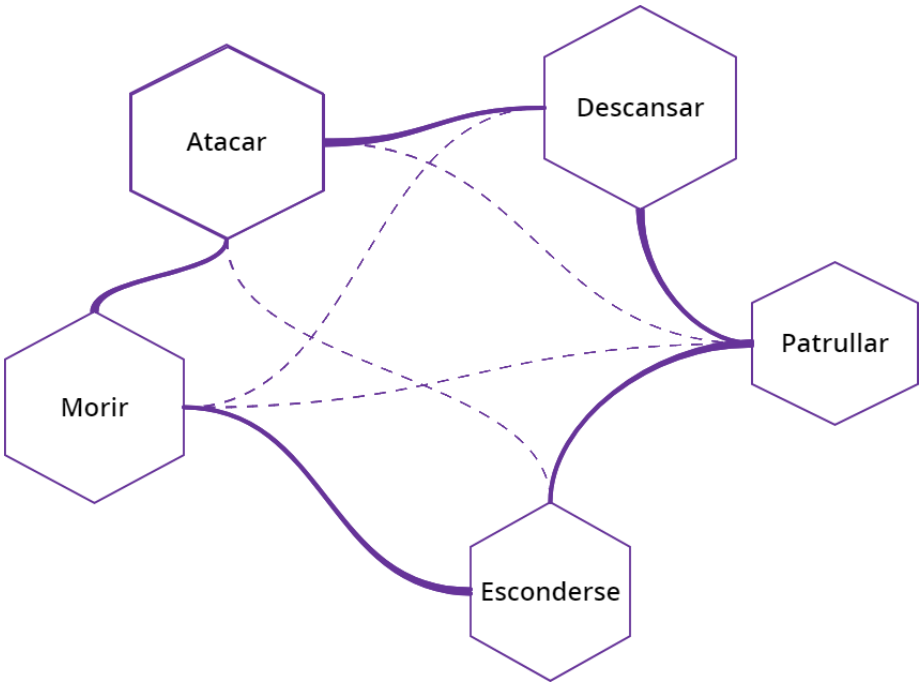

Figura 4. Grafo representando máquina de estados

Cada uno de estos estados está conectado por medio de transiciones, estás se ejecutan cuando cierta condición es cumplida y pueden tener acciones propias que se ejecutarán entre estados, lo cual es útil para inicializar variables o definir métodos entre estados.

Estos estados y transiciones dentro de las máquinas pueden ser representados con un diagrama gráfico como lo vimos anteriormente, en donde los nodos representan los estados y las flechas representan la transición [10].

4.2.5. Árboles de comportamiento

Los árboles de comportamiento cambian el paradigma de los estados y acciones de las máquinas de estados para reemplazarlas por bloques de tareas, estas tareas pueden ser desde preguntar el estado de una variable, ejecutar un método o hasta ejecutar una animación. Estos estados pueden juntarse de forma jerárquica para crear un comportamiento más complejo y completo [2].

Por ejemplo, la tarea actual del agente de la Figura 5 es el de “Encontrar el jugador” y existen 3 maneras de lograrlo, al ser un modelo jerárquico se realizará la sub-tarea de izquierda a

derecha. En caso de que la primera tarea, “Revisar última posición conocida”, no sea exitosa se ejecutará la siguiente tarea “Patrullar sitios de interés” y se continuará comportando de esa manera hasta que alguna de estas sub-tareas sea cumplida a cabalidad, una vez esta sea completada el nodo padre se marcará también como cumplido y permitirá continuar con otra tarea.

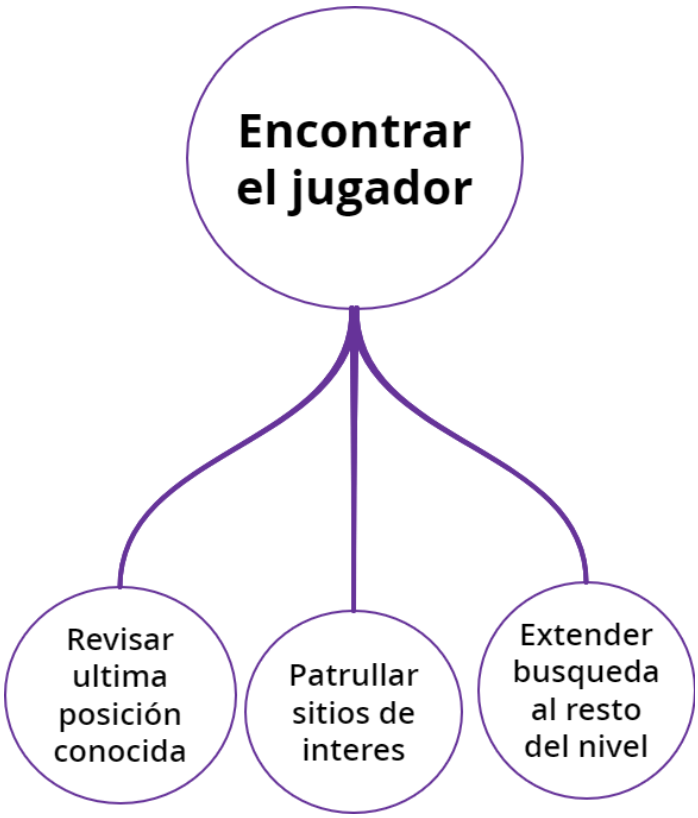

Figura 5. Ejemplo árbol de comportamiento

Este tipo de modelo jerárquico permite diseñar varias tareas independientes, juntarlas y diseñar comportamientos más complejos y variados.

4.2.6. Conceptos de implementación

Para realizar un videojuego es necesario el uso de diversas herramientas de igual manera que de las estrategias de planeación adecuadas para asegurar el éxito del desarrollo. A continuación, se detallará el motor de juego a utilizar, el lenguaje compatible con este y la metodología de planeación y ejecución a emplear en la realización del proyecto.

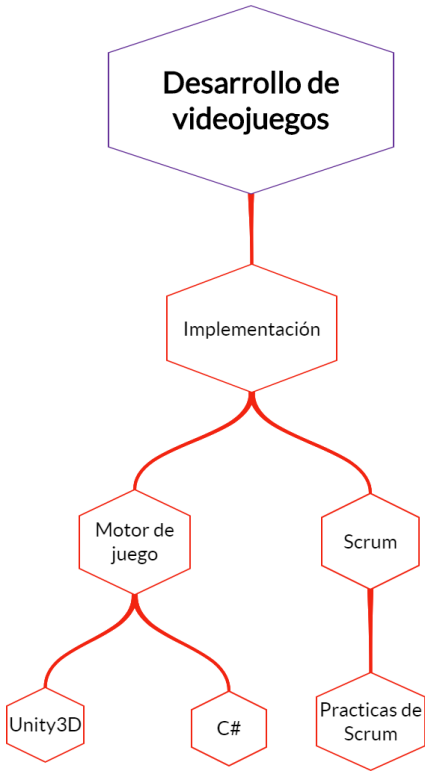

Figura 6. Conceptos de implementación de desarrollo de videojuegos

4.2.7. Motor de juego

El motor de juego es la herramienta que nos provee los recursos para diseñar la física, apariencia, rutinas de software, entre otros componentes del juego para que el programador pueda elaborar y desarrollar la lógica del mismo y poder escribir estas instrucciones en un lenguaje que la computadora pueda entender y ejecutar [11].

Para este proyecto se utilizará el motor de juego gratuito Unity 3D en su versión 5.6. A continuación se describirá como funciona este motor, sus beneficios y el lenguaje de programación a utilizar: C#.

De igual manera cabe resaltar que información más detallada del motor del juego y su uso dentro de la investigación podrá ser encontrado en el apartado 5. Diseño Metodológico de este proyecto.

4.2.8. Unity 3D

Unity 3D es un motor de juego gratuito que permite el desarrollo multiplataforma (Xbox, Play Station, PC, Android, iOS, entre otras). El cual cuenta con una alta gama de herramientas para facilitar el trabajo de los desarrolladores, para que se puedan enfocar en la creación de un juego de muy alta calidad, este motor soporta el lenguaje de programación orientado a objetos C#, al igual que otros lenguajes como JavaScript. Unity también cuenta con una comunidad de aprendizaje que proporciona la oportunidad a nuevos creadores de aprender.

Este motor permite integrar cada uno de los componentes mencionados en el apartado 4.2 de este documento. Por lo que se convertirá en la herramienta más importante en la Fase de implementación del proyecto.

4.2.9. C#

C# (“See Sharp”) es un lenguaje moderno orientado a objetos, que también incluye programación orientada a componentes y se encuentra dentro de la familia de lenguaje de C, junto a C y C++, está estandarizado por ECMA International sobre el estándar ECMA-334 e ISO/IEC, estándar ISO/IEC 23270 [4].
Este lenguaje también soporta programación orientada a componentes, debido a que actualmente el software se basa más en de paquetes de funcionalidad auto contenidos y auto descriptivos, donde la clave de estos es un modelo con propiedad, métodos y eventos; los cuales tienen atributos que proporcionan información sobre el componente y además de esto proporcionas su propia documentación [12].
El lenguaje se utilizará para realizar la codificación de las mecánicas del juego y la implementación de la inteligencia artificial.

4.2.10. Scrum

Scrum es una metodología para el ágil desarrollo de proyectos, la cual se basa en la creación de ciclos o iteraciones y que dentro de esta son llamados "Sprint". Para comprender el funcionamiento de esta metodología es necesario conocer las cinco fases que definen desarrollo ágil:

- Concepto: Se define de forma general las características del proyecto a desarrollar y se definen equipos de trabajos.
- Especulación: En esta fase se plantean los conceptos principales del nuevo proyecto, y a partir de estos se establecen los límites que marcaran el desarrollo del producto, como costos y agendas de trabajo. Se hará la construcción de un producto partiendo de las ideas principales y su impacto en el entorno. Además de esto Cabe destacar que esta fase se repetirá en cada iteración y sus rasgos generales son:
    - Desarrollo y revisión de requisitos generales.
    - Mantener un listado con la funcionalidad que se requieren.
    - Establecimiento de un plan de trabajo el cual definirá fechas, versiones e iteraciones, con el cual se medirá el esfuerzo en el proyecto.

- Exploración: Una vez terminada la fase de especulación y se incrementan los requerimientos del proyecto.

- Revisión: El equipo se reúne y realiza un contraste de los avances hechos con los objetivos planteados
- Cierre: Llegada la fecha de entrega acordada se entregará una versión del producto deseado. Cabe destacar que debido a que se trata de una versión, el cierre no indica que se ha finalizado el proyecto, sino que seguirán existiendo cambios en el mismo denominado mantenimiento, cuyo objetivo es acercarse cada vez más al producto final deseado [13].

Figura 7. Ciclo de desarrollo Scrum, Recuperado de Gestión de proyectos informáticos Metodología Scrum

4.2.11. COCOMO

COCOMO o Construcive Cost Model es un modelo de estimación de software empírico ya que las fórmulas que lo componen están obtenidas desde la práctica. Es uno de los modelos más completos para la estimación del software actualmente, sin embargo, hay que tener en cuenta que no todos los modelos pueden servirles a todos los proyectos y por eso hay que ser cuidadosos con su implementación. Por este motivo, se ha propuesto una escala de modelos sobre COCOMO los cuales son:

- Modelo COCOMO básico, calcula los esfuerzos y costos del desarrollo en función al tamaño del programa expresados en líneas estimadas de código (LDC)
- Modelo COCOMO intermedio, calcula el esfuerzo de costo no solo basando en función del tamaño del programa, sino que también de una evaluación subjetiva del producto, hardware, personas y atributos del proyecto.
- Modelo COCOMO avanzado, combina los factores evaluados en la base intermedia, con los impactos que pueda tener el proyecto durante su ciclo de vida.

También se han establecido tres prototipos de proyectos de software los cuales son:

- ❖ Modo orgánico: son aquellos proyectos de software donde trabajan pequeños equipos con experiencia y desarrollan sobre un conjunto de requisitos rígidos.

- Modo Semiacoplado: Son proyectos donde el tamaño y la complejidad son de un nivel intermedio, además de esto los equipos de trabajo están compuestos por desarrolladores con diversos niveles de experiencia y se trabajan bajo requerimientos medio o poco rígidos.
- Modo Empotrado, Son proyectos desarrollados bajo un conjunto de especificaciones de hardware y software muy restringidas.

Las ecuaciones con la que COCOMO de nivel básico trabaja son:

$$E = a_b KLDC^{b}_{b}$$

$$D = C_b E^{d}_{b}$$

Figura 8. Ecuaciones de COCOMO nivel básico, Recuperado de Las Métricas de Software y su Uso en la Región

Donde E es el esfuerzo aplicado en personas por mes, D es el tiempo de desarrollo en meses, KLDC es el número de líneas de código estimadas en miles, los coeficientes y exponentes son valores constantes determinados por la siguiente tabla [15].

Proyecto de Software	a_b	$^{b}_{b}$	c_b	$^{d}_{b}$
Orgánico	2.4	1.05	2.5	0.38
Semiacoplado	3.0	1.12	2.5	0.35
Empotrado	3.6	1.20	2.5	0.32

Tabla 2. Valores constantes, Recuperado de Las Métricas de Software y su Uso en la Región [15]

5. DISEÑO METODOLÓGICO

En este apartado del documento se detallará el proceso de diseño e implementación del videojuego en el cual se implementaron los arquetipos a comparar, la elaboración de la herramienta de pruebas de usuario y diseño e implementación de la metodología de pruebas de usuario.

5.1 Diseño y diagramación

Con el objetivo de realizar una comparación más cerca de las realidades de desarrollo en la industria de los videojuegos, se desarrolló Drones VR; un videojuego de supervivencia por rondas en primera persona que implementa un agente de inteligencia artificial llamado

“Agente de persecución” el cual fue diseñado especialmente para este videojuego. El agente es implementado utilizando los dos arquetipos que se evaluarán en este proyecto (Máquinas de estados y árboles de comportamiento) y de esta manera poder realizar una comparación equitativa entre ellos.
A continuación, se expondrá el concepto de Drones VR; su objetivo, las mecánicas que lo conforman y de misma manera su diseño e implementación.

5.1 Introducción a Drones VR

Drones VR es un juego de disparos en primera persona en el cual el jugador tiene el objetivo de sobrevivir la mayor cantidad de rondas posibles ante enemigos que se encuentran alimentados con un agente de inteligencia artificial de persecución y que harán su aparición de manera dinámica en el mapa según la posición actual del jugador. Para lograr este objetivo, el jugador contará con un equipamiento de distintas cualidades para derrotar estos enemigos y progresar hasta que sus puntos de vida se agoten. A continuación, se mostrarán en detalle el manejo del jugador, su arsenal y el sistema de aparición de los enemigos.

5.1.1 Manejo del jugador

La interacción del jugador con su personaje dentro del juego se da por medio del teclado y mouse del ordenador. Para esto, se relacionaron las acciones del personaje con diversos métodos de entrada de estos dispositivos cuya distribución puede ser observada en la Tabla 3.

Entrada	Acción
Teclado: W	Caminar al frente
Teclado: A	Caminar a la izquierda
Teclado: S	Caminar hacia atrás
Teclado: D	Caminar a la Derecha
Teclado: Q	Lanzar Granada
Teclado: R	Recargar Arma
Teclado: Espacio	Saltar
Teclado: Escape	Pausa
Mouse: Click Izquierdo	Disparar
Mouse: Click Derecho	Apuntar/Zoom
Mouse: Scroller	Cambiar arma

Tabla 3. Distribución de controles

5.1.2 Interfaz gráfica

Parte fundamental de una experiencia satisfactoria durante una sesión de juego radica en que el jugador pueda estar informado sobre el estado de su personaje, por este motivo se diseñó una interfaz gráfica que le permite al jugador estar consciente de esta información

en tiempo real. Como se puede observar en la Figura 9, el jugador puede consultar la información de sus puntos de vida, ronda actual y estado de su armamento (Munición actual, munición total y cantidad de granadas).

Figura 9. Perspectiva principal del jugador

Adicionalmente la barra de salud cambia de color dependiendo del arquetipo de inteligencia artificial implementado, donde será de color verde si la escena contiene una inteligencia artificial basada en máquinas de estado y azul si implementa los árboles de comportamiento.

Figura 10. Diferencias de barra de salud según implementación

5.1.3 Aparición de enemigos

La aparición de enemigos es el sistema más crítico del videojuego ya que es el que se encargará de inicializar cada uno de estos dinámicamente en el mapa, al igual que determinar la cantidad que aparecerán por ronda y simultáneamente.

Para calcular esto, se hace uso de una ecuación diferencial de crecimiento poblacional la cual se puede ver reflejada en la siguiente ecuación (Figura 11) donde **r** = número de la ronda, **M(r)** = enemigos en **r** ronda, **M(0)** es el valor de enemigos de la primera ronda y **s** es la semilla de crecimiento de la ecuación.

$$M(r) = M_0 * e^{k(r)}$$

$$\frac{Ln(M(1))}{M_0} = K \qquad M_{(1)} = s * M_0$$

Figura 11. Ecuación diferencial de crecimiento poblacional

La implementación de esta ecuación al sistema de aparición de enemigos garantiza que esta será capaz de alimentarse a sí mismo indefinidamente como se puede observar en la Figura 12, que muestra la cantidad de enemigos que aparecerán en cada una de las primeras 25 rondas cuando se cuenta con una semilla de crecimiento de valor 1.2 y M(0) = 4.

```
Poblacion ronda 1.0: 4.8
Poblacion ronda 2.0: 5.76
Poblacion ronda 3.0: 6.911999999999999
Poblacion ronda 4.0: 8.294399999999998
Poblacion ronda 5.0: 9.953279999999998
Poblacion ronda 6.0: 11.943935999999997
Poblacion ronda 7.0: 14.332723199999995
Poblacion ronda 8.0: 17.199267839999994
Poblacion ronda 9.0: 20.639121407999994
Poblacion ronda 10.0: 24.76694568959999
Poblacion ronda 11.0: 29.72033482751998
Poblacion ronda 12.0: 35.664401793023984
Poblacion ronda 13.0: 42.79728215162878
Poblacion ronda 14.0: 51.35673858195452
Poblacion ronda 15.0: 61.62808629834544
Poblacion ronda 16.0: 73.95370355801451
Poblacion ronda 17.0: 88.7444442696174
Poblacion ronda 18.0: 106.4933331235409
Poblacion ronda 19.0: 127.79199974824907
Poblacion ronda 20.0: 153.35039969789887
Poblacion ronda 21.0: 184.02047963747867
Poblacion ronda 22.0: 220.82457556497428
Poblacion ronda 23.0: 264.9894906779692
Poblacion ronda 24.0: 317.9873888135631
Poblacion ronda 25.0: 381.5848665762755
```

Figura 12. Numero de enemigos en las primeras 25 rondas

Una vez el número de enemigos por ronda sea definido por la ecuación, el sistema escogerá el lugar donde los enemigos harán aparición entre cuatro puntos distribuidos en el mapa. Esta decisión es tomada por un sistema que calcula el punto de aparición más lejano a la posición actual del jugador, de esta manera se logra ocultar la aparición de cada enemigo al jugador. La figura 13 muestra la posición de los puntos de aparición en una vista área del mapa marcados con círculos verdes.

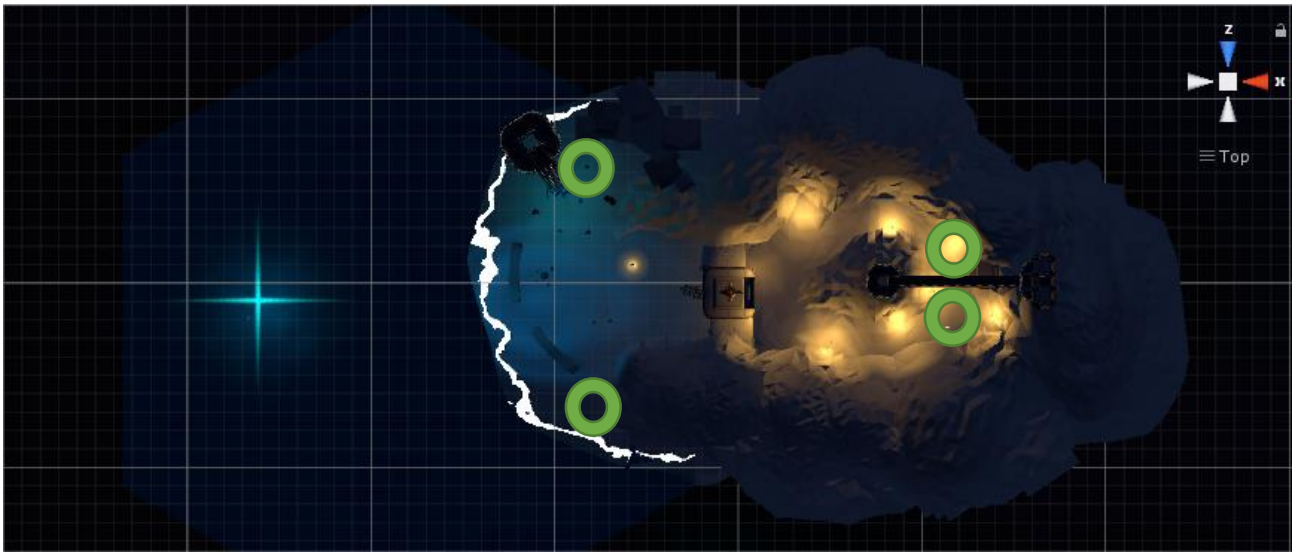

Figura 13. Puntos de aparición en el mapa

Finalmente, cuando el punto es definido, se le encarga el obtener e inicializar al enemigo que se encuentra almacenado en una piscina de enemigos desactivados a la espera de ser llamados.

Información técnica detallada del funcionamiento de esta piscina podrá encontrarse en el 5.2.1 Implementación Mecánicas de juego.

5.1.4 Inteligencia artificial

La inteligencia artificial es otro componente fundamental de la experiencia de juego, al igual que el ítem de más relevancia para cumplir con los objetivos de este proyecto. Este se encargará de suplir de un comportamiento a los enemigos que interactúan con el jugador y de esta manera lograr que cada sesión de juego sea única. Para esto, cada enemigo implementa un agente de comportamiento llamado Agente de Persecución. A continuación, se expondrá el comportamiento que moldeará este agente y sus dos implementaciones.

5.1.4.1 Agentes de persecución

El objetivo de este agente es el de perseguir al jugador, hacerle daño y morir en caso de que sus puntos de vida lleguen o sean menor a 0. Para darle forma a este comportamiento se hizo uso de 2 arquetipos de inteligencia artificial que, después de ser implementados, fueron asignados a 2 personajes diferentes mostrados en la Figura 14 y la Figura 15.

Figura 14. Soldado Drone

Figura 15. Araña Drone

5.1.4.1.1 Máquinas de estados

Este fue el primer arquetipo en modelar el comportamiento del Agente de Persecución. Para lograr esto y como se puede observar en la Figura 16, se diseñó un diagrama de nodos en el cada uno de estos representa un estado que debe cumplir el agente, al igual que las transiciones que conectan estos estados y dan forma al comportamiento.

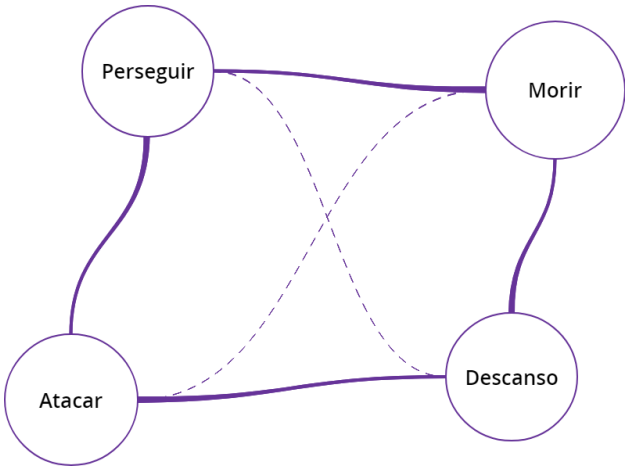

Figura 16. Diagrama de Nodos representando agente de persecución con máquina de estados

Para teoría detallada acerca de las Máquinas de Estados, dirigirse a la sección *4.2.4 Máquinas* de estados de este documento.

5.1.4.1.2 Árboles de Comportamiento

A diferencia de las máquinas de estados, cada nodo dentro de los Árboles de Comportamiento puede representar una acción, una validación o actualización. Dicho esto, para comprender el comportamiento que modelará el árbol de la Figura 17, es importante tener en cuenta que estos se ejecutan de manera jerárquica de izquierda a derecha.

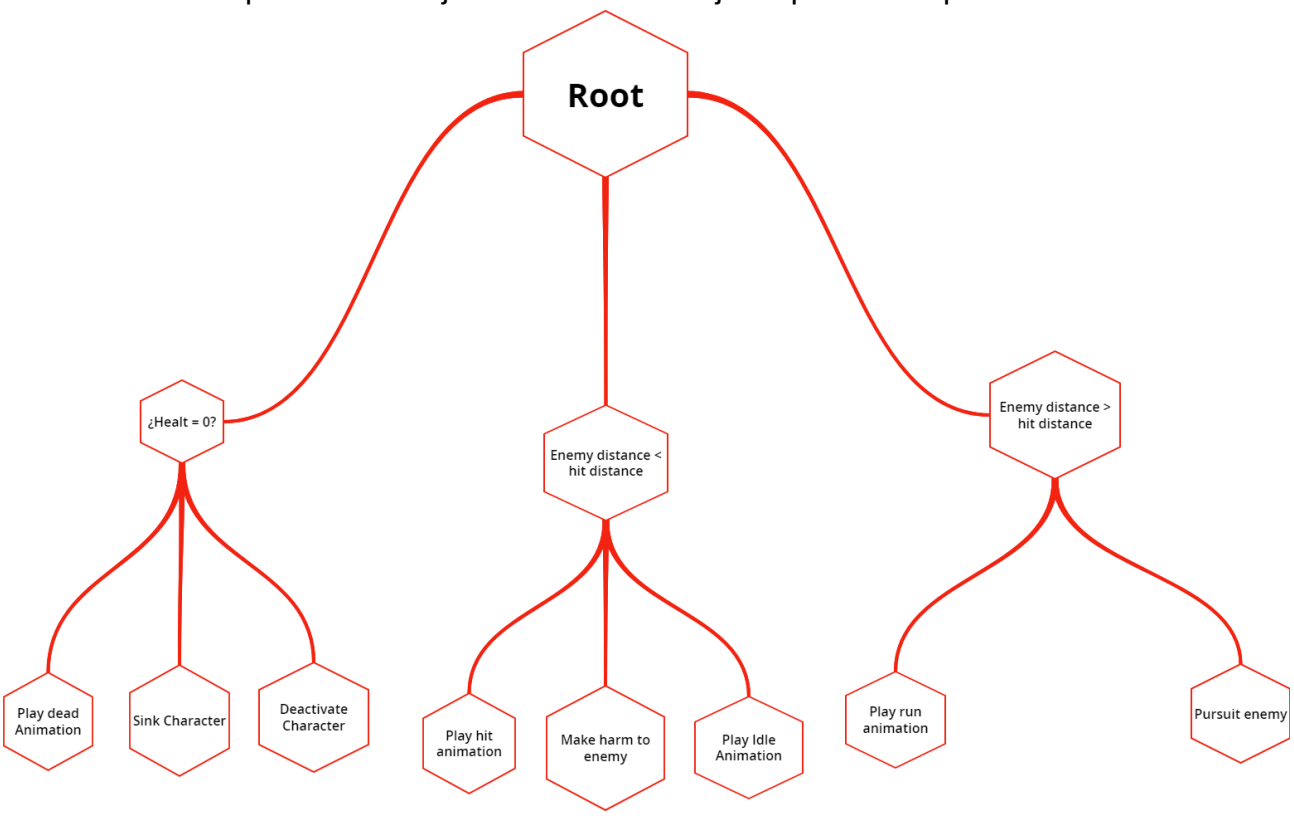

Figura 17. Diagrama de Nodos representando agente de persecución con árbol de comportamiento

Para teoría detallada acerca de los Árboles de Comportamiento, dirigirse a la sección *4.2.5 Árboles de Comportamiento* de este documento.

5.2 Desarrollo e implementación

Una vez terminada la fase de diseño se procederá a desarrollar los distintos módulos que componen el videojuego en el motor gráfico Unity 3D con el lenguaje de programación C#. Cabe destacar que el arte gráfico y la mecánica de disparos en primera persona, fueron obtenidos del *Unity Asset Store*, el cual es una tienda virtual de componentes para apoyar

el desarrollo de individuos o empresas pequeñas. Una lista de los componentes adquiridos podrá ser observada en el Anexo 1. Listado de *Assets* Utilizados.

En los siguientes ítems se explicará el procedimiento de implementación del módulo gráfico, mecánicas, interfaz de usuario e inteligencia artificial.

5.2.1 Implementación de módulo gráfico

El estilo de arte que prevalece en DronesVR es la de polígonos bajos, esto gracias a que su fácil implementación y rendimiento alto permite enfocarse en el módulo de inteligencia artificial el cual es el más importante de este proyecto.

5.2.1.1 Implementación grafica ambiental

La implementación gráfica ambiental fue por medio de arte modular lo que permite unir múltiples segmentos de una pieza de arte para formar otra diferente, con esta técnica fue implementado el mapa principal del juego visto en la Figura 18 y Figura 19.

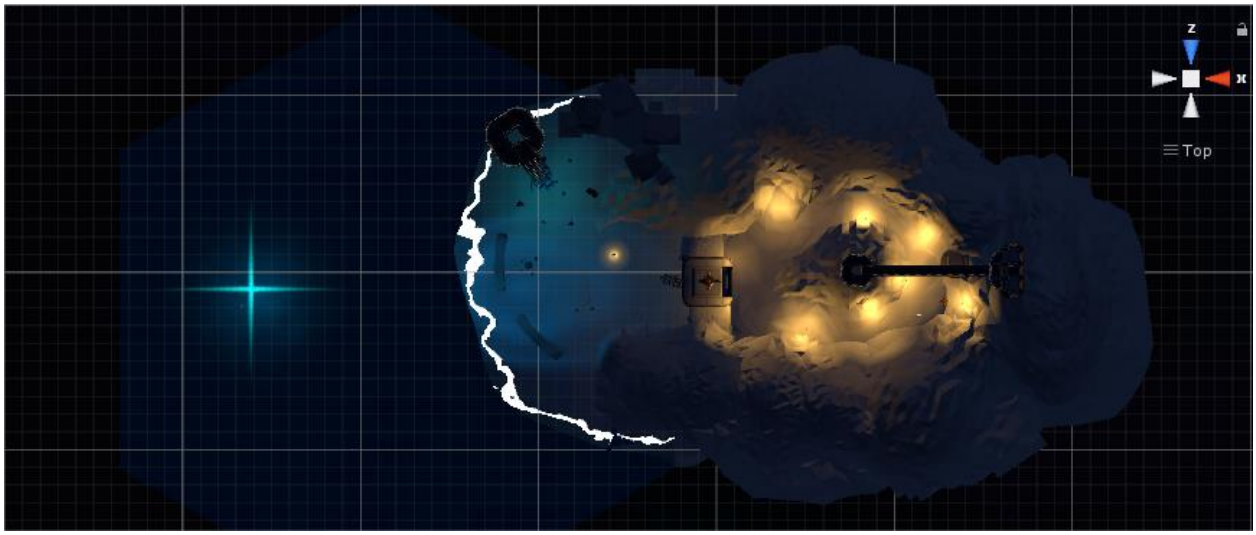

Figura 18. Vista aérea mapa escena principal del juego

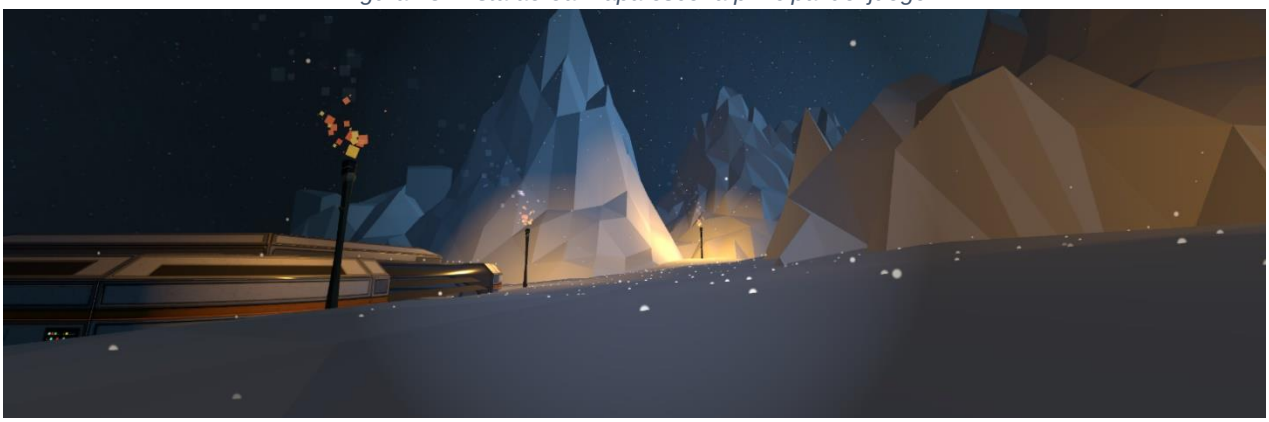

Figura 19. Vista mapa de juego

5.2.1.2 Implementación gráfica de enemigos

Para la implementación de los enemigos fueron utilizados los *assets SciFi Enemies and Vehicles* y *Robot Alator Animated (PBR)* (Ver Anexo: Lista de Assets Utilizados). Cada uno de estos incluye diferentes animaciones propias qué fueron utilizados para darle un mayor realismo al juego, para poder realizar el controlador de dichas animaciones se utilizó el sistema de animaciones de Unity 3D. Este sistema se basa en Máquinas de Estados para controlar la animación actual que ejecuta el personaje. Dicho esto, y como se puede observar en la Figura 20, el controlador de las animaciones del personaje es similar a la Figura 16 que representa el agente de persecución a implementar vía Máquina de Estados.

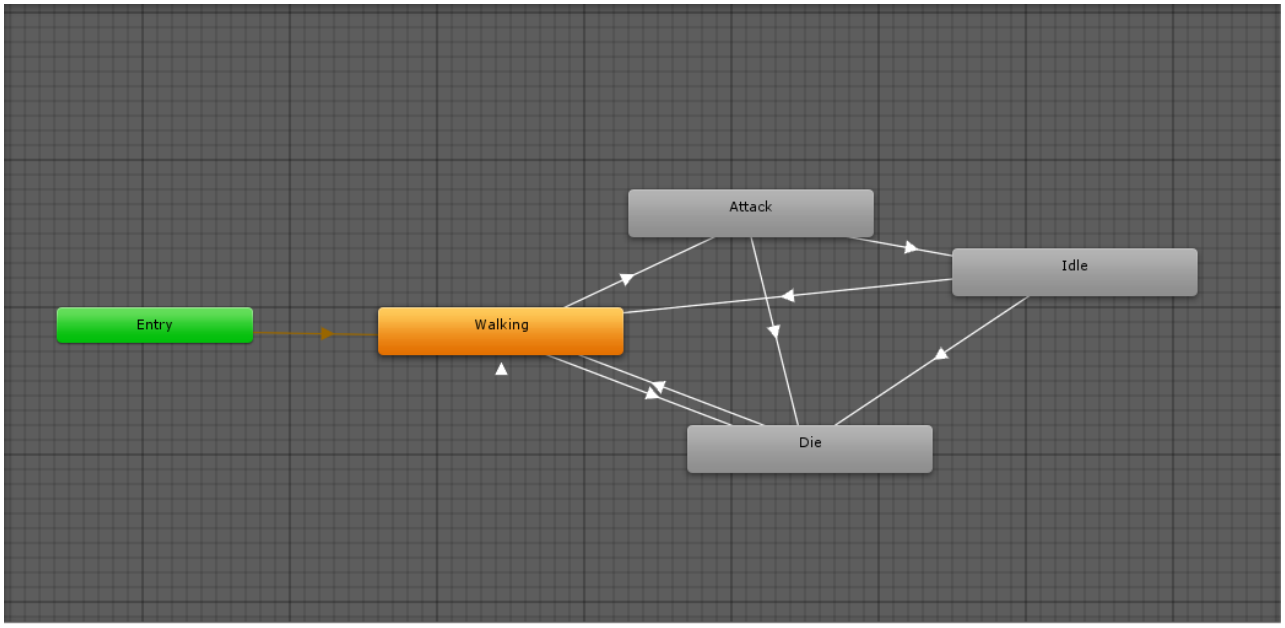

Figura 20. Controlador de animaciones de enemigo

A pesar de que los estados de la máquina dicten que animación se debe ejecutar en determinado periodo de tiempo, son las transiciones las que nos ayudarán a navegar la máquina de manera controlada asegurando un cambio consistente entre animaciones. Para realizar estas transiciones, se creó una referencia del controlador de animaciones dentro de cada implementación de inteligencia artificial lo que permitió a cada arquetipo realizar el cambio de animaciones según fuera requerido.

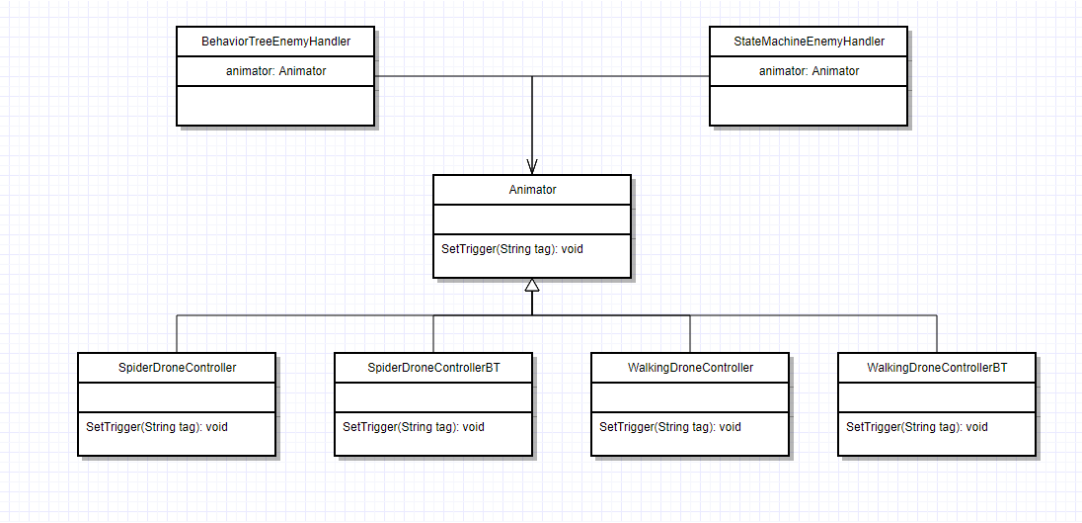

Figura 21. Diagrama de clases arquetipos / animaciones

La Figura 21 muestra como los controladores de inteligencia artificial del enemigo crean un objeto de tipo *Animator*, lo que les permite acceder al método que gestiona las transiciones: SetTrigger. Esto permite que el arquetipo de inteligencia artificial realice las transiciones sin importar el tipo de enemigo o controlador que tenga asignado.

5.2.1 Implementación de las mecánicas de juego

Existen 2 mecánicas de vital importancia dentro del proyecto, la mecánica de manejo de personaje y la mecánica de aparición de enemigos. La primera de estas mecánicas fue implementada utilizando el *asset* FPS Starter Kit – Pro (Ver Anexo: Lista de Assets Utilizados). Este *asset* provee el sistema de armamento y control necesarios para ejercer control total sobre el personaje, este fue integrado con los componentes de interfaz gráfica de usuario e inteligencia artificial como se muestra en la Figura 22.

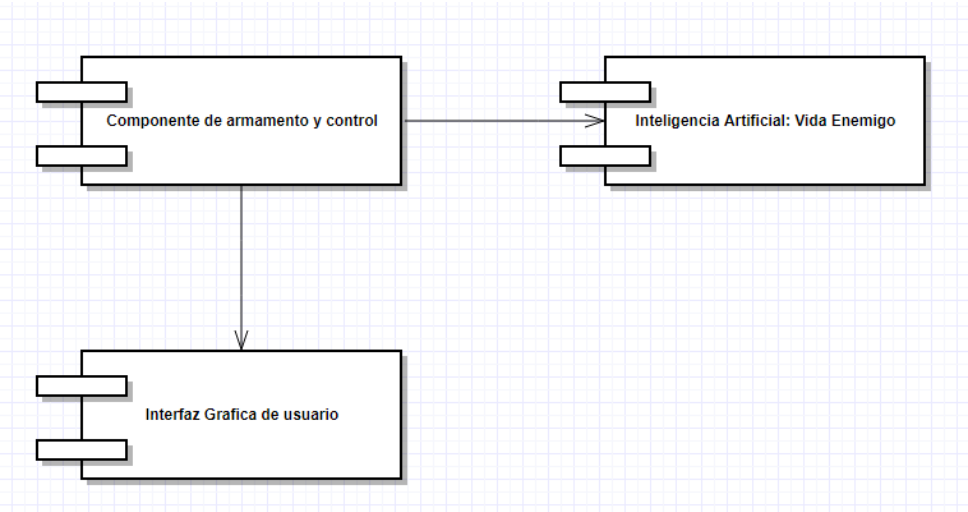

Figura 22. Diagrama de componentes conexión control/interfaz/inteligencia artificial

Por otro lado, la mecánica de aparición de enemigos se maneja por medio de 3 clases:

- *EnemyPoolScript* que se encarga de crear los enemigos y mantenerlos almacenados en una “piscina” a la espera de ser llamados.
- *EnemySpawnHandler* cuya función es de escoger el punto de aparición más lejano al jugador y realizar los cálculos de la cantidad de enemigos por ronda siguiendo lo estipulado en la sección 5.1.3 Aparición de enemigos
- *PointSpawnHandler* que se encarga de llamar al enemigo de la piscina y activarlo en su posición.

Como se puede observar en la Figura 23 estas 3 clases trabajan entre sí para gestionar efectivamente la aparición de los enemigos. Jerárquicamente, *EnemyPoolScript* cuenta con varios objetos de tipo *PointSpawnHandler* para poder elegir cual es el punto de aparición más optimo y llamar su método CreateEnemy(), que se encargará de buscar un enemigo de la piscina en *EnemyPoolScript* y activarlo en su posición, lo que de igual manera da inicio al arquetipo de inteligencia artificial implementado por dicho enemigo.

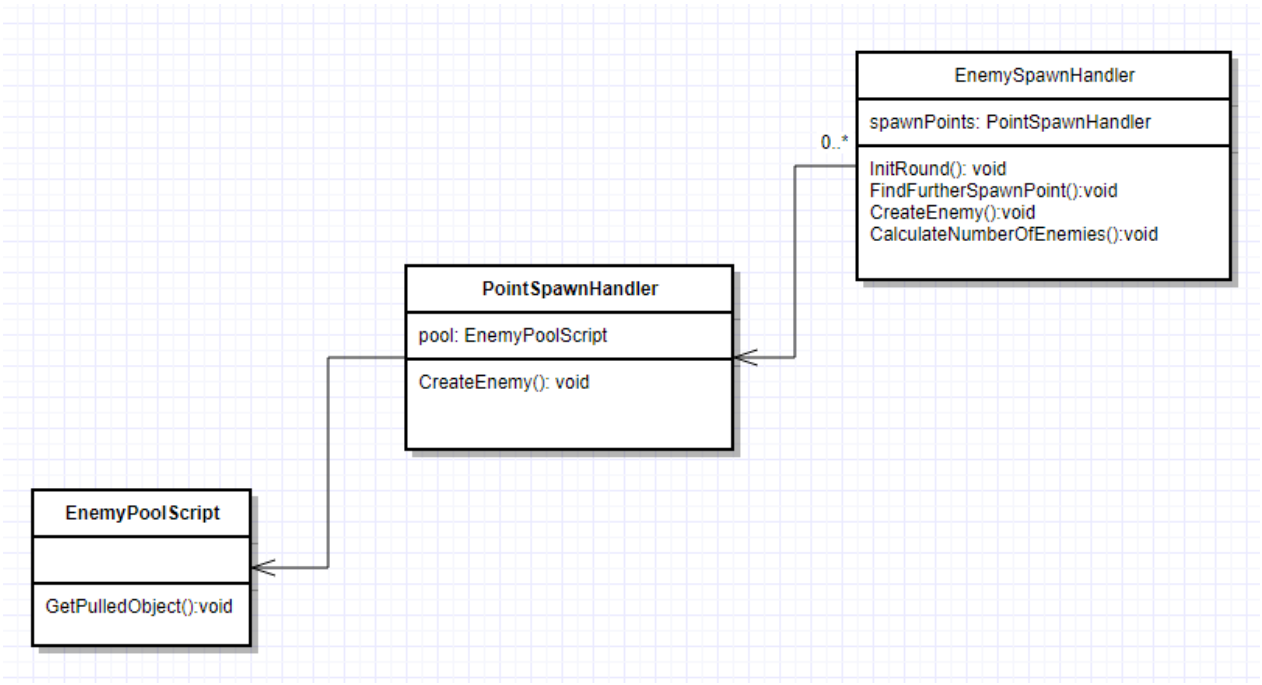

Figura 23. Diagrama de clases de aparición de enemigos

5.2.2 Implementación inteligencia artificial

Las dos versiones que existirán del videojuego serán las mismas en implementación gráfica y mecánicas, la única diferencia, y la más importante, es el arquetipo de inteligencia artificial que este alimentando a los enemigos durante la sesión de juego.

A continuación, se detallará la implementación de los 2 arquetipos a analizar en el proyecto.

5.2.2.1 Maquina de estados

Las máquinas de estados del proyecto fueron implementadas siguiendo el diagrama mostrado en la Figura 15 de este documento. Para realizar dicha implementación se creó la clase controladora *StateMachine* que se encarga de darle forma a la máquina de estado específica del comportamiento visto en la Figura 24.

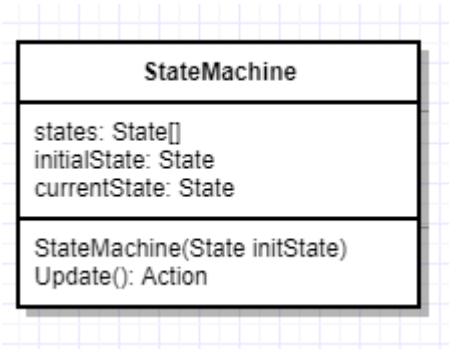

Figura 24. Clase StateMachine

Adicionalmente se crea un controlador que se encargará de crear la máquina de estados y manejar los estados específicos con sus respectivas transiciones. En el caso del proyecto, y como se puede observar en la Figura 25 se creó la clase *StateMachineEnemyHandler*, la cual crea un objeto de los estados específicos:

- *StatePursuitEnemy:* Se encarga de perseguir al jugador
- *StateHitEnemy:* Se encarga de golpear al jugador
- *StateIdle:* Queda inactivo después de golpear al jugador
- *StateDeath:* Desactiva al enemigo cuando sus puntos de vida se agoten

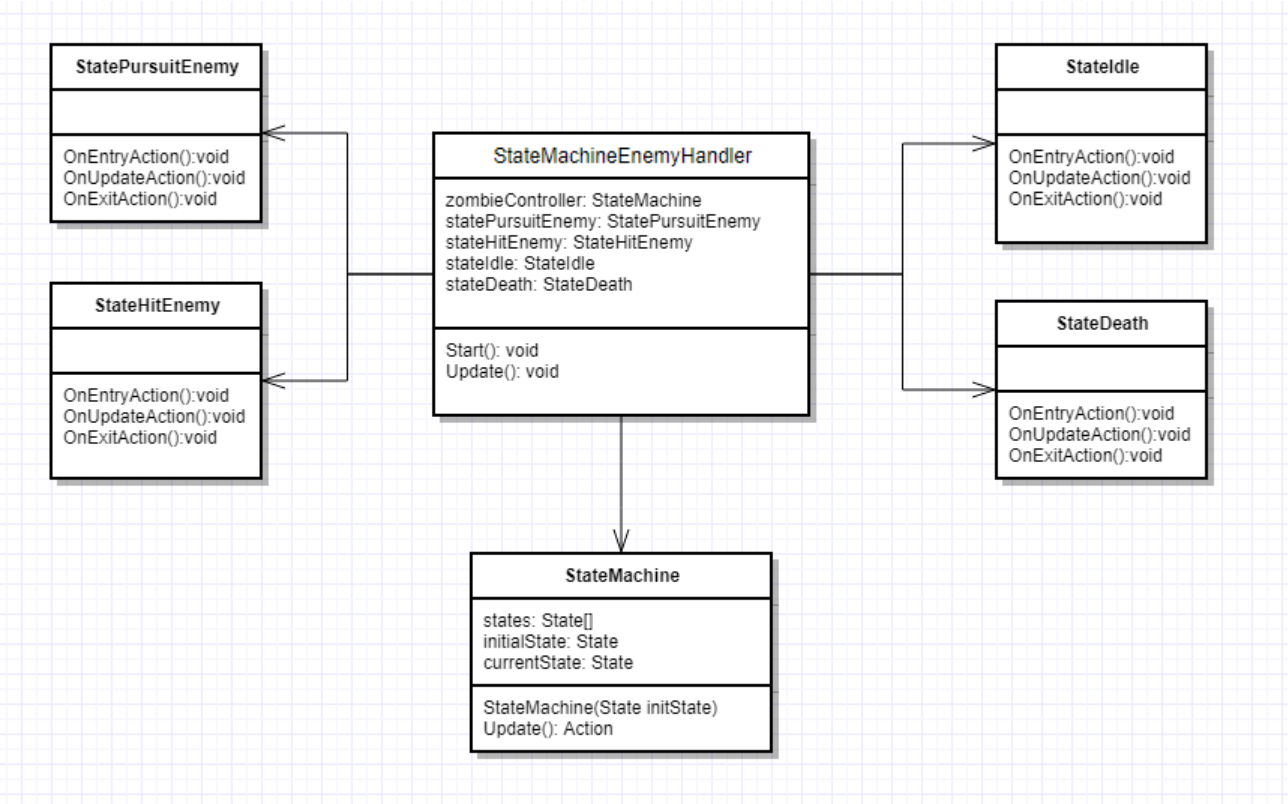

Figura 25. Diagrama de clases máquina de estados

5.2.2.2 Árbol de comportamiento

Para la implementación del Árbol de Comportamiento se utilizó la librería *NPBehave*, una librería para la creación de árboles de comportamiento manejados por eventos y que puede ser encontrada en el siguiente enlace: https://github.com/meniku/NPBehave.

A diferencia de las máquinas de estado, no es necesario crear una clase dedicada para cada comportamiento, en cambio, se genera un controlador que crea un objeto de tipo *Root* y a este se le asignan las acciones o ramas del árbol como métodos, que son encapsulados jerárquicamente en un objeto tipo *Node* y posteriormente es asignado al objeto Root como puede observarse en la Figura 26. Una vez el objeto *Root* se encuentre inicializado empezará a ejecutar las acciones del árbol de manera jerárquica de izquierda a derecha como se muestra en la Figura 17.

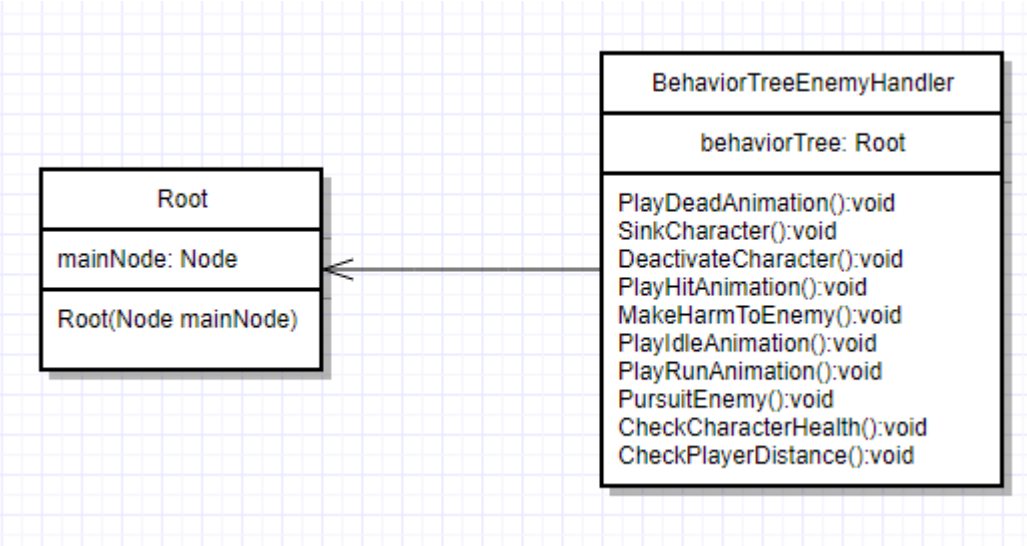

Figura 26. Diagrama de clases máquina de estados

6. RECOLECCIÓN Y ANÁLISIS DE INFORMACIÓN

En este apartado se estipulará la manera en la que serán realizadas las pruebas a los arquetipos de Inteligencia Artificial implementados en el video juego. Para este análisis se realizaron dos tipos de pruebas: De usuario y de rendimiento, al igual que una matriz de implementación.

6.1 Prueba de percepción de usuario

El objetivo de esta prueba es el de encontrar cuál implementación de inteligencia artificial prefieren los jugadores, si las máquinas de estado o los árboles de comportamiento. Con este objetivo en mente, se inició por escoger la población objetivo de la investigación, una vez esta se encontró establecida, se procedió con la implementación de la prueba que constó de dos sesiones de juego, una por arquetipo, y una encuesta que enfrenta los modelos de implementación.

6.1.1 Población

La población a la que se le aplicó la prueba fueron 16 hombres y mujeres de 17 años en adelante, esto debido a que las mecánicas y temáticas del juego clasificarían al video juego en la categoría M de *la ESRB (Entertainment Software Rating Board)*, que es la agencia que da clasificación a los videojuegos para que padres puedan tomar decisiones respecto al contenido de estos. Esta agencia estipula que los juegos que pueden contener violencia intensa son un contenido apto para personas de 17 años o más [14].

De igual manera está población fue segmentada en dos grupos diferentes: 8 Personas que juegan seis o más horas semanales y 8 personas que juegan menos de seis horas semanales, esto con el fin de clasificar qué arquetipo de inteligencia artificial escogen los jugadores que dedican más horas a jugar a aquellos que no lo hacen.

6.1.2 Metodología de aplicación de la prueba

Para esta prueba se buscarán personas mayores de 17 años y se les preguntará si les es posible realizar dos sesiones de prueba para la evaluación de la inteligencia artificial. Una vez la persona acceda a participar de la prueba se le solicitará que firme un Consentimiento Informado el cual será adjuntado en este documento. Posteriormente se le realizó una introducción sobre el propósito del proyecto para enfocar la atención de la persona en la inteligencia artificial al igual que un tutorial de los controles del juego.

Ya terminada la inducción se le indicará al usuario que puede iniciar la primera de las dos sesiones de juego que tendrá que realizar durante la prueba. La primera de estas pruebas se realizará en la compilación del juego que contiene las máquinas de Estados, la cual se identificará gracias a la barra de vida color verde. Mientras que la segunda sesión contendrá el arquetipo de árboles de comportamiento y de igual manera, se identificará con la barra de vida color azul, como se puede observar en la Figura 10.

Una vez el usuario termine de realizar las dos sesiones de juego, se le proporcionará una encuesta que contendrá la siguiente pregunta: ¿Cuál arquetipo de inteligencia artificial considera que generó el comportamiento más creíble? Esta pregunta se realizó basada en la pregunta formulada por MC Namee, quien identificó que lo más viable para este tipo de comparación es analizar la credibilidad de los agentes, debido a que “un sistema es considerado exitoso si sus personajes se comportan continuamente de una manera consistente y creíble” [5].

6.2 Pruebas de rendimiento

Para realizar las pruebas de rendimiento y con el fin de crear un ambiente que permita un análisis justo de los arquetipos, se creó un escenario de pruebas como se muestra en las Figuras 27 y 28. Este escenario cuenta con un punto de aparición de enemigo marcado con una flecha azul, punto de aparición de personaje enfatizado con una flecha amarilla y línea de disparo marcada con una flecha verde.

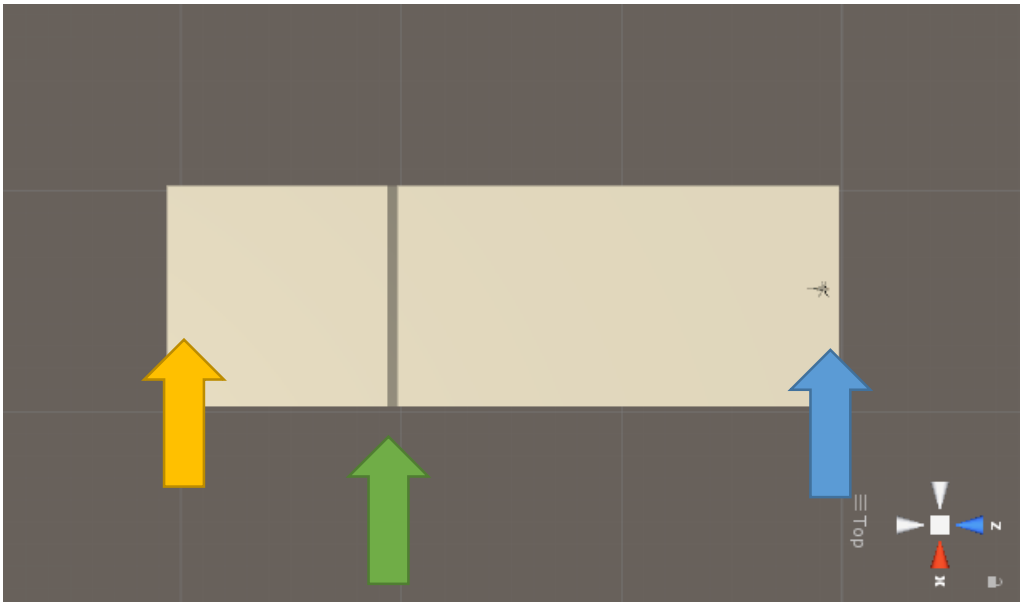

Figura 27. Vista área mapa pruebas rendimiento

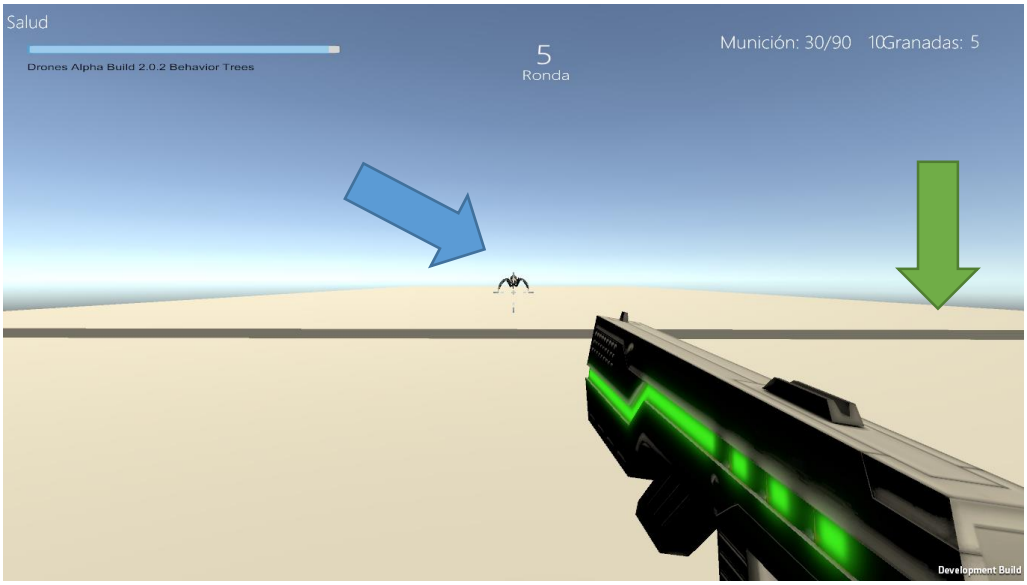

Figura 28. Perspectiva jugador mapa pruebas de rendimiento

El objetivo de este escenario es el de crear una mecánica de pruebas para la inteligencia artificial siguiendo el flujo mostrado en la Figura 29. Esto permitirá que los arquetipos sean

medidos con un comportamiento controlado evadiendo el ambiente impredecible del escenario principal de Drones VR.

Esta prueba se realizará un total de 10 veces por arquetipo, en las cuales para recolectar la información del tiempo de ejecución se utilizará la clase Stopwatch del namespace System.Diagnostic propio de C#. Esta clase provee una serie de propiedades que se utilizan para medir precisamente intervalos de tiempos determinados [6]. Como se puede observar en la Figura 29, el método de inicio de medición de tiempo es empleado cuando el enemigo es activado y continuará recopilando el tiempo de ejecución hasta que la vida del enemigo sea igual o menor a cero. Con esto se medirá el tiempo de ejecución del comportamiento, incluyendo velocidad de transición de estados/ramas.

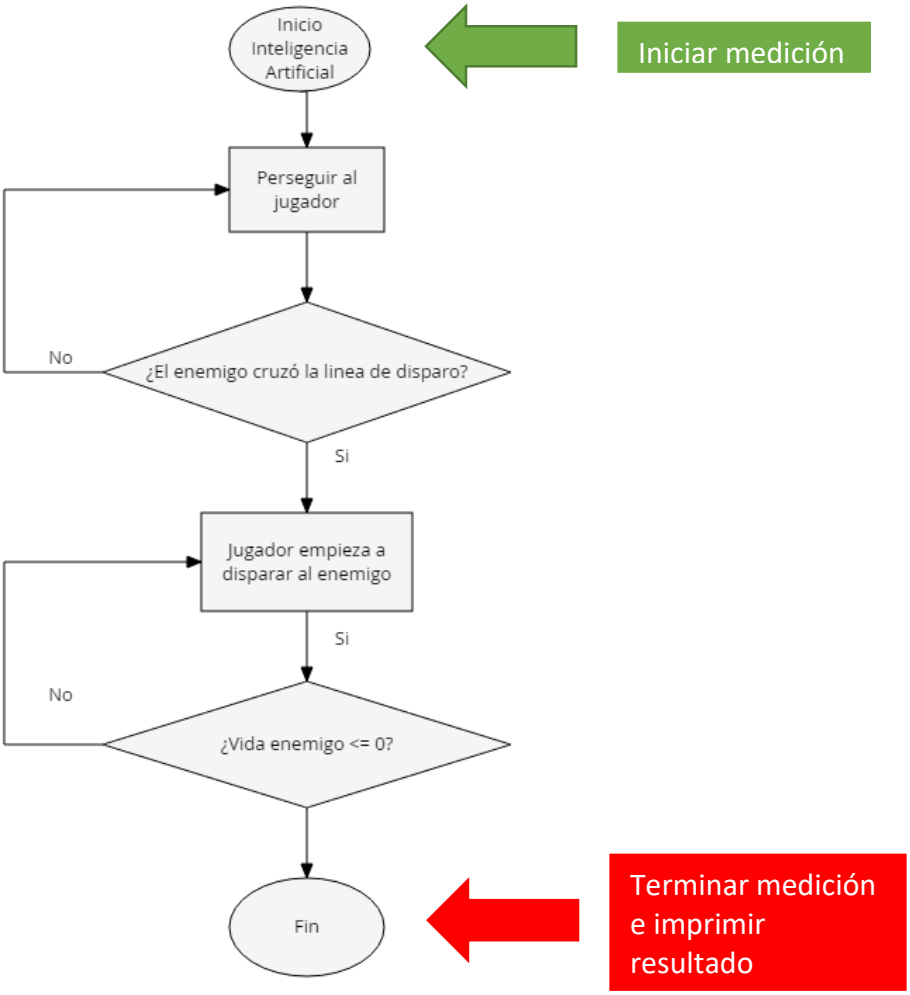

Figura 29. Diagrama de flujo de pruebas de rendimiento

Una vez el ciclo haya llegado a su Fin se imprimirá el tiempo de ejecución de este en la consola de Unity3D como se puede observar en la Figura 30.

Figura 30. Ejemplo muestra de resultado pruebas de rendimiento

6.3 Matriz de implementación

Con el objetivo de medir la facilidad de implementación de cada uno de los arquetipos se realizó una matriz comparativa como la vista en la Tabla x. Esta matriz contrastará las líneas de código y la cantidad de clases necesarias para la implementación de cada uno de los arquetipos, de igual manera con esta información se realizará un análisis con el Modelo Constructivo de Costos COCOMO; expuesto en el segmento 4.2.11. COCOMO.

	Maquinas de Estados	Arboles de Comportamiento
Cantidad de Clases		
Numero de Lineas de Codigo (LOC)		
Diferencia de cantidad de clases		
Diferencia LOC		

Tabla 4. Matriz comparativa de implementación

7. RESULTADOS

Una vez estas pruebas y métodos de medición fueron aplicados se realizó la normalización de los datos y análisis de resultados que serán expuestos a continuación.

7.1 Prueba de percepción de usuario

Como se puede observar en la Figura 31 y la Tabla 5, al realizar la prueba de usuario especificada en la sección 6.1 Metodología de aplicación de la prueba en una población de 16 jugadores (8 que juegan 6 o más horas semanales y 8 que juegan menos de 6 horas semanales), se encontró que 75% de los jugadores que juegan más de seis horas semanales percibieron que la inteligencia artificial elaborada con el Árbol de Comportamiento se comportó de manera más creíble que la implementada con Máquina de Estados. En contraste, los jugadores que dedican menos de seis horas semanales a jugar no demostraron una preferencia marcada entre arquetipos, esto ya que el 50% de jugadores, eligieron el agente de persecución con Árbol de Comportamiento como arquetipo más creíble y el 50% restante aquel que implementa la Máquina de Estados.

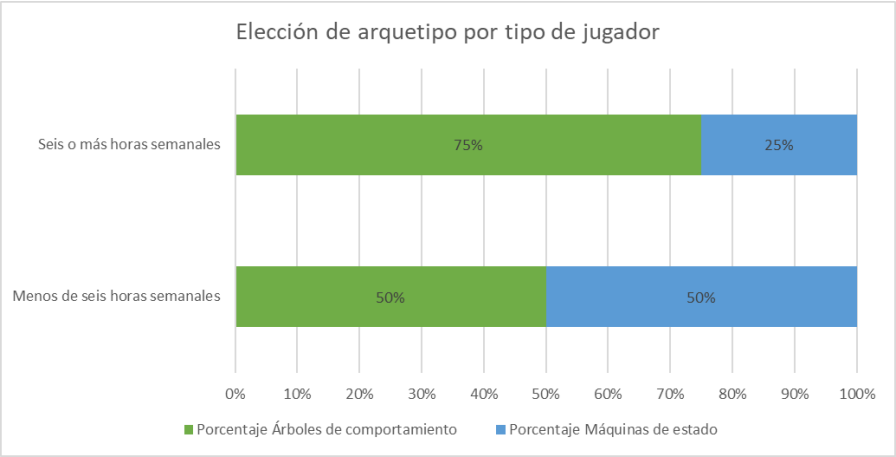

Figura 31. Elección de arquetipo por tipo de jugador

Cuenta de Arquetipo elegido	Cuenta		Porcentaje	
Etiquetas de fila	Árboles de comportamiento	Máquinas de estado	Árboles de comportamiento	Máquinas de estado
Menos de seis horas semanales	4	4	50%	50%
Seis o más horas semanales	6	2	75%	25%

Tabla 5. Elección de arquetipo por tipo de jugador

Después de realizar el análisis global de la población, como se muestra en la Figura 32 y la Tabla 6, se encontró que el 62.5% de esta considera que los Árboles de Comportamiento modelan la conducta más creíble, mientras el 37.5% restante se inclinan hacia las Máquinas de Estados.

Figura 32. Porcentaje global de arquetipo elegido

Arquetipo	Cuenta de Arquetipo elegido	Porcentaje de Arquetipo elegido
Árboles de comportamiento	10	62,50%
Máquinas de estado	6	37,50%

Tabla 6. Porcentaje global de arquetipo elegido

7.2 Prueba de rendimiento

Después de realizar las pruebas establecidas en el ítem 6.2 Pruebas de Rendimiento, se encontró que la inteligencia artificial implementada con las Máquinas de Estado tiene un tiempo de ejecución ligeramente más bajo a comparación de los Árboles de Comportamiento.

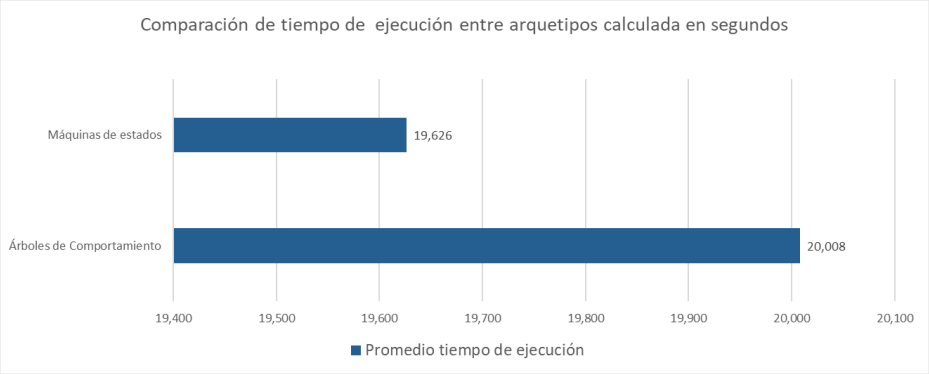

Figura 33. Comparación de tiempo de ejecución entre arquetipos calculada en segundos

	Árboles de Comportamiento	Máquinas de estados
Tiempo ejecución prueba 1	19,83	19,25
Tiempo ejecución prueba 2	20,05	19,75
Tiempo ejecución prueba 3	20,22	19,64
Tiempo ejecución prueba 4	20,01	19,72
Tiempo ejecución prueba 5	19,87	19,54
Tiempo ejecución prueba 6	20,09	19,78
Tiempo ejecución prueba 7	19,93	19,57
Tiempo ejecución prueba 8	20,16	19,63
Tiempo ejecución prueba 9	20,09	19,65
Tiempo ejecución prueba 10	19,83	19,73
Promedio tiempo de ejecución	20,008	19,626
Desviación Estándar	0,138	0,153
Diferencia con el arquetipo contrario	+0,382	-0,382

Tabla 7. Comparación de tiempo de ejecución entre arquetipos calculada en segundos

Como se puede observar en la Figura 33 y la Tabla 7, la inteligencia artificial que implementó el Árbol de Comportamiento tuvo un promedio de ejecución 0.38 segundos mayor (20.00 segundos) que el agente construido con la Máquina de Estados (19.62).

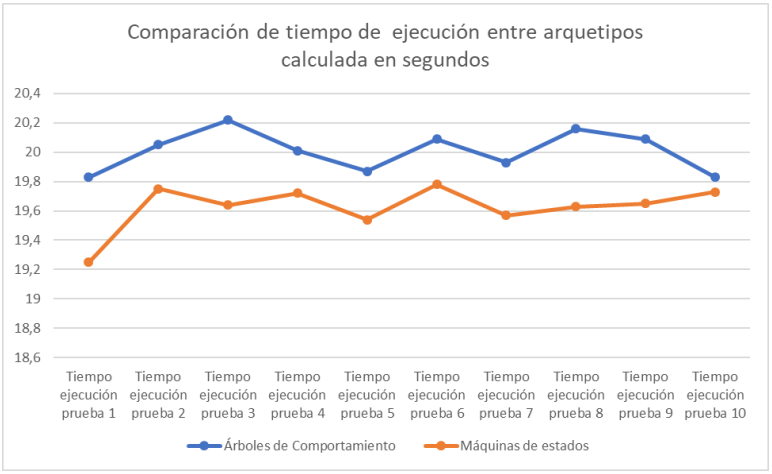

Figura 34. Comparación de tiempo de ejecución por prueba entre arquetipos calculada en segundos

De igual manera como se puede observar en la Figura 34, la Máquina de Estados no solo obtuvo un mejor rendimiento promedio si no que obtuvo el mejor tiempo de ejecución en cada una de las 10 pruebas realizadas.

Cabe destacar que cada una de las 20 pruebas (10 por arquetipo), fueron realizadas utilizando la configuración gráfica mostrada en la Figura 35 e implementadas en el mismo dispositivo; cuyas características se pueden encontrar en la figura 36.

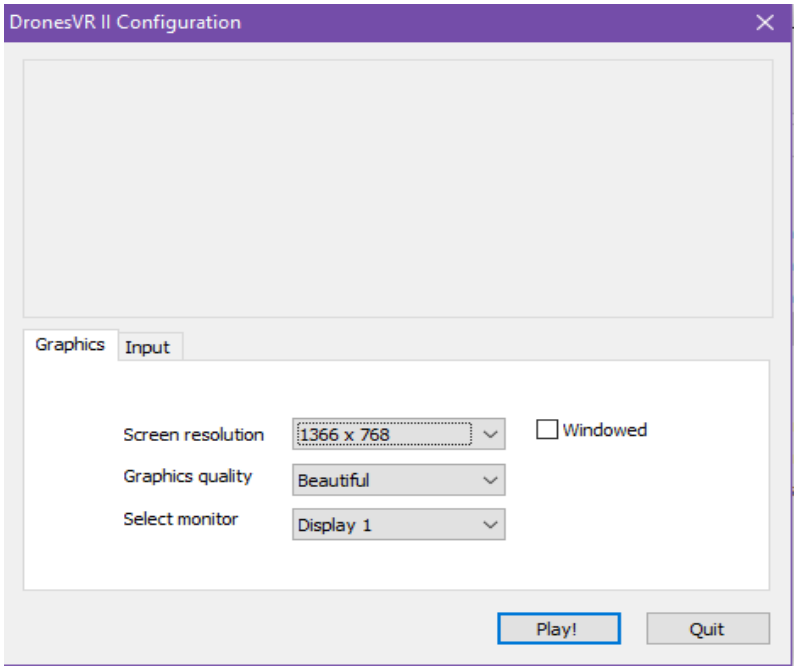

Figura 35. Configuración grafica de la prueba

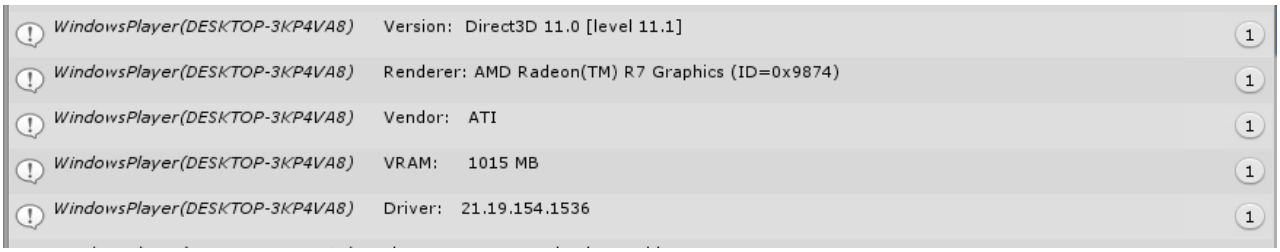

Figura 36. Especificaciones técnicas de maquina donde se realizaron las pruebas

7.3 Matriz de implementación

Con los arquetipos implementados, probados y analizados, se realizó el llenado de la matriz especificada en la sección 6.3 Matriz de implementación. Como se puede observar en la Tabla 7, la inteligencia artificial que realizada con el Árbol de Comportamiento fue, en valor de líneas de código (LOC) y número de clases la más sencilla de implementar ya que solo se necesitó una clase y 155 LOC para lograr su implementación, mientras que la implementación de la Máquina de Estado necesito de 5 clases y 344 líneas de código.

	Maquinas de Estados	Arboles de Comportamiento
Cantidad de Clases	5	1
Numero de Lineas de Codigo (LOC)	344	155
Diferencia de cantidad de clases	4	-4
Diferencia LOC	+189	-189

Tabla 8. Matriz comparativa de implementación

7.4 Matriz comparativa

Finalmente, en esta sección se realizará la matriz que comparará el rendimiento, facilidad de implementación y jugabilidad de los arquetipos de inteligencia artificial implementados: Máquinas de Estados y Árboles de Comportamiento. A continuación, se mostrará la matriz comparativa con su respectivo análisis.

Agente de Persecución			Máquina de Estado	Árbol de Comportamiento
Parámetros de calificación	1	Porcentaje de aceptación global percepción de usuario	37.5%	62.5%
	2	Tiempo de ejecución calculado en segundos	19.626	20.008
	3	Número de líneas de código	344	155

Tabla 9. Matriz comparativa de implementación final

Como se puede observar en la Tabla 8, el agente que implementó el árbol de comportamiento fue el que los jugadores consideraron se comportó de manera más creíble y es el más sencillo de implementar, sin embargo, es el que más tiempo de procesamiento consume lo que a grandes escalas podría descartar su uso en dispositivos

con latencia de procesamiento baja como los dispositivos móviles, pero lo convierte en una opción de importante consideración para el desarrollo en ordenadores y consolas.

Por otro lado, el agente que implementó la máquina de estados tuvo el mejor tiempo de procesamiento lo que lo hace el más viable para videojuegos en dispositivos móviles u ordenadores con baja capacidad de procesamiento.

8. CONCLUSIONES

A lo largo de esta investigación se lograron establecer factores importantes que interceden a la hora de evaluar y clasificar arquetipos de inteligencia artificial para videojuegos.

De igual manera se logró determinar qué arquetipo puede satisfacer decisiones de diseño como población, tiempo de implementación y rendimiento. Para lograr esto, Unity 3D fue una herramienta fundamental gracias a la flexibilidad, escalabilidad y soporte de librerías externas que hacen que sea una opción viable para este tipo de investigaciones.

Los jugadores de hoy en día cada vez son más exigentes en todo sentido, por lo que incorporar inteligencia artificial es una estrategia interesante que le permite al jugador interactuar con videojuegos que simulan el comportamiento de los humanos.

Como trabajo futuro se desea seguir experimentando nuevos arquetipos y otras técnicas de inteligencia artificial que hagan que los videojuegos sean cada vez más “inteligentes”.

9. BIBLIOGRAFÍA

[1] Turing A, «The Imitation Game» *Computing Machinery and Intelligence,* vol. LIX, nº 236, 1950.

[2] Millington, I. Funge, J, *Artificial Intelligence for Games 2*, Morgan Kaufmann. 20 09

[3] Sedgewick, R. Wayne, K. Algorithms. Addison-Wesley, 2011

[4] S. W. G. Anders Hejlsberg, The C# Programming Language, Addison-Wesley, 2006.

[5] B. Mac Namee, Proactive Persistent Agents: Using Situational Intelligence to Create Support Characters in Character-Centric Computer Games, University of Dublin, 2004.

[6] "Stopwatch Class (System.Diagnostics)", Msdn.microsoft.com, 2017. [En línea]. Disponible: https://msdn.microsoft.com/en-us/library/system.diagnostics.stopwatch.aspx. [Consultado: 25- Oct- 2017].

[7] "C# and Java: Comparing Programming Languages", Msdn.microsoft.com, 2017. [En línea]. Disponiblee: https://msdn.microsoft.com/en-us/library/ms836794.aspx. [Consultado: 16- Oct- 2017].

[8]F. Escolano, M. Cazorla, M. Alfonso, O. Colomina and M. Lozano, Inteligencia artificial modelos, técnicas y áreas de aplicación. Madrid, España: Thomson Ediciones Spain, 2003, pp. 3-4.
[9] M. Casasnovas, "F.S.M. Máquinas de Estado Finitas", 2014. [En línea]. Disponible: http://www.profesores.frc.utn.edu.ar/electronica/tecnicasdigitalesi/pub/file/AportesDelCudar/Maquinas%20de%20Estado%20MC%20V5.pdf. [Consultado: 24- Oct- 2017].
[10] U. Technologies, "Unity - Manual: Lo básico de los Estados de Máquina", Docs.unity3d.com, 2017. [En línea]. Disponible: https://docs.unity3d.com/es/current/Manual/StateMachineBasics.html. [Consultado: 22- Oct- 2017].
[11] J. Garfias Frías, "La industria del videojuego a través de las consolas", Scielo.org.mx, 2017. [En línea]. Disponible: http://www.scielo.org.mx/scielo.php?pid=S0185-19182010000200010&script=sci_arttext&tlng=pt. [Consultado: 13- Oct- 2017].
[12] A. Hejlsberg, S. Wiltamuth and P. Golde, "C# Language Specification", ACM DIGITAL LIBRAR, 2003. [En linea]. Disponible: https://dl.acm.org/citation.cfm?id=861332. [Consultado: 17- Oct- 2017].
[13] M. Gallego, Gestión de proyectos informáticos Metodología Scrum. 2017, pp. 32-35.
[14] " Age and content ratings for video games and apps from ESRB", Esrb.org, 2017. [En línea]. Disponible: http://www.esrb.org/ratings/. [Consultado: 28- Oct- 2017].
[15] H. González Doria, Las Métricas de Software y su Uso en la Región. Puebla, México, 2017, pp. 104-106.