BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Animación de personajes virtuales en ambientes virtuales inmersos interactivos

TESIS PROFECIONAL

Tesis presentada para obtener el grado de Licenciatura en Ingeniera en tecnologías de la información.

Facultad de ciencias de la computación

Presenta: Elberfeld Enrique Pérez Gerónimo
Asesor de tesis : Dr. Abraham Sánchez López

Noviembre del 2019

I. Agradecimientos.

Hay muchas personas a las que estoy agradecido por este logro, primero que nada y más importante, me gustaría agradecer sinceramente a mis amados padres, Teófila Geronimo Francisco e Isauro Pérez Navarrete, quienes me han hecho ser quien soy gracias al camino que juntos han recorrido, mostrándome el verdadero significado de ser leal y perseverante. Es por su ejemplo, apoyo y amor que he llegado hasta aquí.

Al Dr. Abraham, por ser un gran profesor, mentor y amigo que nunca me permitió darme por vencido. Gracias por ser un guía estricto y por permitirme poder trabajar en el laboratorio de MOVIS y contar con todos sus recursos.

A mis compañeros del laboratorio de MOVIS cada día aprendiendo algo nuevo todo para terminar de la mejor manera esta travesía llamada universidad

Por último, a la Facultad de Ciencias de la Computación y la Benemérita Universidad Autónoma de Puebla, por facilitar los recursos para la elaboración de este trabajo de tesis y por brindarme una formación de calidad.

II. Dedicatoria.

Dedico este trabajo de tesis a mis padres. Gracias por todo el esfuerzo echo para que lograra terminar mi carrera a pesar de las carencias, lograr mantenerme para que terminara en tiempo y forma la carrera, gracias por tener plena confianza en que podía terminar sin conocer acerca de la carrera, y seguir apoyándome con todo a pesar de las dificultades.

Gracias por seguir adelante y por a poyarme en esta travesía que se veía larga, pero acabo muy rápido, sin más gracias por esto y más motivos esta tesis es dedicada a ustedes.

III. Índice.

I. Agradecimientos..... I

II. Dedicatoria. II

III. Índice. III

IV. Índice de figuras. VII

1. Introducción. 1

1.1. Resumen. 1

1.2. Contenido de la tesis. 3

1.3. Contribución de la tesis. 5

2. Estado del Arte. 7

2.1. Animación conductual de grupos y muchedumbre. 13

2.2. Avance del modelado del entorno para muchedumbre. 17

3. Marco teórico. 20

3.1. Mundos virtuales. 20

3.1.1. ¿Qué ofrecen los ambientes virtuales? 20

3.2. Muchedumbre en mundos virtuales. 21

3.3. Requisitos y restricciones para el modelado de muchedumbre. 23

3.4. Modelo del personaje virtual. 24

3.5. El problema de planificación. 26

3.6. UNITY. 27
3.7. Animación en Unity. 28
3.8. Flujo de trabajo de la Animación 30
3.9. Mapas probabilísticos. 33
3.10. Roadmaps probabilistas. 33
3.11. Métodos locales 34
3.12. Camino lineal 34
3.13. Caminos de tipo A* 35
4. Desarrollo de los ambientes de trabajo. 36
4.1. Unity como herramienta de trabajo. 38
4.1.1. Requisitos del sistema para la Unity. 38
4.1.2. Para ejecutar juegos de Unity 39
4.1.3. Herramientas de Unity. 40
4.2. Creación del mundo Virtual. 44
4.3. Creación de los Modelos. 46
4.4. Creación de la población. 48
4.5. Obtención de la ruta con PRM. 50
4.6. Obtención del camino con Unity. 58
4.7. Animación en unity. 61
5. Propuesta del sistema desarrollado. 62
5.1. Condiciones de la simulación. 64

5.2. Mundo virtual o ambiente. 64
5.2.1. Características físicas del modelo. 65
5.2.2. Características en el software. 65
5.2.3. Funciones. 66
5.3. Personaje. 66
5.3.1. Características físicas del modelo. 66
5.3.2. Características en el software. 67
5.3.3. Funciones del personaje. 67
5.4. Obstáculos. 68
5.4.1. Características físicas del modelo. 68
5.4.2. Características en el software. 68
5.4.3. Funciones. 69
5.5. Interfaces. 69
5.6. Controles por teclado. 71
5.7. Simulación final. 74
6. Pruebas y Resultados obtenidos en la aplicación. 76
6.1. Resultados por escenario. 80
6.2. Escenario 1. 80
6.3. Escenario 2. 81
6.4. Escenario 3 y 4. 82
6.5. Escenario 5. 83

7. Conclusiones. 85

8. Trabajo a futuro. 87

9. Bibliografía.....88

IV. Índice de figuras.

FIG. 2.1 JERARQUÍA DE ANIMACIÓN DE AVATARES. 25

FIG. 2.2. A) CAMINO PRODUCIDO POR EL MÉTODO LOCAL LINEAL B) RESULTADO DE UN CAMINO PRODUCIDO POR EL MÉTODO LOCAL A* 35

FIG. 3.1. UNITY SISTEMAS A EXPORTAR. 27

FIG. 3.2. AC3D 44

FIG. 3.3. MODELO Y PLANO 45

FIG. 3.4. SKETCHUP 45

FIG. 3.5. MODELO FINAL. 46

FIG. 3.6. FUSE 47

FIG. 3.7. MIXAMO 48

FIG. 3.8. CREACIÓN DEL PERSONAJE. 49

FIG. 3.9. CREACIÓN DE LA POBLACIÓN. 49

FIG. 3.10. GRID Y ESCENARIO. 50

FIG. 3.11. GENERACIÓN DEL ROADMAP. 51

FIG. 3.12. OBTENCIÓN DEL CAMINO. 52

FIG. 3.13.CAMINO OPTIMIZADA. 53

FIG. 3.14. DIFERENCIA ENTRE CANTIDAD DE NODOS 54

FIG. 3.15. NODOS AGREGADOS. 54

FIG. 3.16.AVATARES CON PRM. 55

FIG. 3.17. MÚLTIPLES PISOS 57

FIG. 3.18. BAKE. 59

FIG. 3.19. TITULO. 60

FIG. 3.20. MODELO DE ANIMACIÓN. 62

FIG. 3.21. ANIMACIÓN UNITY.....29
FIG. 3.22. PASOS PARA ANIMACIÓN EN UNITY.....31
FIG. 4.1. PUNTOS DE REUNIÓN. 73
FIG. 4.2.SIMULACIÓN DE LA APLICACIÓN. 75
FIG. 5.1 . ESCENARIOS EN UNITY. 78
FIG. 5.2. ESCENARIO 1. 81
FIG. 5.3. ESCENARIO 2. 82
FIG. 5.4.ESCENARIO 3 Y 4. 82
FIG. 5.5. ESCENARIO 5. 83

1. Introducción.

1.1. Resumen.

Gracias a la aparición de tecnologías de hardware y software, hoy en día existe la posibilidad de contar con una nueva clase de aplicaciones interactivas, capaces de manipular personajes 3D con un grado de complejidad cada vez mayor. La complejidad asociada a la simulación del comportamiento de este tipo de personajes con apariencia humana (humanoides virtuales), sugiere la integración de diversas disciplinas, como la psicología, las ciencias dedicadas al estudio del movimiento humano, los gráficos por computador y la inteligencia artificial.

El mayor desafío en la animación de personajes virtuales es la naturalidad en los movimientos al simular una gran cantidad de agentes, debido a que se deben realizar comportamientos complejos en entornos realistas, este es un problema difícil con aplicaciones en robótica, gráficos de computadora y animación. Un sistema multiagente puede ser una herramienta útil para estudiar un rango de situaciones en simulación para planificar y entrenar eventos reales. Los sistemas que soportan dichas simulaciones se pueden usar para estudiar situaciones de emergencia o desastre, incluyendo búsqueda y rescate, control de multitudes civiles, evacuación de un edificio y muchas otras situaciones de entrenamiento, en nuestro caso la simulación de evacuación en la biblioteca central de la BUAP.

Por lo mismo se tiene como objetivo crear escenarios y personajes virtuales reales; en el caso del escenario modelado, se ha escogido la biblioteca central de la BUAP; necesitamos crear de forma eficiente modelos de personas 3D los cuales contaran con un formato que permite poder trabajar con animaciones estándar para humanoides, lo cual se logrará creando situaciones virtuales con programas de cómputo que permiten la manipulación y creación de modelos, así como su programación, para poder involucrar a los modelos de personas en el ambiente o escenario y ver como se comportaría, para que al unirlos todo muestre la realidad con cierta fidelidad.

La parte más importante de la simulación será la interacción entre los personajes y la planificación de su camino, si es que se desea que se pueda añadir un nivel de precisión y detalle, necesario en el estudio de situaciones del mundo real. A través de nuevas tecnologías aparecidas a lo largo del tiempo, estas permiten que los modelos sean realistas, a diferencia de otras propuestas que se centran menos en el diseño de los modelos, regularmente por falta de recursos o simulaciones a muy grande escala, donde más que ver el resultado individual se centran en resultados poblacionales.

El enfoque tomado para los comportamientos, nos permite plantear un problema de planificación, lo cual se lleva a cabo a través de animaciones y la obtención de un camino. El objetivo es sembrar las bases para una simulación de evacuación, empezando con la creación de diversos caminos para cada personaje y la mejora de estas en base a escenarios virtuales, así como notar que mejoras se pueden hacer al sistema, para que su implementación pueda ser variable y adaptable dependiendo del ambiente y la situación.

1.2. Contenido de la tesis.

Las aplicaciones graficas 3D en tiempo real, que reproducen escenarios tridimensionales y proporcionan cierto grado de interacción entre el usuario y distintos elementos del mundo virtual simulado, han experimentado un notable auge gracias a la evolución tecnológica desarrollada por sus principales componentes, es decir, los subsistemas gráficos y de computación, las técnicas de animación, etc. Dentro de este ámbito, las aplicaciones graficas que requieren la implementación de algún tipo de personaje o criatura virtual, con un grado de autonomía, que representen una realidad o problema que se pueda sucintar en distintas simulaciones en entorno virtuales con personajes autónomos que interactúan entre ellos. Los personajes virtuales son muy importantes en su ambiente; además de distintas formas de representarlos y llegar al grado de crear una simulación donde los personajes interactúan entre ellos de tal manera que logren un objetivo. En nuestro caso escapar de un incendio y con ello ver cómo sería una evacuación real en la biblioteca central de la BUAP, para con ello poder medir tiempos en un entorno virtual y poder analizarlo.

Para cumplir con la propuesta desarrollada en este trabajo de tesis, se plantea el siguiente objetivo general.

- Desarrollar el ambiente de trabajo, probando diferentes métodos con los cuales el personaje pueda llegar a su destino, de tal manera que la propuesta del sistema desarrollado sea una simulación que permita

trabajar con ambientes virtuales inmersivos interactivos los cuales permitan la interacción con personajes virtuales humanos.

Igualmente, se plantean los siguientes objetivos específicos.

- Analizar alguna estrategia para hacer eficiente la interacción de humanos virtuales.
- Desarrollar escenarios 3D inmersivos e interactivos.
- Proponer técnicas de animación de personajes virtuales.
- Presentar diversas situaciones (escenarios) donde los personajes puedan desenvolverse.
- Relacionar la interacción con diversos personajes.

Para llevar a cabo dichos objetivos cada capítulo mostrara cómo se fueron desarrollando y cómo el resultado una aplicación final la cual se fue creando.

Estado del arte: contiene toda la historia y evolución tanto de vida artificial, juegos, simulaciones y diferentes áreas de gráficos así como puntos importantes a tener en cuenta.

Desarrollo de los ambientes de trabajo: en esta parte del trabajo se dará una descripción de las herramientas utilizadas y cómo se enlazan con cada uno de los puntos básicos para llevar a cabo nuestra aplicación.

Propuesta del sistema desarrollado: una vez establecido cada aspecto básico y cómo se lleva acabo con las herramientas de desarrollo, se desarrolla el porqué de la aplicación, cada parte importante del mismo y cómo se llevó a cabo.

Pruebas y resultados obtenidos en la aplicación: al haber concluido con la aplicación, esta se debe poner a prueba, y por lo mismo este capítulo describe las pruebas así como los resultados de la misma en cada prueba preestablecida.

Por último las conclusiones y el trabajo a futuro, los cuales nos dirán si los objetivos fueron alcanzados.

Llevar a cabo una simulación realista es posible gracias al apoyo de nuevas tecnologías que permiten la creación de modelos y aplicaciones que permiten crear la simulación, en este caso Unity; el cual logra integrar todos los modelos, debido a una gran cantidad de herramientas para así poder exportarla a diferentes plataformas.

1.3. Contribución de la tesis.

Conforme avanza la tecnología, las aplicaciones para crear simulaciones son más robustas, tanto que incluso pueden simular situaciones específicas que al ser comparadas con las reales casi no se notaría la diferencia, dichas situaciones cuentan con gran detalle en los modelos utilizados, lo cual demanda una gran cantidad de recursos de hardware. Por lo tanto, entre más personajes, las simulaciones que requerían mas recursos.

Cabe destacar que la implementación y adecuación de un algoritmo de obtención de ruta (PRM) en el motor escogido Unity ya que no solo da la opción de implementar el algoritmo; Unity tiene su propio algoritmo de obtención de ruta.

Es por eso que tanto el ambiente, como los personajes, así como sus animaciones e interacciones con otros personajes, deben ser planeados con diferentes algoritmos que mejoren el uso de los recursos, con el fin de poder tener una población lo suficientemente grande para cubrir nuestro ambiente, pero sólo lo suficiente para poder mostrar en tiempo real la simulación sin perder realismo y así tener una muchedumbre con gran detalle y autónoma interactuando consigo misma con el fin de que cada personaje escape de un incendio.

El hecho de que tanto las herramientas sean aprovechadas y los algoritmos sean implementados de la mejor manera en Unity, mejora mucho lo que cada personaje es capaz de hacer, además de mejorar el aprovechamiento de los recursos y la integración de cada una de las herramientas que tiene Unity, las cuales al trabajar con estándares crean un ambiente de trabajo que facilita la unión de diversos modelos.

2. Estado del Arte.

Actualmente concurren diversos trabajos tecnológicos y conceptuales, los cuales originaron a los Mundos Virtuales (MV) existentes.

Los antiguos pensadores como Platón y Descartes ya tenían una visión acerca de la construcción de una virtualidad presentada como realidad ante nuestra mente; Platón exponía la "alegoría de la cueva" en donde plantea que nuestra interpretación del mundo es una realidad virtual que nuestro pensamiento asume como verdadera. También Descartes, precursor de la teoría del "cerebro en la cubeta", se cuestionó si acaso un demonio afectó su percepción de la realidad para engañar a su juicio.

Uno de los principales avances del siglo XX fue el concepto de Realidad Virtual (RV) que nace en 1965 como resultado del trabajo de Iván Sutherland en su artículo "The ultimate display", siendo en 1968 cuando se creó el primer sistema de realidad virtual que consistía en un casco que permitía al usuario ocupar el mismo espacio entre objetos virtuales. Aunque, se considera que el primer acercamiento a la RV se dio con Morton L. Heilig, quien en 1957 creó una máquina llamada Sensorama, la cual le ofrecía al espectador la posibilidad de presenciar una película en tercera dimensión y, simultáneamente, percibir olor, un sonido estereofónico, vibraciones a través del asiento, como viento en el rostro; con lo cual se creaba una ilusión de realidad. Dicha invención nos da una idea de los esfuerzos que se han realizado para generar espacios sensoriales virtuales con la creación de escenarios a voluntad.

En inicios de los setenta (Murillo A, 1996), el desarrollo de sistemas simuladores de vuelo fue la siguiente frontera, el objetivo de no arriesgar las vidas de los aspirantes a pilotos era el principal aliciente.

Hacia la década de los ochenta, la realidad virtual era considerada como una tecnología viable, ya existían guantes y visores como dispositivos para la interacción con los sistemas de RV.

Para los noventa, la industria se encontraba formalizada y distintos actores como Sun, Sense, Industrias W, División, Silicon Graphics e IBM lanzaron sus plataformas y sistemas para la creación y ejecución de escenarios de RV.

Los orígenes de los MV en la Internet nacen con VRML (Lenguaje de modelado de realidad virtual por sus siglas en inglés) en 1994, impulsado por VRML Consortium, es un lenguaje cuyo objetivo es la creación de objetos tridimensionales interactivos; su estructura es jerárquica de marcas que utilizan nodos, eventos y campos para construir escenarios virtuales reales estáticos y dinámicos.

La madurez del lenguaje se consolida en su versión VRML97 implementándose en el mercado desde 1997 hasta el 2002. Desde 1998 hasta 2003 a la par se desarrolló el lenguaje X3D (extensible 3D) basado en su precursor VRML, pero integrado con el lenguaje XML, lo cual permitió una correcta integración con los distintos servicios web debido a su flexibilidad en la transmisión de datos a distintas aplicaciones.

No sólo las disertaciones filosóficas o los logros tecnológicos han aportado a la conceptualización de los MV, también lo han hecho las obras literarias. La imaginación de escritores de ciencia ficción ha estimulado la imaginación de

los desarrolladores de tecnología desde hace poco más de un siglo. Como ejemplo tenemos la novela del género Cyberpunk Neuromancer de William Gibson y Snow Crash de Neal Stephenson donde se plantean conceptos como cibertualidad y metaverso, ambos términos han servido para dar una visión al imaginario de este tipo de ambientes.

Otro de los principales impulsores del desarrollo de tecnologías que originaron los MV fueron los videojuegos; en 1979 las universidades con acceso a internet empezaron a popularizar los juegos MUD (Multiuser Dungeon) que eran juegos de rol fantásticos en interfaces textuales. A mediados de los noventa, Ultima Online un juego multiusuario online tuvo gran impacto debido a sus gráficos que transportaban al usuario a un ambiente virtual. Fue a fines del año 2004 que el lanzamiento de World of Warcraft un MMORPG (Massively Multiplayer Online Role Playing Games) cautivó a los usuarios por sus gráficos 3D, interacción y comunidades de usuarios.

En junio del 2003, uno de los principales hitos en cuanto a MV sucedieron, la empresa Linden Lab, lanzó Second Life (SL), un Mundo Virtual 3D multiusuario representados por avatares controlados por seres humanos detrás de una computadora, el mundo posee la característica de tener una economía y moneda propia llamada Linden Dollars (L$). El MV tuvo buena aceptación, pero lento crecimiento, pero fue desde 2005 que tuvo un crecimiento exponencial llegando a un millón de usuarios en 2006, a los 6.16 millones en mayo de 2007 y en enero de 2010 se reportaron 18 millones de cuentas. La empresa responsable de Second Life reportó en 2011 ganancias por más de 75 millones de dólares. Actualmente cuenta con 32,871,977 residentes y con más de 60,000 usuarios conectados al mismo tiempo.

Los logros desarrollados por los esfuerzos de Linden, llevo a considerar a Second Life como una plataforma de apoyo a actividades académicas online, e incluso la empresa ofrece espacios y contratos específicos para instituciones educativas y empresas comerciales.

Una característica que comparten los mundos virtuales multiusuario como es el caso de Second Life o World of Warcraft, y que se ha extendido con las redes sociales, es la socialización. Lorca y Pujol (2008) indican que las herramientas informáticas ayudan al usuario a desarrollarse en tres ámbitos: comunicación donde se pone en común conocimientos; comunidad o conformación de grupos y cooperación como la realización de acciones conjuntas. Orihuela (2008) retoma estos ámbitos y los explica como niveles donde, para la creación de comunidades, es necesario contar previamente con la comunicación y colaboración.

Este tipo de MV permite el establecimiento de relaciones, ya sea por intereses, amistad, ideas o parentescos donde se puede intercambiar bienes (digitales como la compra de espacios u objetos virtuales) o servicios que permite la interacción entre usuarios como iguales.

Desde el punto de vista antropológico las características de una comunidad son el sentido de pertenencia, exclusión, e identidad. En los nuevos espacios o mundos virtuales, los individuos formulan su propia identidad y fortalecen su idea de pertenencia. Hay quienes consideran este tipo de comunidades como una evolución a las comunidades tradicionales; esta nueva forma de socialización toma los mundos virtuales (y en general el ciberespacio) como lugar de socialización, donde las comunidades son espacios para compartir

conocimientos y aprendizajes, cuya permanencia en el tiempo dependerá de la cohesión del grupo y de las características y objetivos del mismo.

Para Feri las nuevas comunidades (virtuales) no son opuestas o sustitutas de las comunidades presenciales, sino una nueva modalidad del actuar comunicativo. Un aspecto crítico es la construcción de la identidad (donde uno puede pasar anónimamente o construir nuevas identidades) que en diversos mundos virtuales se construyen a partir de la generación de avatares y constituyen un nuevo yo a partir de la interacción social virtual.

Después del éxito de Second Life, en 2007 surgieron muchos proyectos inspirados en sus logros como Wonderland de Sun Microsystems (actualmente Oracle-Sun), MV de código abierto y basado en tecnologías JAVA que se encuentra al día de la presente publicación en la versión 0.6; Opensimulator otro MV de código abierto desarrollado por programadores voluntarios quienes aplicaron ingeniería inversa de la liberación del cliente de conexión con SL en enero de 2007, en julio del mismo año obtuvieron su primera versión funcional que es compatible con SL; por último Lively de Google, desarrollaron micro mundos 3D que podían albergar hasta veinte usuarios con la característica de poderlos incrustar (embebidos) en sitios externos al servidor, este proyecto se finalizó el 31 de diciembre de 2008, pero han sido desarrollados proyectos similares, como NewLively (www.newlively.com) o como Vivaty el cual permite integrar un MV en redes sociales como facebook o twitter.

Otra de las evoluciones provenientes de los MV y los más recientes son los MMOLE (Massively Multilearner Online Learning Environment) que podría traducirse como Ambiente de Aprendizaje en línea Masivo para Multiaprendices, su principal objetivo es el aprendizaje a través de salones

virtuales para múltiples usuarios. El ejemplo actual más representativo es el software comercial llamado ProtoSphere de la empresa ProtonMedia (http://protonmedia.com).

Beloola es otro servicio que provee mundos virtuales donde es posible generar espacios enriquecidos (sin la necesidad de instalar un software), que funciona directamente en el navegador a través de WebGL11.

Para el futuro de los MV, la Aceleration Studies Foundation (ASF http://www.accelerating.org/) en conjunto con diversas asociaciones, desarrolló un documento titulado "Metaverse Roadmap" (Smart, Cascio, & Paffendorf, 2007) el cual pretende ser un referente [1].

2.1. Animación conductual de grupos y muchedumbre.

Los seres humanos son posiblemente las criaturas más complejas que se conocen, por lo que también son las criaturas más complejas de simular. Una animación conductual de las multitudes humanas (humanoides) se basa en fundamentos de simulaciones grupales de entidades mucho más simples, en particular las bandadas de aves y escuelas de peces. Amkraut, Girard y Karl llamaron a Eurhythmy, la primera animación de procedimiento de las aves de los pájaros virtuales, para la cual el primer concepto fue presentado en The Electronic Theatre en SIGGRAPH en 1985 (la versión final fue presentada en Ars Electronica en 1989). El movimiento de flujo se logró mediante un campo de fuerza vectorial global que guía un flujo de flujos. En su trabajo pionero, Reynolds describió un modelo de comportamiento distribuido para simular el movimiento agregado de una bandada de aves. El documento técnico fue acompañado por un cortometraje animado llamado "StanleyandStella in: Breakingthe Ice", que se muestra en el Electronic Theater at SIGGRAPH ’87. La idea revolucionaria era que el comportamiento complejo de un grupo de actores puede obtenerse mediante reglas locales simples para los miembros del grupo en lugar de alguna condición global impuesta. El flujo se simula como un sistema complejo de partículas, con las aves simuladas (llamadas boids) como partículas. Cada boid se implementa como un agente independiente que navega de acuerdo con su percepción local del entorno, las leyes de la física simulada y el conjunto de comportamientos. Los boids intentan evitar las colisiones entre sí y con otros objetos en su entorno, combinan velocidades con compañeros cercanos de la flota y se mueven hacia el centro de la misma. El movimiento

agregado de la fuga simulada es el resultado de la interacción de estos comportamientos relativamente simples de las aves simuladas individuales. Más tarde, Reynolds extendió su trabajo al incluir varios comportamientos de dirección como búsqueda de objetivos, evitación de obstáculos, seguimiento de camino o vuelo, e introdujo un sencillo controlador de comportamiento de máquinas de estado finito y optimizaciones de consultas espaciales para la interacción en tiempo real con grupos de caracteres. Xiaoyuan Tu y Demetri Terzopoulos propusieron un marco para la animación de los peces artificiales, además de los comportamientos individuales complejos basados en la percepción del entorno, los peces virtuales han exhibido movimientos colectivos sin guion como comportamientos de evasión escolar y de depredadores análogos al intercambio de votos para ello utilizaron una combinación de sistemas de partículas y redes de transición para modelar muchedumbre para la visualización de espacios urbanos. En el nivel inferior, las fuerzas atractivas y repulsivas, análogas a las eléctricas físicas, permiten a las personas moverse por el entorno. Las metas generan fuerzas atractivas, los obstáculos generan campos de fuerza repulsiva. El comportamiento de nivel superior está modelado por redes de transición con transiciones que dependen del tiempo, visitas a ciertos puntos, cambios en las densidades de la población local y eventos globales. Brogan y Hodgins simularon comportamientos grupales para sistemas con dinámicas significativas. En comparación con los boids, se logra un movimiento más realista al tomar en cuenta las propiedades físicas del movimiento, como el impulso o el equilibrio. Su algoritmo para controlar los movimientos de las criaturas procede en dos pasos: primero, un modelo de percepción determina las criaturas y los obstáculos visibles para cada individuo, y luego un algoritmo de colocación determina la posición deseada

para cada individuo, dadas las ubicaciones y velocidades de las criaturas y obstáculos percibidos. Los sistemas simulados incluían grupos de robots con una sola pierna, ciclistas y sistemas de masa puntual.

Musse y Thalmann presentaron un modelo jerárquico para la simulación en tiempo real de multitudes humanas virtuales. Su modelo se basa en grupos, en lugar de individuos: los grupos son estructuras más inteligentes, donde los individuos siguen la especificación de los grupos. Los grupos se pueden controlar con diferentes niveles de autonomía: la muchedumbre guiada sigue (como ir a un lugar determinado o reproducir una animación en particular) dada por el usuario en tiempo de ejecución; la muchedumbre programadas siguen un comportamiento con guion; y la muchedumbre autónoma usan eventos y reacciones para crear comportamientos más complejos. El entorno comprende un conjunto de puntos de interés, que representan objetivos y puntos de paso; y un conjunto de puntos de acción, que son objetivos que tienen algunas acciones asociadas. Los agentes se mueven entre los puntos de paso siguiendo las curvas de Bezier. Recientemente, otro trabajo fue la exploración de un modelo basado en las jerarquías. Niederberger y Gross propusieron una arquitectura de agentes jerárquicos y heterogéneos para aplicaciones en tiempo real. El comportamiento se redefinió mediante la especialización de los tipos de comportamiento existentes y la herencia múltiple ponderada para la creación de nuevos tipos. Los grupos se definen a través de patrones recursivos y basados en módulo. El motor de comportamiento permite la especificación de una cantidad máxima de tiempo por ejecución para garantizar una velocidad de cuadros mínima y constante. Ulicny y Thalmann presentaron una simulación de comportamiento de multitud con una arquitectura modular para un sistema multiagente que permite un comportamiento autónomo y con secuencias de comandos de

agentes que admiten variedad. En su sistema, el comportamiento se calcula en capas, donde las decisiones se toman mediante reglas de comportamiento y la ejecución se maneja mediante máquinas jerárquicas de estado finito. Más recientemente, ha propuesto un modelo de muchedumbre en tiempo real basado en dinámicas continuas. En su modelo, un campo de potencial dinámico integra la navegación global con obstáculos en movimiento, resolviendo de manera eficiente el movimiento de grandes muchedumbres sin la necesidad de evitar explícitamente la colisión. La complejidad percibida de la simulación de muchedumbre se puede aumentar usando niveles de detalle (LOD). O'Sullivan et al. describió una simulación de multitudes y grupos con nivel de detalles de geometría, movimiento y comportamiento. En el nivel geométrico, las técnicas de subdivisión se utilizan para lograr una reproducción suave de los cambios de LOD. En el nivel de movimiento, los movimientos se simulan utilizando niveles adaptativos de detalle. Los subsistemas de animación con diferentes complejidades, como un reproductor de fotogramas clave o un módulo de tiempo real, se activan y desactivan en función de las heurísticas. Para el comportamiento, LOD se emplea para reducir los costos computacionales de actualizar el comportamiento de los caracteres que son menos importantes. Los personajes más complejos se comportan de acuerdo con sus motivaciones y roles, los menos complejos solo juegan fotogramas clave aleatorios [2].

2.2. Avance del modelado del entorno para muchedumbre.

El modelado del entorno está estrechamente relacionado con la animación de comportamiento. El propósito de los modelos del entorno es facilitar la simulación de entidades que habitan en sus entornos circundantes. La credibilidad de las criaturas virtuales puede mejorarse enormemente si se comportan de acuerdo con su entorno. Por el contrario, el suspenso de la creencia puede ser destruido inmediatamente si realizan algo que no se espera o no se permite en el mundo real, como pasar por la pared o caminar sobre el agua. Por lo tanto, los mayores esfuerzos han sido dirigidos a representaciones y algoritmos que evitan que ocurran conductas "prohibidas"; hasta hace muy poco tiempo, los dos principales problemas de inteligencia artificial relacionados con la industria del desarrollo de juegos eran la prevención de colisiones y la planificación de caminos. La mayoría de la población en el mundo desarrollado vive en ciudades; Es allí donde la mayoría de las actividades humanas tienen lugar hoy en día. En consecuencia, la mayor parte de la investigación se ha realizado para el modelado de ciudades virtuales. Farenc et al. Introdujo un entorno informado dedicado a la simulación de humanos virtuales en el contexto urbano. El entorno informado es una base de datos que integra información semántica y geométrica sobre una ciudad virtual. Se basa en una descomposición jerárquica de una escena urbana en entidades ambientales, como barrios, bloques, uniones, calles, etc. Las entidades pueden contener una descripción de los comportamientos que son apropiados para los agentes ubicados en ellos; por ejemplo, una acera dice que se debe caminar, o

un banco le dice que se debe sentar. Además, la base de datos del entorno se puede usar para un trazado de un camino que se personaliza de acuerdo con el tipo de cliente que solicita el camino, de modo que, por ejemplo, un peatón obtendrá caminos que usen aceras, pero un automóvil tendrá caminos que pasen por carreteras. Otro modelo de una ciudad virtual para una animación de comportamiento fue presentado por Thomas y Donikian. Su modelo está diseñado con el énfasis principal en la simulación de tráfico de vehículos y peatones. La base de datos del entorno se divide en dos partes: una estructura jerárquica que contiene un árbol de regiones poligonales, similar a la base de datos del entorno informado; y un Estructura topológica con una gráfica de una red viaria. Las regiones contienen información sobre las direcciones de circulación, incluidos los posibles cambios de un camino en las intersecciones. Los agentes luego usan la base de datos para navegar por la ciudad. En un trabajo reciente, Sung et al. presentó un nuevo enfoque para controlar el comportamiento de una multitud al almacenar información de comportamiento en el entorno utilizando estructuras denominadas situaciones. En comparación con los enfoques anteriores, las estructuras (situaciones) ambientales pueden superponerse; los comportamientos correspondientes a tales situaciones de aplicación superan las compuestas utilizando distribuciones de probabilidad. Las funciones de comportamiento definen las probabilidades de transiciones de estado (activando clips de movimiento) en función del estado de las características del entorno o del estado pasado del agente. En el lado centrado en cuestiones de planificación de un camino más genéricas, se han realizado varios trabajos. Kallmann et al. Propuso un algoritmo de planificación de un camino rápido basado en una triangulación de Delaunay completamente dinámica. Bayazit et al. usó planes de trabajo globales para mejorar los

comportamientos de grupo en entornos geométricamente complejos. Los grupos de criaturas exhibieron comportamientos tales como la búsqueda del hogar, la búsqueda de objetivos, la cobertura o el pastoreo, mediante el uso de reglas integradas tanto en los miembros de la flota individual como en los planes de trabajo. Tang et al. usó un algoritmo A * modificado que trabajaba en una cuadrícula superpuesta sobre un terreno generado por un mapa de altura. Recientemente, Lamarcheand Donikian presentó una estructura topológica del entorno geométrico para una planificación jerárquica de caminos rápida y un algoritmo de navegación reactivo para multitudes virtuales. Más recientemente, el trabajo presentado por Pettré et al. muestra cómo calcular de forma automática y robusta una exploración de varios niveles mediante gráficos de cilindros tridimensionales. Este trabajo también muestra cómo reutilizar el cálculo de planificación de un camino resultante para unos pocos cientos de agentes que pueden reaccionar a la congestión a lo largo del camino [2].

3. Marco teórico.

3.1. Mundos virtuales.

Genéricamente, podemos decir que los ambientes virtuales son aplicaciones que pueden ejecutarse en red. Permiten la colaboración, aprendizaje y simulación en diferentes escenarios, tales como la medicina, el arte, la arquitectura, la educación, etc. Son ambientes que posibilitan la recreación de espacios compartidos, donde el usuario se presenta como un avatar que puede interactuar con el escenario y con otros usuarios representados de la misma manera, recreando una sociedad virtual. En educación, están siendo utilizados en distintas regiones tales como Norteamérica, Europa y Asia. Los temas propuestos para la investigación se centran especialmente en: modelización científica, desarrollo de aplicaciones para aulas virtuales 3D, percepción del sujeto (residente virtual), procesos comunicativos y otros [3].

3.1.1. ¿Qué ofrecen los ambientes virtuales?

Reúnen distintos grupos de usuarios dentro del mundo virtual sin necesidad de desplazarse a un lugar físico.

- Admiten la incorporación de contenidos de aprendizaje en distintos formatos (videos, textos, fotos, etc.).
- Son persistentes, o sea que el entorno sigue existiendo, aunque el usuario no esté conectado.

- El elemento clave es la sensación de presencia y actividad que obtiene el usuario donde la carga visual es más fuerte que la textual.
- Permiten el aprendizaje, creación y exploración de modelos tridimensionales. La persona es la protagonista adoptando un rol activo a través del avatar.
- Los escenarios virtuales nos dan la posibilidad de recrear situaciones que ofrezcan riesgos de vida, permitiendo probar realidades complejas en las cuales no se pone en peligro la seguridad del usuario.
- Pierre Lévy señala: "Las técnicas de simulación, en particular las que ponen en juego imágenes interactivas, no reemplazan los razonamientos humanos, sino que prolongan y transforman las capacidades de imaginación y de pensamiento".

Estos entornos virtuales son parte de la Web 3.0 y sobre la base de sus posibilidades de comunicación y cooperación ayudarían a crear nuevas oportunidades en los procesos formales y no formales de adquisición de conocimiento.

Para este trabajo se planteó un escenario en el cual se puedan mostrar técnicas de RCP (resucitación- cardio- pulmonar). De allí la importancia del estudio del avatar y las técnicas asociadas para dar movimiento al mismo [3].

3.2. Muchedumbre en mundos virtuales.

En este contexto, deben abordarse diversos aspectos de la simulación, incluida la animación de comportamiento, el modelado de entornos y la representación

de muchedumbre. Si no hay una representación satisfactoria, incluso el mejor modelo de comportamiento no será muy convincente. Si no hay un buen modelo de comportamiento, incluso una simulación con el mejor método de renderización parecerá estúpida después de unos pocos segundos. Si no hay un modelo apropiado del entorno, los personajes no se comportarán de manera creíble, ya que realizarán acciones en los lugares equivocados o no se realizarán en absoluto. El objetivo de la animación conductual es facilitar el trabajo de los diseñadores al permitir que los personajes virtuales realicen movimientos complicados de forma autónoma o semiautónoma que, de lo contrario, requerirían grandes cantidades de trabajo de animadores humanos; o, en el caso de aplicaciones interactivas, los modelos de comportamiento permiten que los personajes respondan a las acciones iniciadas por el usuario. Para que un comportamiento tenga sentido, además de los personajes, su entorno debe ser modelado, no solo gráficamente sino también semánticamente. De hecho, un repertorio de posibles comportamientos depende en gran medida de lo que es y lo que no está incluido en un modelo del entorno. Sucede muy a menudo que el entorno es visualmente rico, pero la interacción de los personajes con él es mínima.

Finalmente, para aplicaciones interactivas, es necesario mostrar un conjunto variado de caracteres virtuales de una manera eficiente. Los personajes renderizados deben "ajustarse" visualmente al entorno, deben verse afectados por la luz y otros efectos de la misma manera que su entorno. A continuación, presentaremos trabajos representativos para cada uno de estos temas agrupados de acuerdo con su enfoque principal [2].

3.3. Requisitos y restricciones para el modelado de muchedumbre.

La muchedumbre en tiempo real trae desafíos diferentes en comparación con los sistemas que involucran un pequeño número de personajes interactivos (por ejemplo, la mayoría de los juegos de computadora contemporáneos) o aplicaciones en tiempo no real. En comparación con las simulaciones de agente único, la principal diferencia conceptual es la necesidad de una gestión eficiente de la variedad en todos los niveles, ya sea visualización, control de movimiento, animación o reproducción de sonido. Como lo indican las experiencias cotidianas, los humanos virtuales que forman una multitud deben verse diferentes, moverse diferentes, reaccionar de manera diferente, sonar diferente, y así sucesivamente. Incluso si asumir la simulación perfecta de un solo humano virtual sería posible, crear una simulación que involucre a múltiples humanos de este tipo sería una tarea difícil y tediosa. Se necesitan métodos que faciliten el control de muchos personajes; sin embargo, tales métodos aún deben preservar la capacidad de controlar agentes individuales. En comparación con las simulaciones no en tiempo real, el desafío técnico principal es la mayor demanda de recursos computacionales, ya sea su capacidad de procesamiento general, rendimiento de gráficos o espacio de memoria. Uno de los factores de restricción más importantes para las simulaciones de multitud en tiempo real es la representación de muchedumbre. Métodos rápidos y escalables tanto para calcular el comportamiento, como para tener en cuenta elementos de entrada desconocidos de antemano y para generar grandes y variadas multitudes.

Si bien las simulaciones en tiempo no real pueden aprovechar la ventaja de conocer una ejecución completa del escenario simulado (y, por lo tanto, por ejemplo, pueden ejecutarse de manera iterativa sobre varias opciones posibles seleccionando la mejor solución global), las simulaciones en tiempo real tienen que reaccionar a la situación como se desenvuelve en el momento [2].

3.4. Modelo del personaje virtual.

Se pueden definir distintos aspectos de la animación de un avatar, los cuales constituyen una jerarquía cuya base se encuentra en la computación gráfica y a partir de la cual se ha trazado una evolución hasta llegar a la denominada Vida Artificial. Según lo describe Terzopoulos los aspectos que conforman la animación de un avatar, son:

- Geométrico
- Cinemático
- Físico
- De comportamiento
- Cognitivo

Dentro de lo que se conoce como computación gráfica, uno de los primeros logros en materia de animación fue la combinación de modelos geométricos que

empleaban cinemática directa e inversa para la animación. Luego se agregaron principios físicos que permitieron obtener movimientos más realistas mediante

simulación dinámica, desarrollando modelos físicos para animar partículas, cuerpos sólidos, fluidos y gases. El nivel más evolucionado del aspecto físico involucra modelos biomecánicos que en conjunto con el comportamiento y el aspecto cognitivo conforman lo que dio en llamarse Vida Artificial (avatares similares al ser humano y a la vida real, tanto como sea posible). El aspecto de comportamiento de la animación se obtiene mediante avatares autónomos que reaccionan a estímulos del ambiente. El aspecto cognitivo de la animación de avatares es el resultado de la aplicación de técnicas de IA (Inteligencia Artificial), las cuales comprenden básicamente: El razonamiento, la representación de conocimiento y la planificación. Al emplear el aspecto cognitivo en la animación, se obtienen avatares que poseen libre albedrío y la capacidad de tomar decisiones.

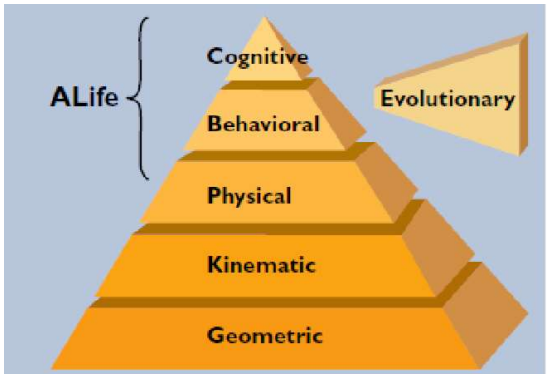


Fig. 3.1 Jerarquía de animación de avatares.

Es posible que las regiones del mundo virtual no reciban un gran número de visitas de usuarios, por lo que para generar la idea de que una región se

encuentra habitada se recurre a los denominados NPC (Non-Player Character) [3].

3.5. El problema de planificación.

El problema de planificación de movimientos ha originado un número importante de trabajos de investigación durante los últimos años en robótica. En su forma más simple, es un problema intrínsecamente difícil.

Al inicio de los años 90 aparecieron los métodos de planificación aleatoria de movimientos. Estos métodos, introducen la aleatoriedad para desaparecer la noción de completitud en lugar de la completez en probabilidad: si una solución existe, la probabilidad de éxito del algoritmo tiende hacia 1 cuando el tiempo tiende hacia el infinito.

Durante la década pasada, los planificadores basados en roadmaps probabilísticos se han convertido en una herramienta poderosa para la planificación de movimientos de robots con muchos grados de libertad. La idea central del PRM clásico es llevar a cabo un muestreo aleatorio del espacio de configuraciones de un robot para construir una red de caminos llamada roadmap. Esta red captura la conectividad del espacio libre por medio de un grafo o un árbol. Los planificadores de tipo PRM son simples de implementar y eficientes para múltiples aplicaciones [2].

3.6. UNITY.

Unity es una herramienta de integración que está pensada para unir contenido; ya sea arte, programación y diseño, para asi crear un videojuego de manera rápida, cabe dejar en claro que Unity no se trata de una herramienta para crear contenido como modelos de casas, personajes, etc. pues para eso están los diferentes programas en cada una de las respectivas áreas.

Unity está pensado para un desarrollo multiplataforma, de tal manera que después de trabajar con este motor se pueda exportar a diferentes plataformas, como plataformas móviles u ordenadores de sobremesa como se muestra en la siguiente figura.

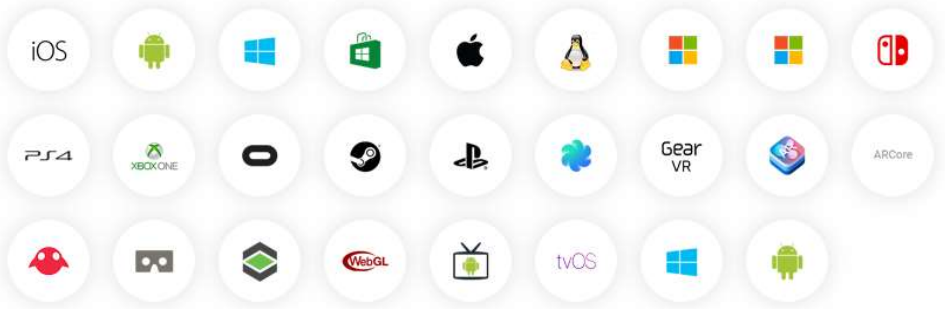


Fig. 3.2. Unity sistemas a exportar.

Unity cuenta con una versión de acceso gratuito muy completa que ayuda de gran manera si se va a crear videojuegos, simulaciones, etc. Solo cobraría si se supera los $100k por lo cual se da un gran margen mara la investigación.

Cuenta con una comunidad de usuarios muy importante por lo cual se puede encontrar en la red mucha información sobre cómo generar contenido y utilizar dicho programa. En este tema y en el siguiente, vamos a tratar la interfaz básica de este motor, de este programa [4].

3.7. Animación en Unity.

Unity tiene un sistema de animación excelente y sofisticado llamado Mecanim que proporciona:

- Un flujo de trabajo y configuraciones de animaciones fácil para todos los elementos de Unity incluyendo objetos, personajes, y propiedades.

- Soporte para importar clips animaciones y animaciones creados con Unity.

- Animación humanoide retargeting - habilidad para aplicar animaciones del modelo de un personaje a otro.

- Un flujo de trabajo simplificado para alinear clips de animación.

- Una previsualización conveniente de clips de animación, transiciones e interacciones entre estos. Esto les permite a los animadores funcionar de manera más independiente a los programados, dar prototipos y

previsualizar sus animaciones antes de que el código del juego es conectado.

- El manejo de interacciones complejas entre animaciones con una herramienta visual de programación.

- Animar diferentes partes del cuerpo con diferente lógica.

- Características de capas (layering) y de masking.

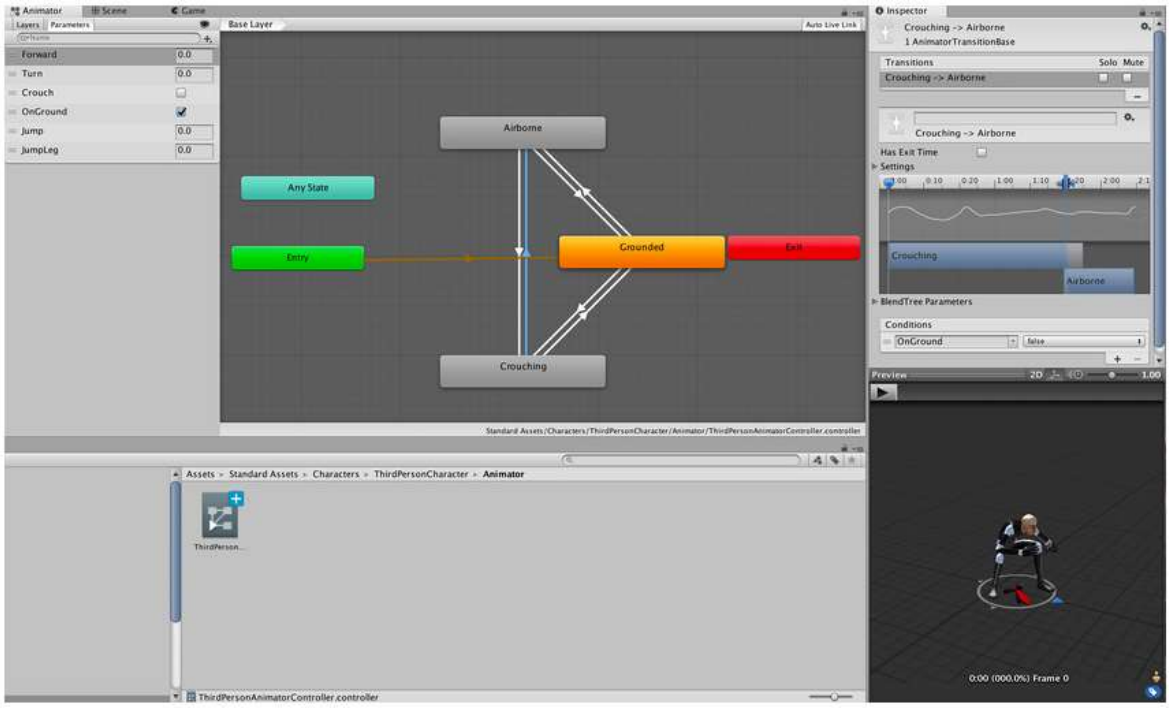


Fig. 3.3. Animación Unity

3.8. Flujo de trabajo de la Animación

El sistema de animación de Unity se basa en el concepto de clips de animación, que contiene información sobre cómo ciertos objetos deben cambiar su posición, rotación u otras propiedades a lo largo del tiempo. Cada clip puede ser pensado como una sola grabación lineal. Los clips de animación de fuentes externas son creados por artistas o animadores con herramientas de terceros, como Autodesk, 3ds Max, o Autodesk, Maya, o provienen de estudios de captura de movimiento u otras fuentes.

Luego, los clips de animación se organizan en un sistema estructurado similar a un diagrama de flujo llamado Controlador Animator. El controlador animador actúa como una "máquina de estado" que realiza un seguimiento de qué clip debe estar reproduciéndose actualmente, y cuando las animaciones deben cambiar o mezclarse.

Un simple Animator Controller podría solamente contener uno o dos clips, por ejemplo, para controlar un powerup de girar y rebotar, o para animar una puerta que se abre y se cierra en el tiempo correcto. Un más avanzado Animator Controller puede contener docenas de animaciones humanoides para todas las acciones de los personajes principales, y pueden mezclarse entre múltiples clips al mismo tiempo para proporcionar un movimiento fluido a medida que el jugador se mueve alrededor de la escena.

El sistema de animación de Unity también tiene numerosas características especiales para el manejo de personajes humanoides que le brindan la capacidad de redirigir la animación humanoide de cualquier fuente (por ejemplo: captura de movimiento, la Tienda de Activos o alguna otra biblioteca de animación de terceros) a su propio modelo de personaje. Además de ajustar las definiciones musculares. Estas características especiales están habilitadas por el sistema Avatar de Unity, donde los personajes humanoides se asignan a un formato interno común.

Cada una de estas piezas: los clips de animación, el controlador de animador y el avatar, se reúnen en un GameObject a través del componente animador. Este componente tiene una referencia a un controlador animador y (si es necesario) el avatar para este modelo. El controlador de animación, a su vez, contiene las referencias a los clips de animación que utiliza [4].

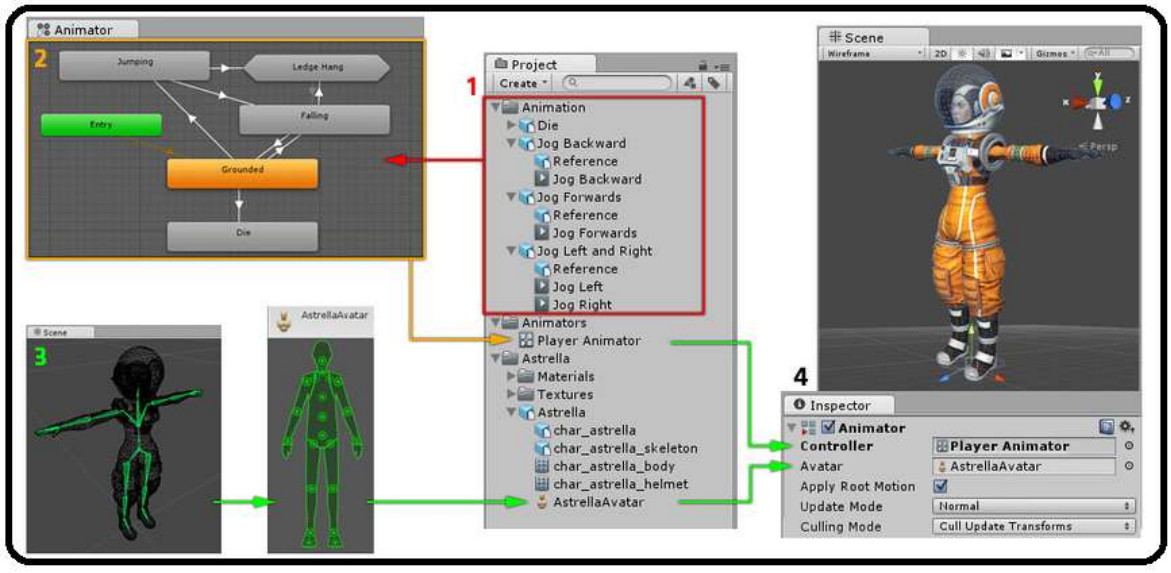


Fig. 3.4. Pasos para animación en Unity

En la figura anterior se describen los siguientes detalles del sistema de animación Unity:

1. Los clips de animación se importan desde una fuente externa o se crean dentro de Unity. En este ejemplo, se importan animaciones humanoides capturadas en movimiento.

2. Los clips de animación son colocados y arreglados en un controlador de animación. Esto es una vista de un controlador de animador en la ventana Animador. Los Estados (los cuales pueden representar animaciones o estados de máquinas anidadas) aparecen como nodos conectados por líneas. Este controlador de animador existe como un activo en la ventana del Proyecto.

3. El modelo del personaje amañado (en este caso, el astronauta “Estrella”) tiene una configuración de huecos, los cuales son mapeados en el formato Avatar común de Unidad. Este mapa está almacenado como un activo Avatar como parte del modelo del personaje importado, y también aparece en la ventana del Proyecto cómo es mostrado.

4. Al animar el modelo de personaje, tiene un componente animador adjunto. En la vista del Inspector que se muestra arriba, puede ver el Componente Animador que tiene tanto el Controlador Animador como el Avatar asignados. El animador los usa juntos para animar el modelo. La referencia de Avatar solo es necesaria cuando se anima un personaje humanoide. Para otros tipos de animación, sólo se requiere un controlador de animador.

5. El sistema de animación de Unity viene con muchos conceptos y terminología.

3.9. Mapas probabilísticos.

Los mapas probabilísticos (PRM), desde sus inicios, han demostrado un tremendo potencial para solucionar el problema de encontrar una ruta libre de colisiones en entornos multidimensionales .La idea central del PRM es distribuir un conjunto de puntos (nodos) de manera aleatoria en el espacio de configuración libre de colisiones.

3.10. Roadmaps probabilistas.

El principio subyacente a los métodos PRM es muy simple. Se pretende construir un grafo (o un árbol) G = (V, E) capturando la conectividad del espacio de configuraciones libre (CS_{libre}). Los nodos V son configuraciones libres generadas aleatoriamente según una distribución uniforme. Los arcos de G, almacenados en el conjunto E, son caminos factibles uniendo los nodos de V, construidos por un método local L1.

Los algoritmos de detección de colisiones permiten verificar si las configuraciones consideradas corresponden o no a configuración admisibles para las posiciones correspondientes del robot. La mayor parte de los métodos

PRM, tienen la misma estructura general. El método es habitualmente dividido en dos fases: una fase de aprendizaje y una fase de búsqueda (consulta) [5].

3.11. Métodos locales

Los métodos locales pueden ser deterministas o no deterministas. Un método determinista produce siempre el mismo resultado cuando este se aplica al mismo problema, por el contrario, un método no determinista producir resultados diferentes. El método determinista tiene ventajas que no presenta el no determinista ya que este no necesita almacenar las informaciones adecuadas sobre los caminos en el roadmap, ya que podemos utilizarla para reproducir los caminos a medida que sean necesarios [5].

3.12. Camino lineal

El método local más utilizado es el método lineal. Este método pertenece a la clase de métodos estrictamente deterministas. El camino calculado por este método es siempre una línea recta o una interpolación lineal entre q_i y q_f en el espacio de configuraciones. La Figura 2.2 A), muestra el resultado de un camino producido por este método [5].

3.13. Caminos de tipo A*

Los métodos locales de este tipo son ejemplos de métodos deterministas potentes [1]. Estos utilizan una simple función de potencial para guiar su búsqueda de q_i a q_f. Este potencial permitir al camino de cambiar su curso y de deslizarse a lo largo de la región de los obstáculos del espacio de configuraciones, tanto tiempo como la distancia a q_i no aumente. La Figura 2.2 B), muestra un ejemplo de camino producido por un método local de este tipo. En cada paso, el planificador calcula la distancia entre sus 8 vecinos inmediatos en la rejilla a la configuración final, y procede con la configuración vecina libre de colisión que tiene la distancia más pequeña a la configuración final [5].

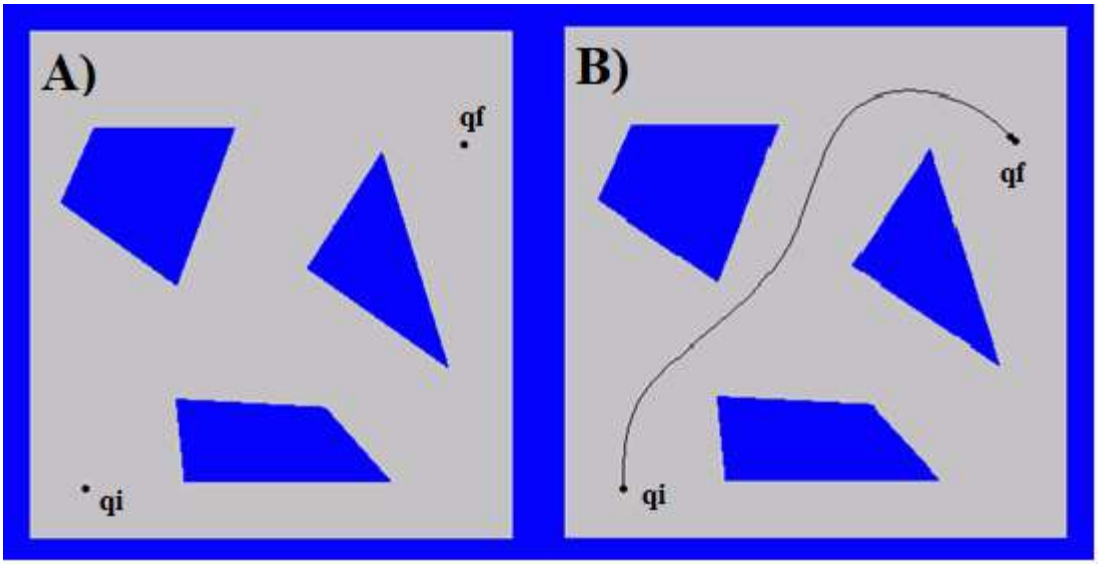


*Fig. 3.5. A) Camino producido por el método local lineal B) Resultado de un camino producido por el método local A**

4. Desarrollo de los ambientes de trabajo.

La creación de simulación de evacuación requiere de una plataforma capaz de entrelazar diferentes áreas para poder converger en una simulación realista, en nuestro caso Unity la cual es de gran ayuda en distintos ámbitos y procesos de la tesis como se mostrará en los siguientes puntos.

- Modelado del escenario: El escenario es una creación que al ser muy detallada se tiene que desarrollar en distintos programas para que el modelo quede muy realista. Para finalmente importarlo a Unity el cual ofrece herramientas para importar el escenario y poder trabajar con él además de poder unir diferentes modelos y humanoides en un conjunto con el escenario.

- Personajes: En este punto se tomará en cuenta cómo se hacen los modelos de personas y cómo se obtiene las animaciones para cada uno de los modelos previamente creados y elegidos. Por último, todo es exportado a Unity el cual permite el manejo de diferentes animaciones para distintos personajes por medio de la modalidad humanoide para cada personaje, por medio de estándares de animación.

- PRM: Es un algoritmo que se usa para la obtención de caminos, implementado para el uso de un RoadMap creado a través de un grid que ante el uso de A* obtiene un camino, que al ser refinado el personaje puede empezar a moverse por él.

- Obtención del camino: El camino para cada personaje es indispensable para que este pueda moverse, debido a esto Unity es de gran valor al permitir la creación de algoritmos para la obtención de un camino, además de tener su propio algoritmo para una optimización en tiempo real.

- Animación: Cada personaje debe contar con una animación, la cual tiene estar creada bajo cierto formato, ya que todos los personajes usan la misma animación, una vez obtenidas diferentes animaciones con una herramienta de Unity llamada animator se controlarán con diferentes estados cada una de las animaciones para que dependiendo el estado o situación del personaje se active la animación correspondiente.

- Simulación: con todo lo anterior mencionado antes, Unity es una herramienta muy útil en los diferentes aspectos de la simulación.

- Pruebas: Unity es de gran utilidad en este aspecto ya que permite integrar accesorios y exportar a diferentes plataformas además de proporcionar herramientas que permiten mostrar los datos en ejecución sin interrumpirla.

4.1. Unity como herramienta de trabajo.

Unity como se describe arriba es una gran herramienta para nuestros propósitos, ya que podemos integrar tanto las animaciones, modelos y scripts al mismo tiempo para poder editarlos, pero ¿qué es Unity? Unity es un motor de videojuegos el cual no es una herramienta especializada si no que es una herramienta para integrar todo y lograr un objetivo, ejemplo crear videojuegos o simulaciones en nuestro caso, generalmente los motores de videojuegos existen para el desarrollo propio, que normalmente son hechos por empresas que desarrollan videojuegos. De esta manera, controla con más precisión y detalle la tecnología usada. Por otra parte, existen motores comerciales que son productos que compañías desarrolladoras compran para así poderse ahorrar tiempo en el desarrollo de un motor propio. En este ejemplo, vamos a utilizar Unity, para ver cómo se va integrando el arte dentro de un motor. Unity es un ejemplo de un motor comercial.

4.1.1.Requisitos del sistema para la Unity.

Los requisitos para trabajo con Unity dependen el uso y el objetivo final, en este caso Unity propones dos tipos.

Para desarrollo

OS: Windows 7 SP1+, 8, 10, 64-bit version only; macOS 10.12+

No se han probado las versiones de servidor de Windows & OS X.

CPU: Soporte para el conjunto de instrucciones SSE2.

GPU: Tarjeta de video con capacidad para DX10 (shader modelo 4.0).

El resto depende principalmente de la complejidad de sus proyectos.

Requisitos adicionales para el desarrollo de plataformas:

iOS: Mac computer running minimum macOS 10.12.6 and Xcode 9.4 or higher.

Android: Kit de Desarrollo Android SDK y Java (JDK); el IL2CPP scripting backend requiere Android NDK.

Plataforma Windows universal: Windows 10 (64 bits), Visual Studio 2015 con componente C++ Tools o posterior y SDK para Windows 10 [4].

4.1.2.Para ejecutar juegos de Unity.

Por lo general, el contenido desarrollado con Unity puede ejecutarse bastante bien en todas partes. Qué tan bien se ejecuta depende de la complejidad de su proyecto. Requisitos más detallados:

Escritorio:

OS: Windows 7 SP1+, macOS 10.12+, Ubuntu 16.04+

Tarjeta de video con capacidad para DX10 (shader modelo 4.0).

CPU: compatible con el conjunto de instrucciones SSE2.

El reproductor de iOS requiere iOS 9.0 o superior.

Android: OS 4.1 o posterior; ARMv7 CPU con soporte NEON o CPU Atom; OpenGL ES 2.0 o posterior.

WebGL: Cualquier versión de escritorio reciente de Firefox, Chrome, Edge o Safari

Plataforma Windows universal: Windows 10 y una tarjeta de gráficos con capacidades DX10 (modelo de shader 4.0).

4.1.3.Herramientas de Unity.

Unity cuenta con muchas herramientas y características para cada área de trabajo, como se ha mencionado anteriormente estas son algunas para cada área en el desarrollo de videojuegos.

- Animación.

    Animaciones extraíbles

    Control total de las animaciones en tiempo de ejecución.

    Llamadas de eventos desde la reproducción de la animación.

    Sofisticadas jerarquías y transiciones de máquinas de estados.

Mezcla de formas para animaciones faciales

- Gráficos

  Iluminación.

  Sombreado basado en la física

  Sondas de reflexión

  Sistema modular de partículas impulsado por curvas y gradientes.

  Herramientas de interfaz de usuario intuitivas

- Mejoramiento

  Perfilado avanzado de memoria

  Eliminación de oclusión con potencia de Umbra

  Agrupamiento de activos

  Soporte de nivel de detalle (LOD)

  Tamaño de la estructura de desmontaje

  Sistema de trabajo multihilo

- Audio

  Mezcla y masterización en tiempo real.

Jerarquías de mezcladores, instantáneas y efectos predefinidos.

Aprende más Tutoriales

Física 2D y 3D

Box2D con una amplia gama de efectores, juntas y colisionadores.

NVIDIA® PhysX® 3.4

- Scripting

  C # 7.2.

- Integración nativa de Visual Studio

  Características de AI con búsqueda avanzada de caminos automatizados y mallas de navegación.

El Editor de Unity presenta herramientas múltiples que permiten una edición e iteraciones rápidas en los ciclos de desarrollo, lo que incluye el modo Play para tener vistas previas rápidas del trabajo en tiempo real.

Editor todo en uno: Disponible en Windows, Mac y Linux. Incluye una variedad de herramientas ideales para los artistas que les permite diseñar experiencias y mundos de juegos envolventes, así como un conjunto robusto de herramientas

para desarrolladores destinadas a implementar la lógica del juego y un juego de alto rendimiento.

2D y 3D: Unity apoya tanto el desarrollo de la tecnología 2D como el de la 3D con prestaciones y funcionalidades para necesidades específicas en los diversos géneros.

Herramientas Al pathfinding: Unity incluye un sistema de navegación que permite crear NPC que pueden moverse con inteligencia por el mundo del juego. El sistema usa mallas de navegación que se crean automáticamente a partir de la geometría de tu Escena, o incluso obstáculos dinámicos, para alterar la navegación de los personajes en el tiempo de ejecución.

Flujos de trabajo eficientes: Los Prefabs de Unity, que son Game Objects preconfigurados, proporcionan flujos de trabajo eficientes y flexibles que permiten trabajar con confianza sin la preocupación de cometer errores que consumen mucho tiempo.

Interfaces de usuario: Nuestro sistema UI incorporado permite crear interfaces de usuario de forma rápida e intuitiva.

Motores de física: Aprovecha Box2D, el nuevo sistema de física de pila de tecnología basada en datos (Data-Oriented Technology Stack, DOTS) y la compatibilidad con NVIDIA PhysX para un juego muy realista de alto rendimiento.

Herramientas personalizadas: Puedes expandir el Editor con las herramientas que se necesiten para estar a la par del flujo de trabajo del resto del equipo. Crea y agrega extensiones personalizadas o identifica lo que se necesita en la Asset

Store, que ofrece miles de recursos, herramientas y extensiones para agilizar tus proyectos.

4.2. Creación del mundo Virtual.

El mundo virtual fue creado a través de múltiples herramientas debido a que se necesita un mundo convincente. Primero que nada se usó AC3D, la cual es una herramienta muy poderosa al momento de modelar un modelo, gracias a esto el modelo escogido quedo muy parecido a cómo es en realidad; el escenario escogido es la Biblioteca Central la cual cuenta con múltiples pisos y cuenta como base para futuros proyectos en ella.

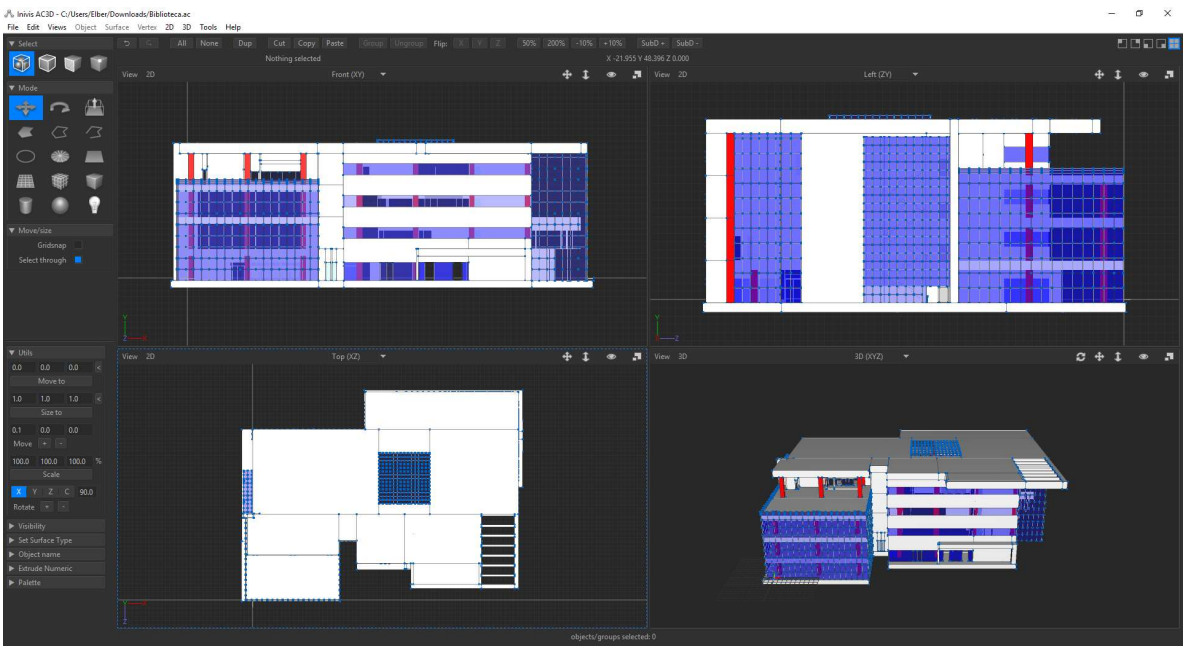


Fig. 4.1. AC3D

Como se puede ver en la siguiente figura el modelo es realista y muy detallado, también se optó por mantener los colores parecido a los planos.

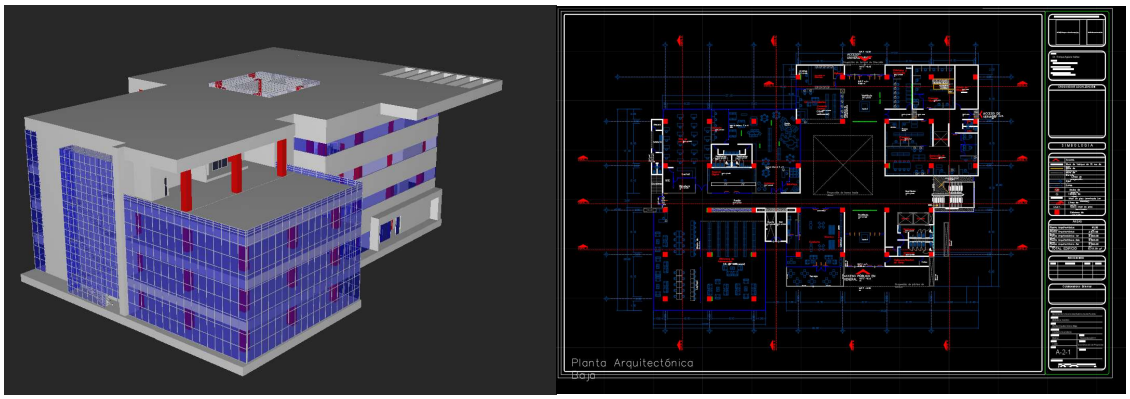


Fig. 4.2. Modelo y plano

Una vez concluido el modelo toca texturizarlo para que quede lo más parecido a la realidad, eso se obtendrá gracias al uso de sketchup, el cual gracias a sus herramientas ayuda a texturizar el modelo.

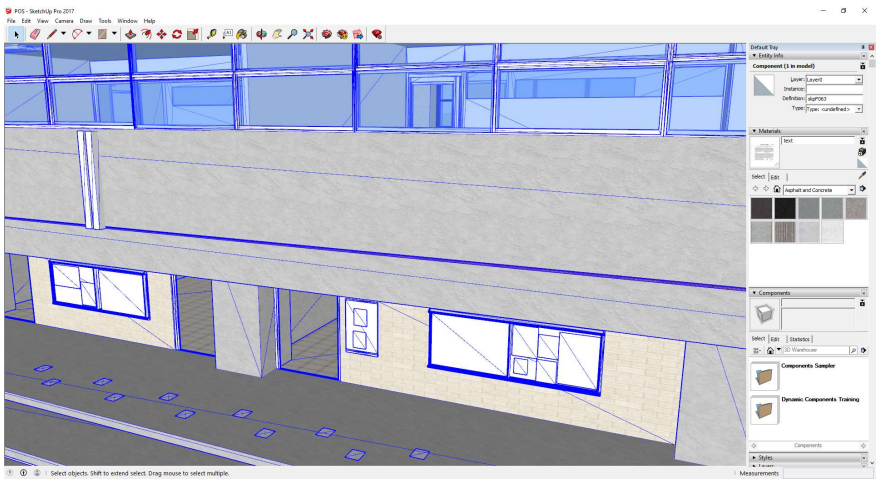


Fig. 4.3. Sketchup

Gracias a esto una vez obtenido el modelo se importará al proyecto de Unity para poder trabajar con él.

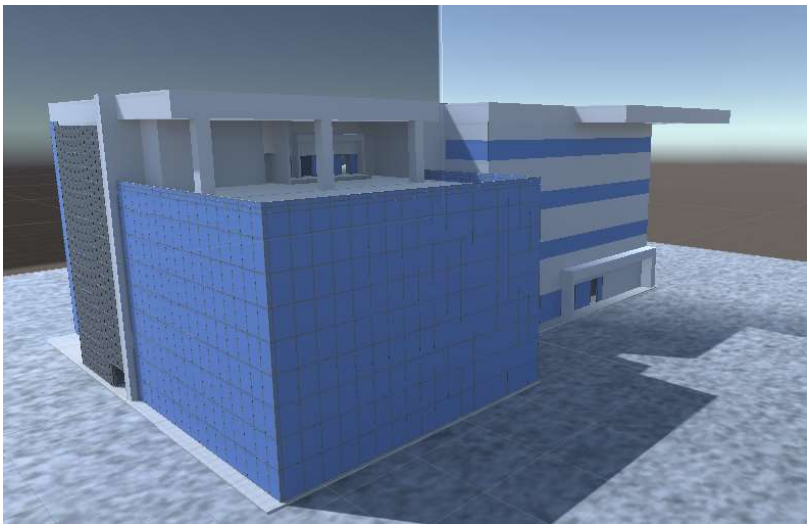


Fig. 4.4. Modelo final.

4.3. Creación de los Modelos.

El poder crear modelos con rasgos únicos y diferentes para cada edad y genero ayuda a crear una población muestra de calidad, la cual representar un sector único de la población como son sexo (hombre, mujer), edad (niño, joven, adulto, 3ra edad), y uno que otro con rasgos de color de piel o cabello diferente.

Por esto para la creación de personajes se optó por usar el programa FUSE el cual es una gran herramienta para la creación de personajes que cuenta con opciones de configuración, desde la modificación de los rasgos de los personajes hasta la integración de su ropa, además FUSE permite modificar las texturas del modelo, lo cual da como resultado personajes variados como se observa en la Fig. 3.6.

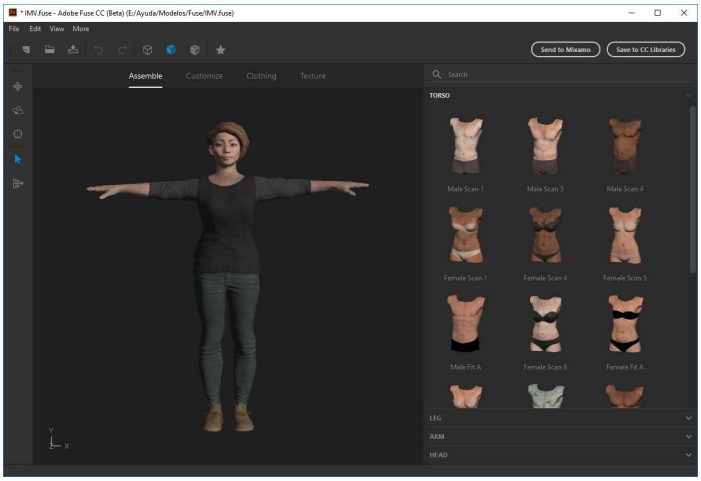


Fig. 4.5. FUSE

Una vez que los modelos son hechos, estos son exportados a mixamo el cual cuenta con más modelos además de una variedad de animaciones con un estándar establecido para el uso en Unity de los modelos, además de que se puede probar cómo se vería con una pequeña simulación del mismo modelo con diferentes animaciones como se ve en la Fig. 3.7.

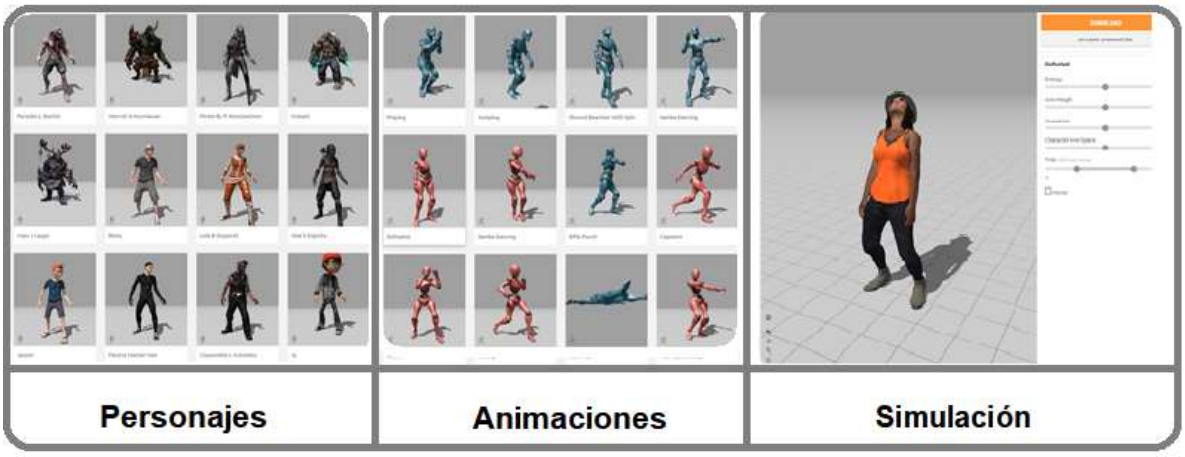


Fig. 4.6. Mixamo

4.4. Creación de la población.

Para dar una variedad de personajes a cada textura se le hace una modificación que depende de la cantidad de texturas destinadas a cada pieza de ropa al mismo tiempo que el modelo obtiene su vestimenta se le hace una deformación al mismo en sus características corporales; para que al crear una gran cantidad de personajes del mismo modelo cada personaje sea distinto, gracias a que la deformación es ligera, basada en cálculos probabilísticos. Cada personaje creado se ve muy realista.

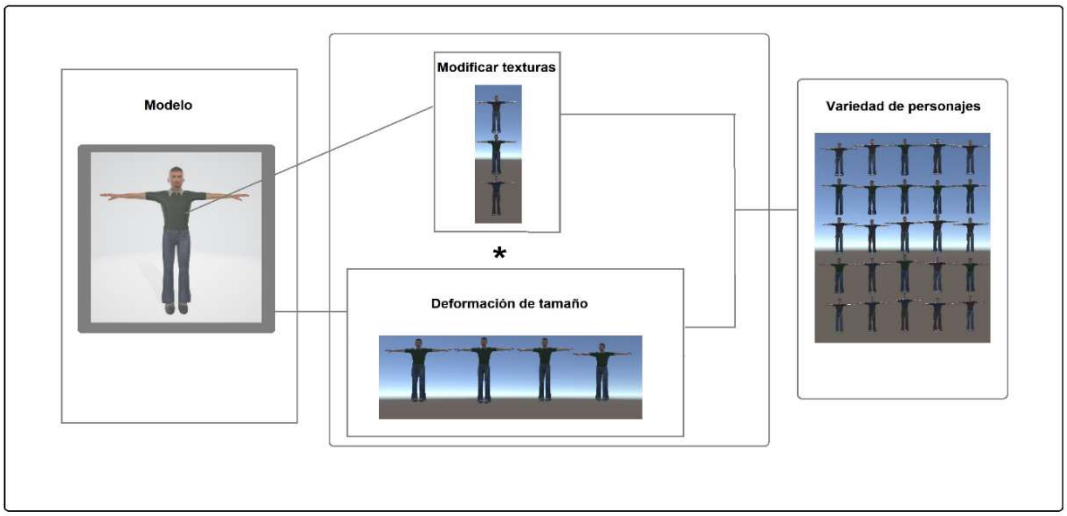


Fig. 4.7. Creación del personaje.

Una ves creado el sistema para la generacion de múltiples personajes de un mismo modelo, se aplicara a un conjunto de modelos, dando así una variedad de personajes, para poder crear una muchedumbre con variedad y con rasgo unicos en cada personaje.

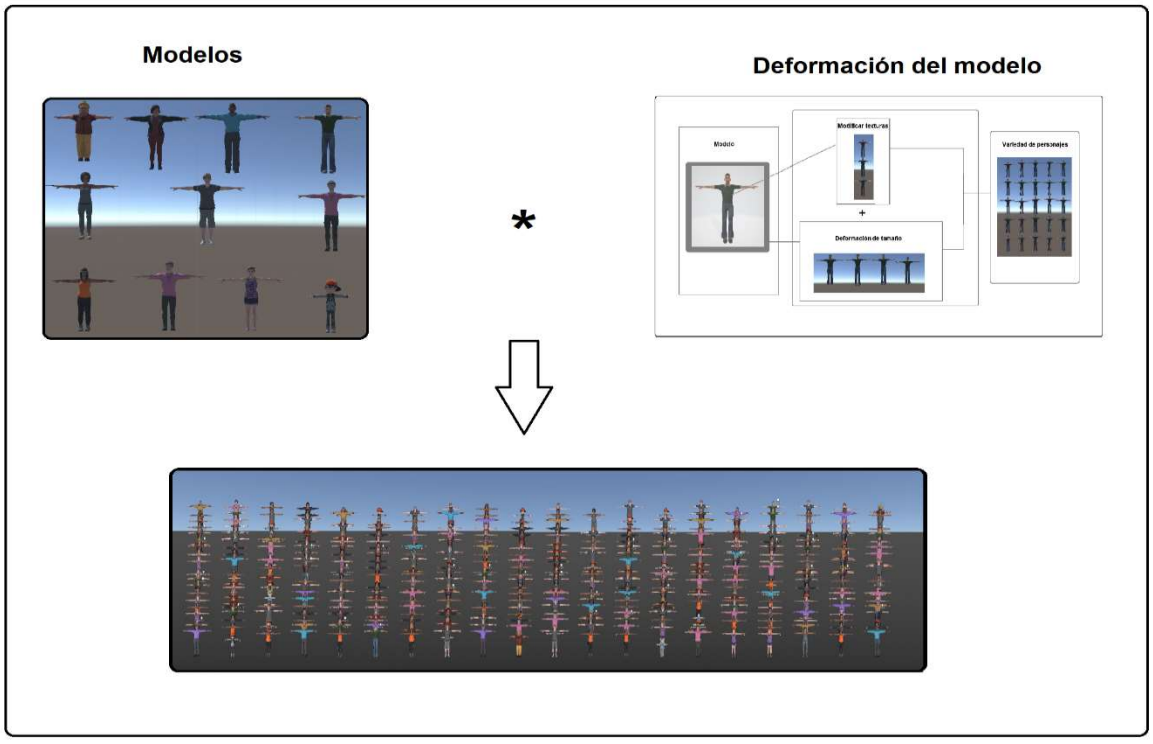


Fig. 4.8. Creación de la población.

4.5. Obtención de la ruta con PRM.

Debido a la importancia de la obtención del camino para nuestros personajes y que estos puedan ir de un punto inicial “su posición” a un destino. De ahi la importancia del algoritmo PRM y la obtención del camino en tiempo de ejecución aceptable a través de un roadmapbased, el cual será el mismo para todos los personajes, para no crear múltiples roadmaps.

Comenzaremos con la creación de lo que es el roadmap el cual se obtiene una vez creado el grid G donde se va a trabajar, y asi crear el Roadmap como se muestra en la Fig. 3.10.

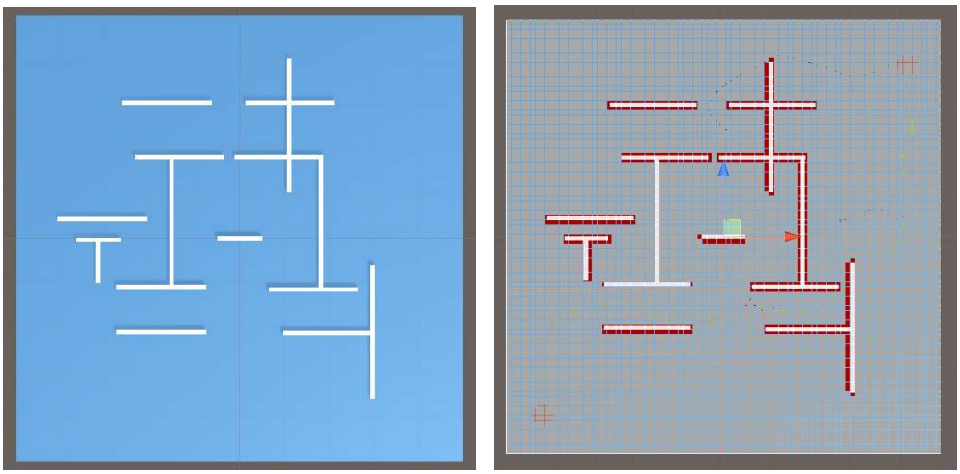


Fig. 4.9. Grid y escenario.

RM= (V, E) se crea a partir del conjunto de vértices V, los cuales a través del grid se comprueba que puedan ser usados y donde las entradas son: el número de vecinos K y la distancia D a considerar entre los vértices. Debido a la

naturaleza del problema, el RM será utilizado para diferentes agentes, por lo tanto, se comprobará si alguna arista (camino) no pasa por un lugar con obstáculos como se muestra en la Fig.3.11. y en el algoritmo 1.

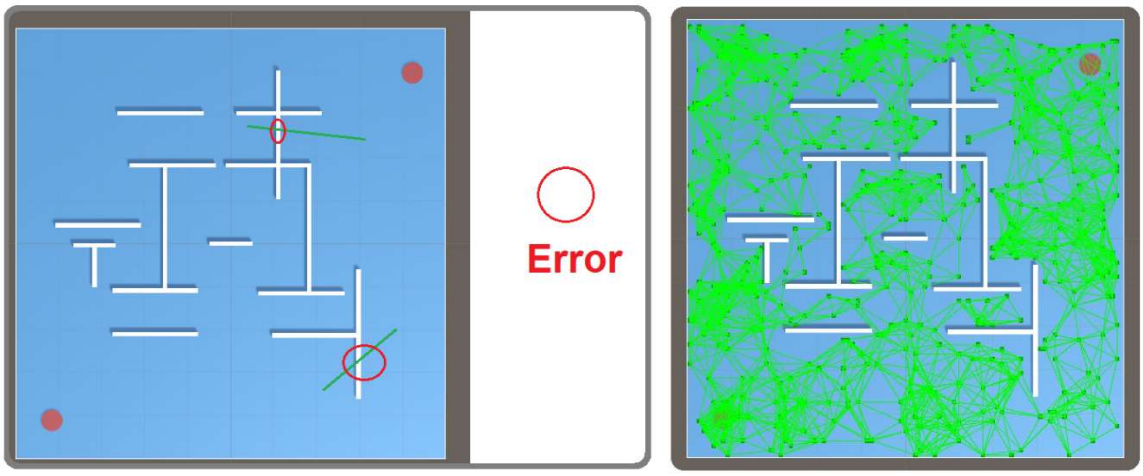


Fig. 4.10. Generación del RoadMap.

Algoritmo 1. RoadMap.

```
V=RM.generarVertices(N);
Para todo v1,v2 ∈ V hacer
  si v1!=v2  y comprobarDistancia(v1,v2,D) y
comprobarVecinos(v1,v2,K) entonces
    si RM.comprobarCamino(v1,v2) entonces
    RM.crearCamino(v1,v2)
    Fin si
  Fin si
Fin Para todo
```

Una vez terminado el RM se crea un camino a través del algoritmo A* el cual contiene dos conjuntos de nodos N los cuales son los nodos abiertos A y los nodos cerrados (Analizados) C, un nodo auxiliar Aux, el cual ira iterando para obtener el camino, pero inicialmente será el que contenga el vértice más cercano Vc del RM, y el Nodo Nv el cual será un nuevo nodo, un conjunto de vértices

R el cual contendrá el resultado. El algoritmo recibe de entradas: un punto inicial Pi , un punto final Pf ,y una distancia Dc que vea que tan cerca estará el nodo con el Pf y el RM hecho con anterioridad como se muestra en la Fig. 3.12. y el algoritmo 2.

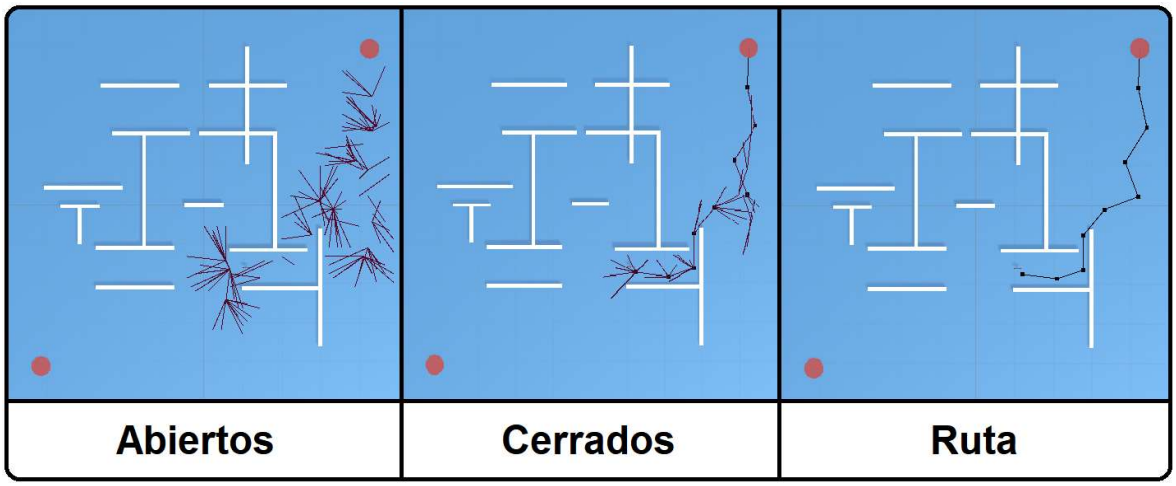


Fig. 4.11. Obtención del camino.

Algoritmo 2. A asterisco.

```
Vc=RM.Cercano(Pi)
Aux=Nuevo Nodo(Vc)
A.Agregar(Aux)
Hacer
  Para todo v ∈ Aux.octenerVecinos() hacer
    Si v ∉ A y v ∉ entonces
      Nv=nuevoNodo(v)
      Nv.AgregarPadre(Aux)
      Nv.CalcularCosto();
      A.Agregar(Nv)
    Terminar si
  Terminar para todo
  C.Agregar(Aux)
  A.Eliminar(Aux)
  Aux=A.OctenerMenorCosto()
Mientras Aux.Distancia(Pf) >Dc o A.Tamaño()!=0
Miestras Aux !=null
  R.agregar(Aux.ObtenerVertice ())
  Aux=Aux.Padre()
Fin Mientras
```

Después de obtener el camino a este de le agrgan curvas para que el avatar se mueva más naturalmente como se ve en la siguiente imagen.

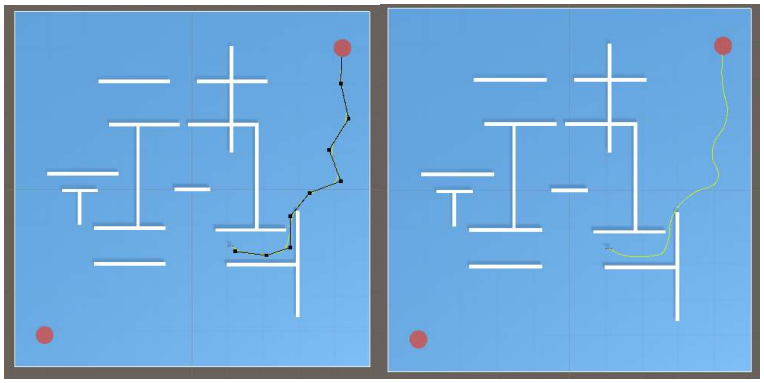


Fig. 4.12.Camino optimizado.

Como se puede ver en la Fig.3.14. entre más nodos mejor será el camino, pero a mayor cantidad, mayor cantidad de recursos y tiempo a utilizar pero entre menor cantidad de nodos corre el peligro de no crear un camino, es por eso que se optó por implementar puntos donde el algoritmo tiene que pasar a fuerza, sólo en caso necesario como se muestra en la Fig.3.15.

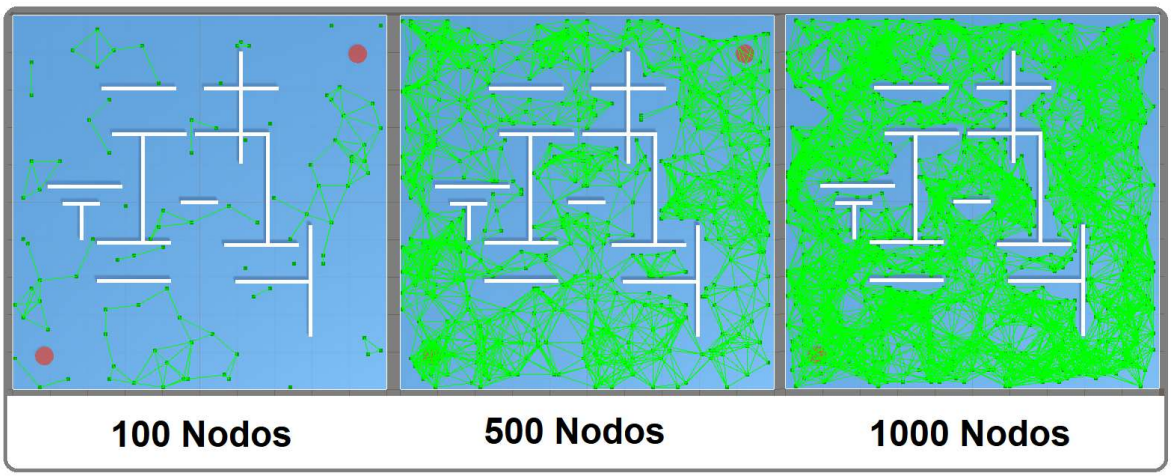


Fig. 4.13. Diferencia entre cantidad de nodos

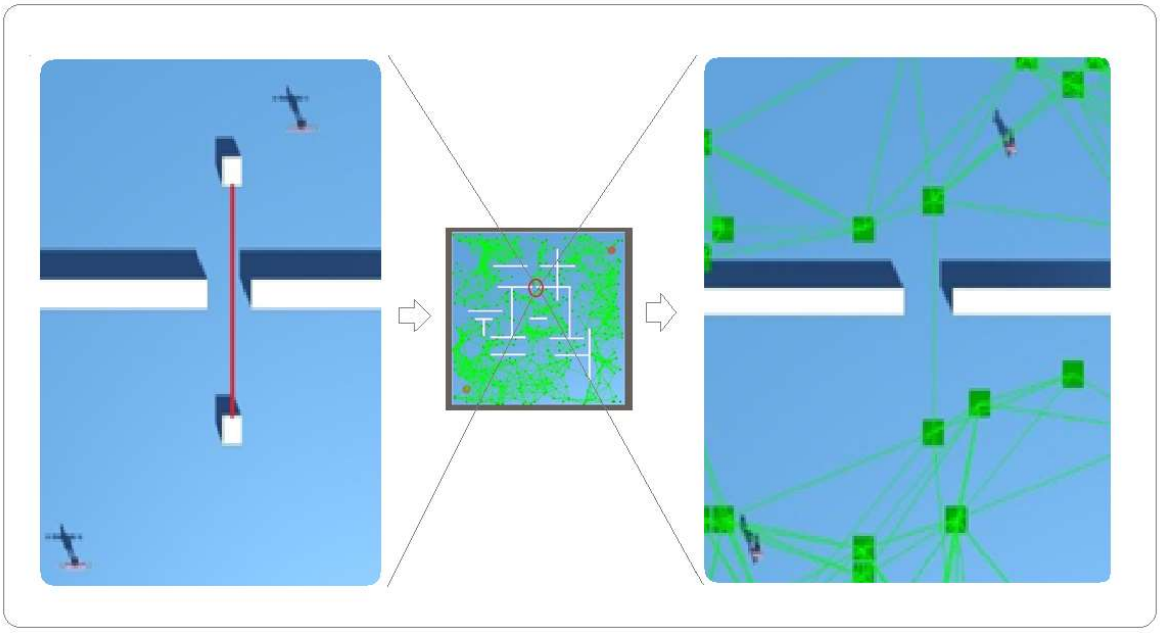


Fig. 4.14. Nodos Agregados.

Una vez establecido los parámetros se hace una simulación del resultado final de PRM, el cual es eficiente porque el mismo RoadMap es usado para todos los avatares y con el puedan usar A* sin estar recalculando el RoadMap. Esto ayuda demasiado ya que una vez obtenido los cálculos son más rápidos como se ve en

la Fig. 3.16. donde cada agente se mueve hacia su destino final, cada uno con su respetivo camino el cual fue obtenido a través del roadmap preestablecido en el momento de inicializar la aplicación.

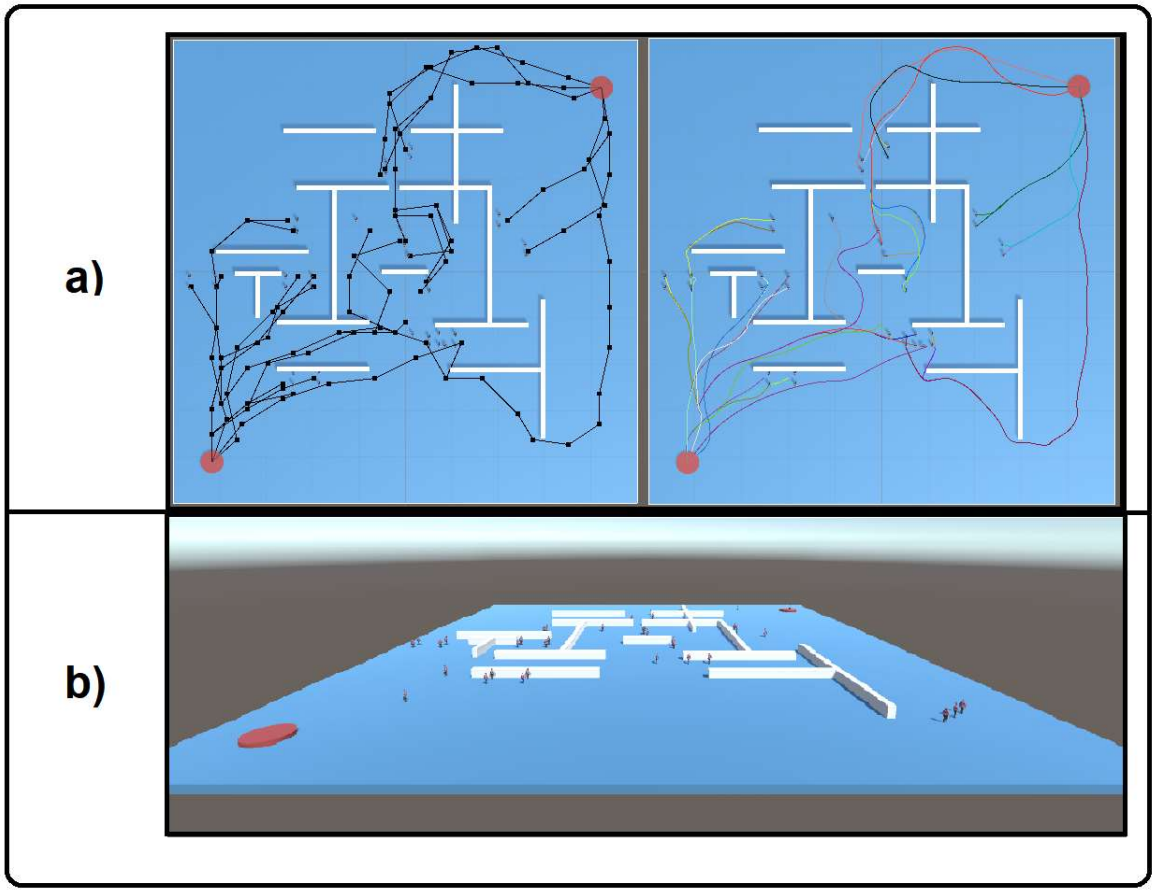


Fig. 4.15.Avatares con PRM.

También como ya se ha mencionado entre menor cantidad de nodos mayor posibilidad de que el algoritmo no encuentre su camino, a mayor número de nodos mejor será el camino creado como se ve en la tabla 1 pero también incrementa el tiempo de ejecución.

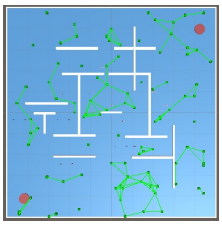
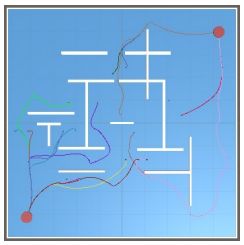
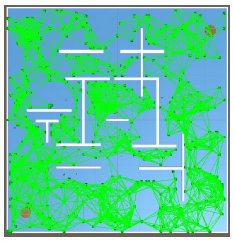
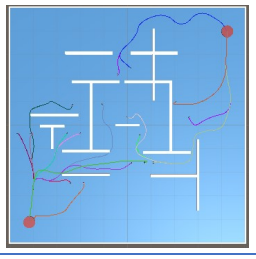
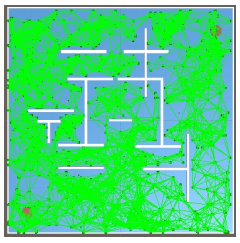
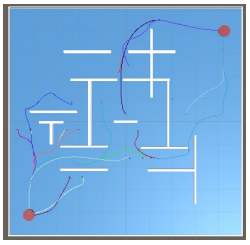
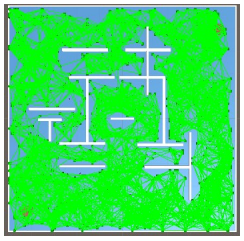
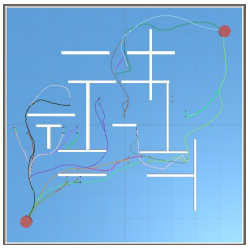
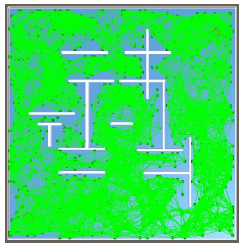
Numero de nodos.	**Tiempo.**	**Caminos encontrados.**	**Imagen del camino.**	**Imagen del roadmap.**
100	No hay ya que no se generó la ruta.	0/17	No hay imagen debido a que no se generó el camino por lo mismo y no inicio la animación por eso no hay un tiempo.	
500	1.9848976s	17/17		
700	3.1888626s	17/17		
1000	5.2900819 s	17/17		
1000	5.3709617 s	34/34		

Tabla 1.Comparacion de PRM con diferente cantidad de nodos.

Una vez establecidos los parámetros para una simulación optima, se agrega otro paso complejo, el cual es que el algoritmo funcione en distintos pisos, debido a que el modelo cuenta con 4 pisos los cuales tendrán un G y un RM cada uno, como se ve en la Fig. 3.17. En este caso los caminos Rs siempre serán los mismos para las escaleras, ya que su camino es definido desde un inicio, por lo tanto, los elementos Rs y Es son un conjunto de vértices del camino como se puede apreciar en la Fig.3.17. y en el algoritmo 3.

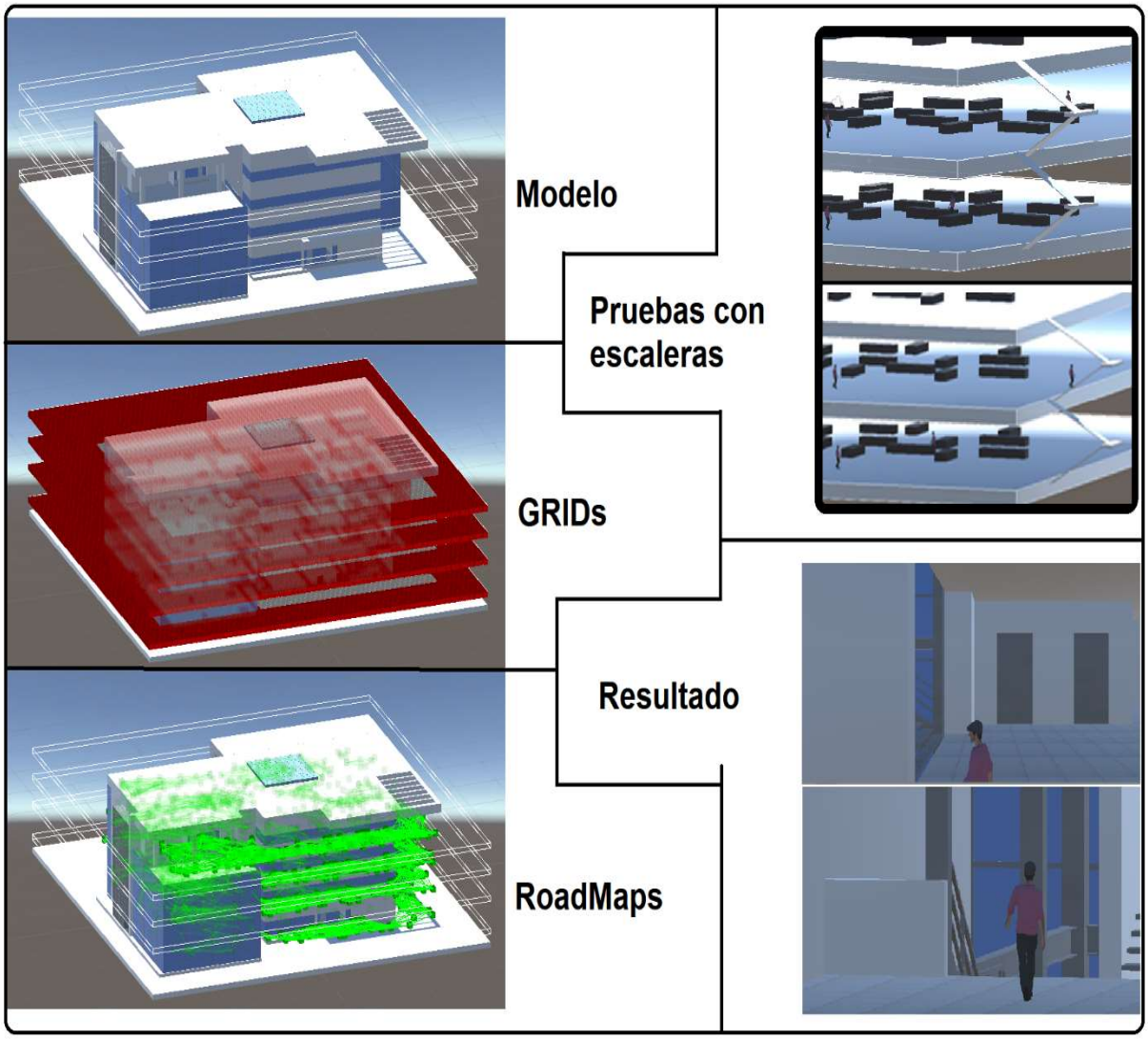


Fig. 4.16. Múltiples pisos

Algoritmo 3. Múltiples pisos.

```
Escaleras=true
Camino=Rs.OctenerCamino();
Camino.Animar();
Mientras Camino!=null entonces
  Si Camino.Finalizada() entonces
    Si Escaleras==false entonces
      Camino=Rs.SiguienteCamino();
Camino.Animar();
      Escalera=true;
    Terminar si
    Si Escaleras==true entonces
      Es.Animar();
Escalera=false;
    Terminar si
Terminar si
Terminar mientras
```

4.6. Obtención del camino con Unity.

Para implementar el algoritmo PRM se empieza con el barrido de las zonas (Bz) a través de los objetos estáticos (Os). ¿Qué significa estáticos? Son objetos que no se moverán una vez terminado el Bz se ve claramente donde es caminable antes de iniciar el Bz se tienen que definir el tamaño del avatar; para saber por dónde podrá pasar. Ya que las zonas de navegación (Zn) se crean de acuerdo al tamaño estándar del modelo, también se define el Angulo de inclinación para saber la pendiente donde se podrá navegar, los cuales son de mucha ayuda en escenarios con múltiples pisos.

Gracias a que esto se hace antes de la ejecución, se logra tener definidas las Zonas de navegación antes de comenzar la simulación como se ve en la figura siguiente.

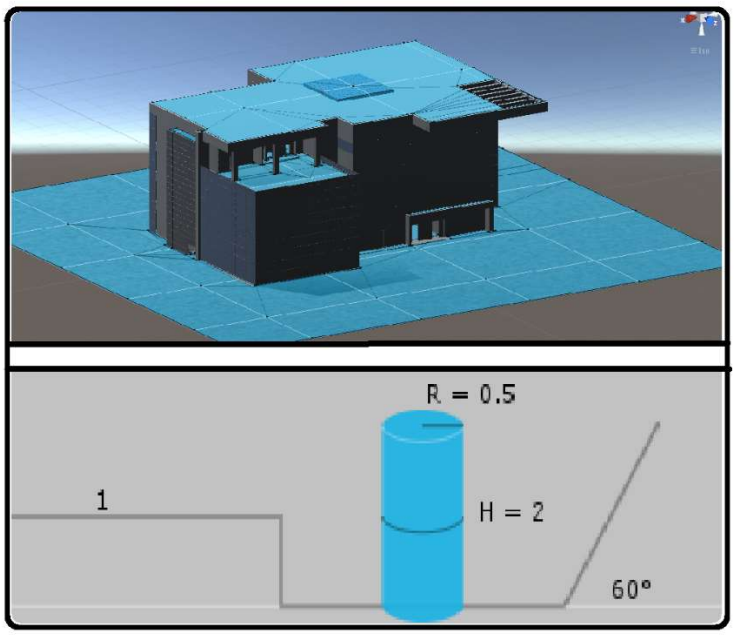


Fig. 4.17. Bake.

Una vez cargadas las Zn se puede obtener el camino (R) dando al agente la zona o punto (Pf) donde se desee llegar, en caso de no poder llegar el algoritmo devolverá el lugar más cercano posible al camino, pero debido a los obstáculos posiblemente sólo se ha bloqueado temporalmente, en este algoritmo se toma en cuenta los obstáculos y donde estan ubicados otros agentes.

En tiempo real lo anterior se agradece ya que si el obstáculo es activado el camino es modificado en tiempo real a la vez que si un agente se encuentra con otro agente simplemente lo evadirá o esperará a que se mueva dependiendo la cantidad de agentes obstaculizadores como se ve en la siguiente imagen.

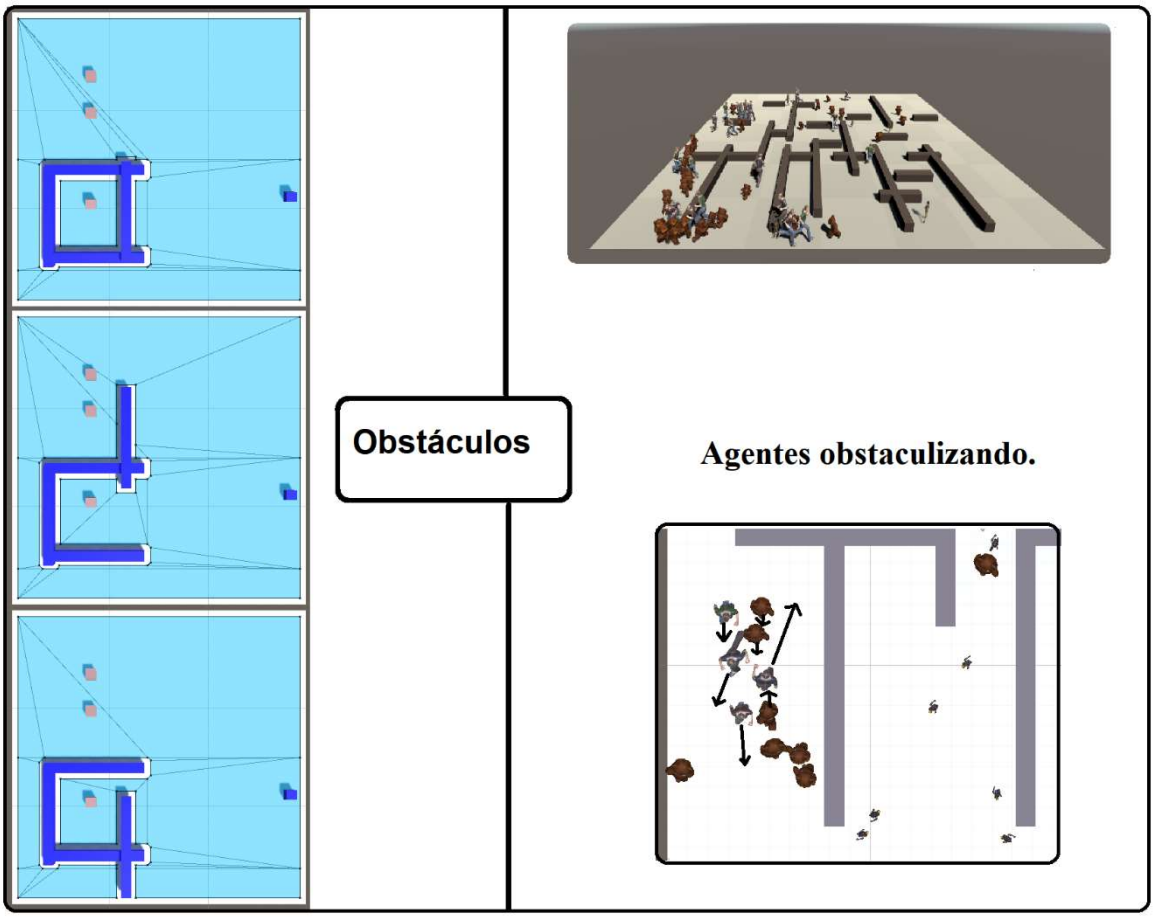


Fig. 4.18. Titulo.

4.7. Animación en Unity .

La animación en todos los casos anteriores ha estado presente de forma básica, pero para la simulación de evacuación se necesita la combinación de varias animaciones básicas las cuales se activarán al modificar ciertas variables que activarán uno u otro estado como se ve en la Fig 3.20, la animación se divide en 3 partes la animación como tal, el modelo y los scripts que activan la animación, una vez que se activa la animación el modelo sufre trasformaciones tales que logran el efeto de movimiento, lo cual da a lugar a animaciones complejas entre más animaciones básica se junten.

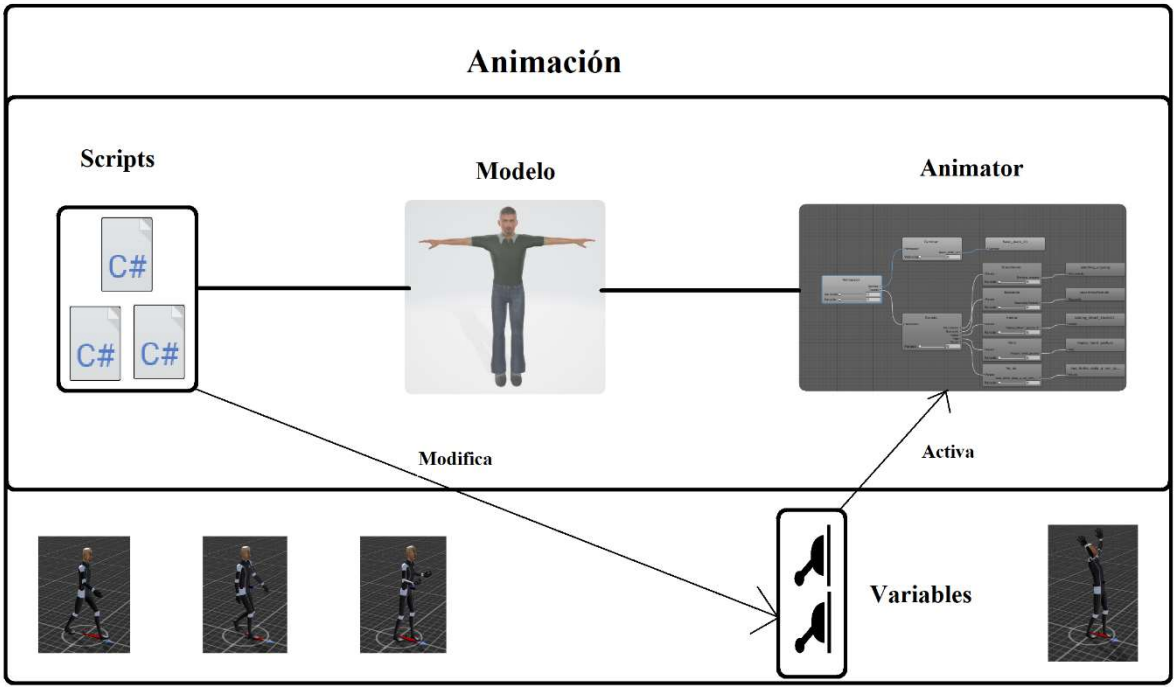


Fig. 4.19. Modelo de animación.

Una vez que se obtienen animaciones base; estas se animan de una u otra forma dependiendo la programación.

5. Propuesta del sistema desarrollado.

Debido a trabajos anteriores como son la facultad virtual de la FCC y FIQ se tomó la idea de modelar la biblioteca central y crear situaciones donde los

personajes evacuen dicha instalación y ver como se desarrollarían dichas situaciones en el ambiente real.

La propuesta unirá todo lo anteriormente visto desde la creación de la población la cual trabajará junto con el ambiente para que cada personaje obtenga cierto nivel de inteligencia para simular lo que es una evacuación en caso de incendio.

Se escogió la situación del incendio porque da cabida a innumerables situaciones, ya que se puede modificar por donde el fuego se propaga en nuestro escenario, de tal manera que se obligue al personaje a probar diferentes caminos, y con ello moverse de acuerdo al ambiente y comportamientos dados.

Los lugares que se incendian serán puntos específicos que logren bloquear el camino donde los personajes pueden caminar y estos puedan actualizarlo, de tal manera que tengan un cambio y modifique su estado.

Empezaremos por los modelos, el por qué se escogieron y como se adatan cada uno a las animaciones escogidas.

Como se plantea con anterioridad, mixamo es una gran aplicación que permite ver cómo se comportaría un personaje con una animación, para ello sólo se necesita el modelo y la animación.

Una vez escogida la población se comenzará con lo que es la integración de la población con Unity, ya que los modelos tendrán que tomar en cuenta su rol como un personaje dentro de nuestra simulación.

5.1. Condiciones de la simulación.

Como puede notar en la descripción anterior la simulación se llevó a cabo en un único escenario: la Biblioteca Central, con la finalidad de recrear una escena donde los personajes tengan que evacuar el edificio en ciertos puntos.

Como tal la idea es programar una aplicación donde escenario y personajes interactúen sin un agente externo, ejemplo nuestro usuario solo en un inicio dejara los parámetros base para que la simulación sea hecha como el usuario lo solicite, y con esto poder probar que pasaría en caso de incendio. Esto gracias al motor de videojuegos Unity que través del lenguaje C# o JS y un conjunto de herramientas que permiten crear aplicaciones completas desde un videojuego ,apps de VR y RA y simulaciones.

5.2. Mundo virtual o ambiente.

El ambiente es de gran importancia debido a que es donde nuestra simulación se llevara a cabo y donde se desenvolverán nuestros personajes. Como tal el ambiente es el modelo de la biblioteca central de la BUAP en CU puebla.

5.2.1.Características físicas del modelo.

- El edificio cuenta con 4 pisos cada uno con diferentes áreas.
- Escaleras de emergencia y escaleras normales ambas conectan con todos los pisos.
- Cada piso conecta con una salida de emergencia contada a la escalera de emergencia.
- Dos puertas de entrada/salida en la planta baja.
- Los cubos y oficinas se encuentran cerradas.
- El modelo es a escala del edificio real.
- En el último piso se ubica una puerta para la terraza.

5.2.2.Características en el software.

- Mapa de navegación resultante del mapeo.
- Generador de personajes: éste se activará una vez que se inicia la simulación.
- Controlador del tiempo y personajes: Una vez iniciada la simulación éste script iniciará un cronometro que controlara y verificara que tan cerca estan los personajes del punto de encuentro, si está muy cerca de otros personajes para así poder finalizar la simulación y parar el tiempo.
- Controlador de inicio: Este controlador modificará los datos iniciales antes de la simulación una vez que el usuario los introduzca en las interfaces de inicio.

5.2.3.Funciones.

- Mapearse así mismo para obtener el mapa de navegación.
- Generar personajes con los datos introducidos por el usuario.

5.3. Personaje.

Nuestra simulación tiene como objetivo salvaguardar del incendio a los personajes que tienen características únicas como son edad y sexo.

5.3.1.Características físicas del modelo.

- Texturas únicas del usuario que dará color de vestimenta y rasgos físicos.
- Rasgos únicos de personaje como altura y ancho del personaje.
- Esqueleto. este es indispensable para hacer la animación, ya que contiene los huesos que ayudan a generar una animación estándar en un modelo tipo humanoide.
- Mesh o *skin* (piel). La cual debe estar conectada o unida al esqueleto con el fin de definir qué partes del mesh del personaje se van a mover cuando una articulación (joint) determinada es animada.

5.3.2.Características en el software.

- El personaje cuenta con un controlador de animación con diferentes animaciones para humanoides, para así poder animar a todos los modelos que aunque diferentes, al tener el mismo estándar en los huesos y articulaciones se podrán animar con el mismo clip de animación.
- Controlador de rasgos. Este controlador es indispensable para que una vez que el personaje va a ser creado deforme el tamaño del modelo y cambie sus texturas propias.
- Controlador del camino este es indispensable, ya que pedirá al sistema el mapeo establecido con anterioridad, para crear el camino y poder moverse o actualizar el camino en tiempo real.

5.3.3.Funciones del personaje.

- Encontrar un camino de evacuación.
- Modificar el camino en caso necesario.
- Animarse de manera adecuada a la situación.
- Verificar estado.

5.4. Obstáculos.

Los obstáculos son los que modifican el camino de nuestros personajes, ya que se comunican con el controlador del mapeo, el cual automáticamente cambia el camino de nuestros personajes y lo cambia en tiempo real. Estos son de gran ayuda al momento de simular la salida del personaje en caso de que su camino principal se encuentre obstaculizado y así cambiar en tiempo real el camino.

5.4.1.Características físicas del modelo.

En este caso los modelos son dos, un modelo de fuego y otro de puerta de emergencia.

Fuego: Es un modelo con animación de partículas para dar más realismo al mismo y con un tamaño adecuado para cerrar el paso de los personajes, cada modelo de fuego está ubicado en escaleras y en puertas principales.

Puertas de emergencia: Las puertas de emergencia son puertas que aparecen y desaparecen dependiendo de los datos impuestos al comienzo.

Texturas únicas para cada tipo de obstáculo.

5.4.2.Características en el software.

Controlador de obstáculo independiente que activará o desactivará el estado del obstáculo.

Controlador único para la textura del obstáculo.

Para el obstáculo de tipo fuego se agregará un controlador especial para el manejo de las partículas.

5.4.3.Funciones.

Obstaculizar el paso de los personajes.

Desaparecer o aparecer dependiendo de los datos insertados por el usuario.

En caso del obstáculo de fuego es necesario que emita las partículas y simulen un incendio.

5.5. Interfaces.

Las interfaces son necesarias para solicitar y/o mostrar datos en tiempo de ejecución son, de gran ayuda para que el usuario pueda insertar datos iniciales y poder iniciar la simulación, además de notificar al usuario el tiempo y las acciones llevadas a cabo, como se verá a continuación.

Interfaz: Inicio

Descripción

Esta interfaz será usada para que el usuario inserte los datos iniciales para la simulación, como lo son la cantidad de personajes

Como se puede apreciar en la imagen la interfaz sólo cuenta con lo que es una entrada la
cual ayudara a obtener la cantidad de datos y un botón para la siguiente interfaz.

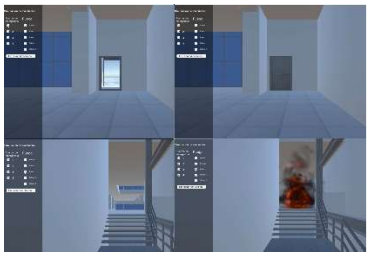
Interfaz: Obstáculos

Descripción

Esta interfaz será usada para activar y desactivar los obstáculos cada vez que se presiona automáticamente se dirigirá al obstáculo para visualizar el resultado.

Como se puede ver en la imagen, al modificar cada obstáculo cambiará la animación dependiendo del tipo

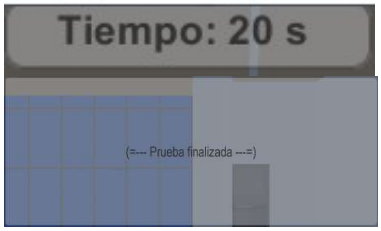
de obstáculo, pero la acción es la misma; desaparecer y aparecer y como se había mencionado antes, el obstáculo modifica el camino por el que pasarán los personajes, por lo cual son de gran relevancia.

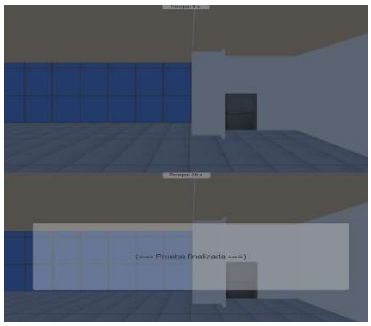


Interfaz: Simulación

Descripción

Esta interfaz será usada para la visualización del tiempo durante la simulación, pero cuando esta termina lanzará una alerta de que ha sido acabada la simulación.





Como se puede ver en la imagen, el hecho de tener el tiempo en la simulación dará a la persona a cargo de la simulación la capacidad de medir el tiempo y no sólo eso sino de darse cuenta cuando está terminada la simulación

5.6. Controles por teclado.

En esta parte vemos los comandos para que el usuario se mueva en la simulación a través del teclado y el ratón con la cámara y ver cómo se desarrolla nuestra ejecución en tiempo real sin afectar a los avatares. Cómo se puede apreciar en la siguiente tabla cada acción en el teclado afecta la visualización de la simulación.

Dispositivo.	**Acción.**	**Botones involucrados.**
Teclado	Rotar cámara	A) Rota izquierda.
		D) Rota derecha
	Ascender/Descender cámara	W) Ascender
		S) Descender
	Desplazar la cámara en la simulación	Esto se hace a través de las flechas de navegación del teclado. 
	Mover la cámara hacia los puntos de reunión	Esto se hace la siguiente forma. Z) Punto de reunión 1 X) Punto de reunión 2 C) Punto de reunión 3

	Mover a los diferentes pisos.	En este caso la cámara termina mirando a los puntos de reunión alternativos en caso de que las salidas sean bloqueadas V) Piso 1 B) Piso 2 N) Piso 3 M) Piso 4

Como se puede apreciar en la siguiente imagen cada botón que se presiona se dirigirá a los puntos de reunión lo cual es de gran ayuda para saber cómo van nuestros personajes respecto a los puntos de encuentro en la simulación.

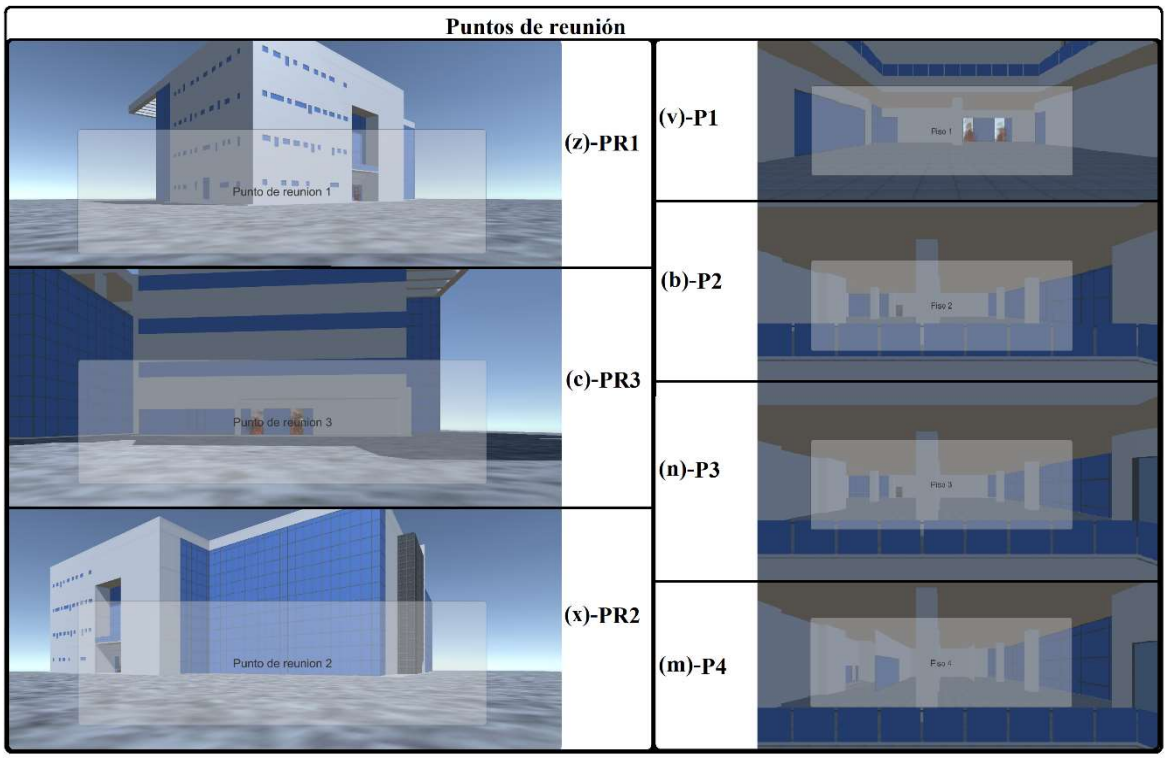


Fig. 5.1. Puntos de reunión.

5.7. Simulación final.

Como resultado final tenemos una app que es capaz de crear una simulación de incendio en tiempo real que permite a un usuario crear diferentes situaciones para dar una simulación realista que pueda brindar datos relevantes para el usuario y pueda comprender como se comportarían las personas en un incendio real.

La simulación puede modificar sus parámetros ya que Unity cuenta con un manejador de interfaces para que cada interfaz se enlace con cada controlador para empezar los eventos ya que al iniciar se solicitaran los datos necesarios para cada controlador y así empezar la simulación como sea requerido.

Una vez iniciada la simulación el poder visualizar cómo se comportan nuestros personajes en tiempo de ejecución a través del desplazamiento de la cámara, esto es de suma importancia ya que asi el usuario puede darse una idea del comportamiento de personas en entornos reales.

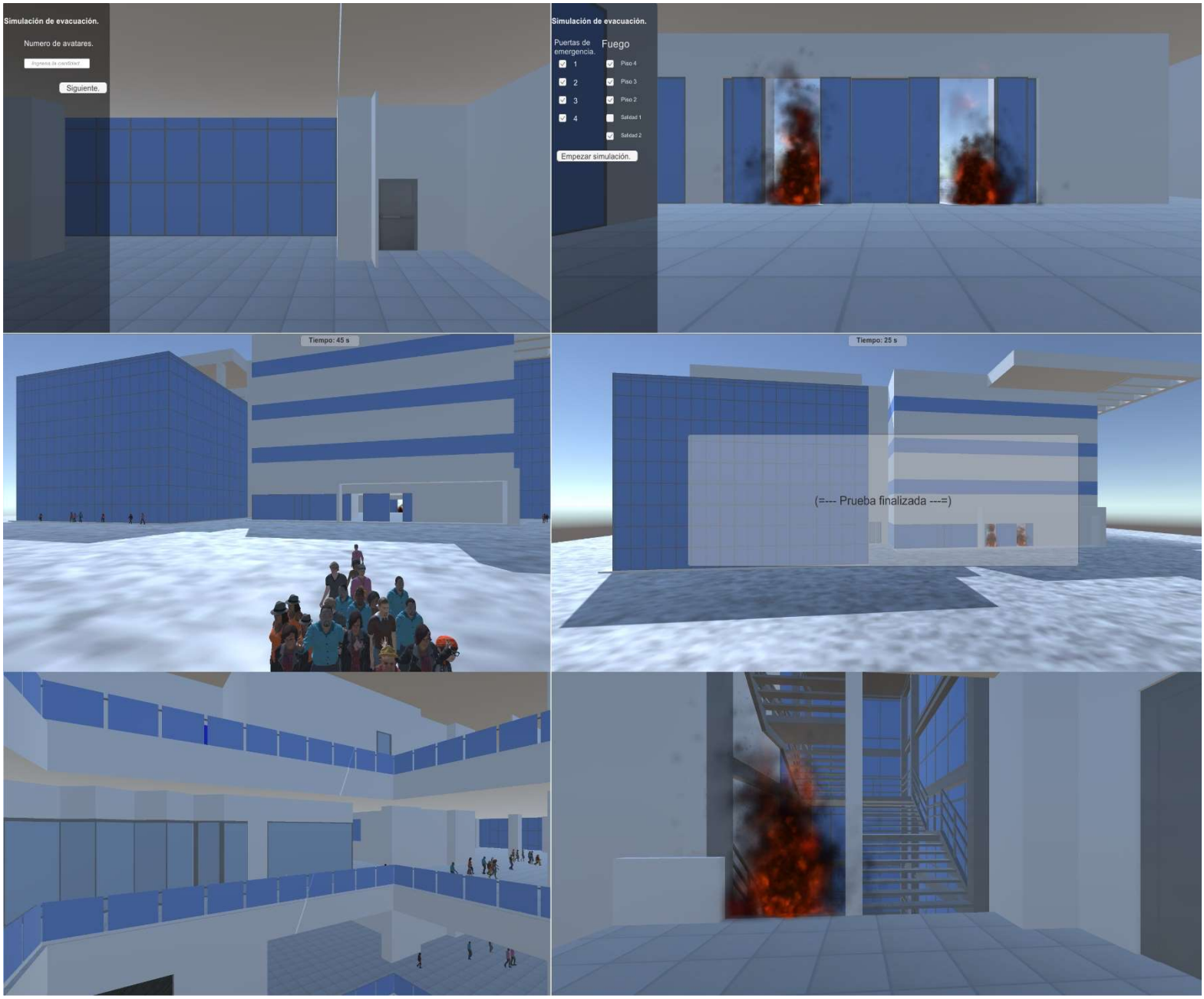


Fig. 5.2.Simulación de la aplicación.

6. Pruebas y Resultados obtenidos en la aplicación.

Una vez terminada la aplicación se iniciaron varias pruebas con diferentes parámetros para probar la adaptabilidad de la simulación, para poder dar una respuesta a situaciones reales y con ello prevenir futuros desastres.

Para estas situaciones se toma en cuenta la cantidad de personas en la simulación, la configuración del escenario es necesaria para crear diversos ambientes con diferentes resultados de tal manera que se refleje el comportamiento humano en la situación dada y probar posibles comportamientos.

Cada simulación se hace con 5 escenarios distintos cada uno tiene rasgos únicos que ayudaron a entender cómo funciona la simulación y cómo las personas podrían moverse en el entorno real a través de la simulación.

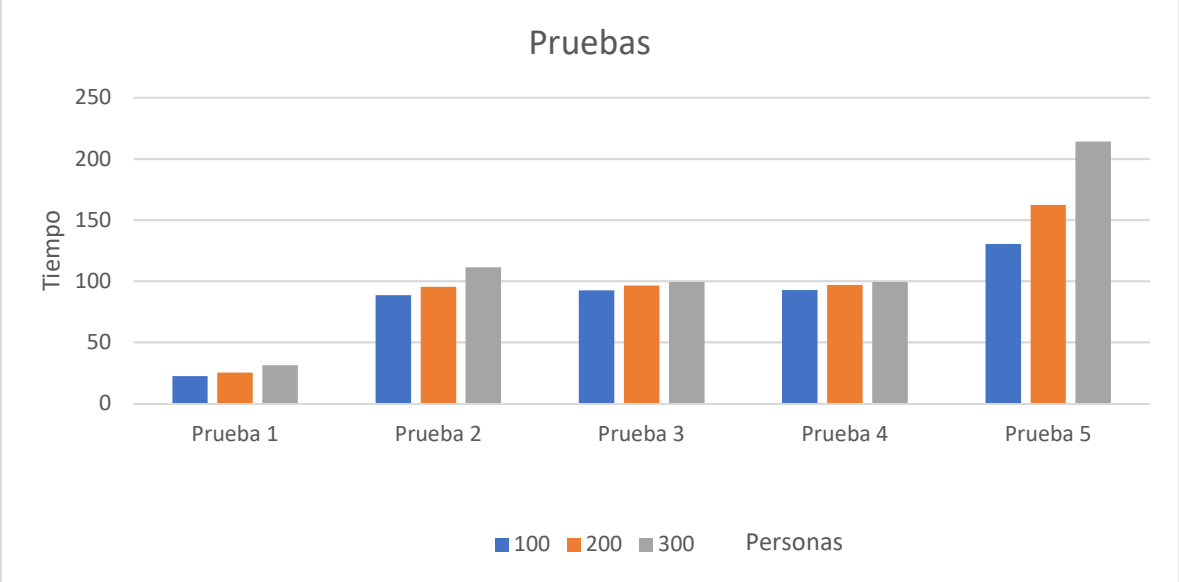
Los escenarios fueron construidos de tal manera que cada uno de ellos cuente con sus propias configuraciones y muestre diferentes situaciones posibles como se ve en la siguiente tabla.

Obstáculo	Escenario 1	Escenario 2	Escenario 3	Escenario 4	Escenario 5
Puerta de emergencia 1	Activado	Desactivado	Activado	Activado	Activado
Puerta de emergencia 2	Activado	Desactivado	Activado	Activado	Desactivado
Puerta de emergencia 3	Activado	Desactivado	Desactivado	Desactivado	Activado
Puerta de emergencia 4	Activado	Desactivado	Desactivado	Desactivado	Desactivado
Escalera Fuego piso 2	Activado	Desactivado	Activado	Activado	Desactivado
Escalera Fuego piso 3	Activado	Desactivado	Activado	Activado	Desactivado
Escalera Fuego piso 4	Activado	Desactivado	Activado	Activado	Desactivado
Salida 1	Activado	Desactivado	Activado	Desactivado	Activado
Salida 2	Activado	Desactivado	Desactivado	Activado	Activado

También como se ve en la siguiente imagen cada configuración es establecida en el input de nuestra aplicación.

Fig. 6.1 . Escenarios en Unity.

Cada una de estas pruebas se simularon con 100,200 y 300 personas como se ve en la siguiente gráfica y en las tablas posteriores.



Grafica 1. Pruebas.

Escenario	Escenario 1		
Cantidad de personas en la simulación	100	200	3
Tiempo promedio	22.3333333	25.333333	31.333333
Tiempo promedio	22	25	31

Escenario	Escenario 2		
Cantidad de personas en la simulación	100	200	300
Tiempo promedio	88.6666667	95.333333	111.33333
Tiempo promedio	88	95	111

Escenario	Escenario 3		
Cantidad de personas en la simulación	100	200	300
Tiempo promedio	92.3333333	96.333333	99.333333
Tiempo promedio	92	96	99

Escenario	4		
Cantidad de personas en la simulación	100	200	300
Tiempo promedio	92.6666667	97	99.333333
Tiempo promedio	92	97	99

Escenario	Escenario 2		
Cantidad de personas en la simulación	100	200	300
Tiempo promedio	130.333333	162.33333	214
Tiempo promedio	130	162	214

6.1. Resultados por escenario.

Cada escenario fue creado con el fin de probar una muestra de los posibles escenarios que pueden ser creados y así saber cuáles serán los resultados en cuestión de tiempo, cantidad de personas y que pasaría si hay casos donde las personas no pueden salir y cuales serían sus alternativas si los caminos fueran bloqueados.

En este punto se puede apreciar que sólo puede haber tres casos, donde salgan todos, donde no salga nadie y donde unos si salgan y unos no, cada uno con sus múltiples posibles configuraciones, las cuales nos mostraran diferentes resultados como los obstáculos en cada piso o cómo salen las personas en cada una de las pruebas realizadas.

6.2. Escenario 1.

Como se ve en las tablas anteriores, este el escenario con menor tiempo debido a que en este caso todas las personas sólo pueden quedarse dentro del edificio ya que todas las salidas son bloqueadas, lo cual hace que los personajes no se muevan como se verá en las siguientes imágenes.

Fig. 6.2. Escenario 1.

Como se ve en la imagen todas las personas se van al punto alterno que es dentro del edificio y no salen debido a que no hay un camino de salida, por lo mismo nadie llega a los puntos de salida propuestos, por lo que se termina la simulación, porque se quedan dentro del edificio y no se moverán de esa posición.

6.3. Escenario 2.

Como se ve en las tablas anteriores sorprendentemente este escenario no tiene el mayor tiempo y eso se debe que al estar todo abierto todos los personajes llegan a su punto de reunión por el mejor camino y no toman caminos alternos, si no que optan por el camino más rápido que como se verá en los siguientes escenarios.

Fig. 6.3. Escenario 2.

Como se ve todas las personas llegan a su punto de reunión y ninguno en su punto alterno.

6.4. Escenario 3 y 4.

En este caso ya empezamos con los híbridos los cuales no tienen todo cerrado y todo abierto. Como los casos anteriores en estos dos ejemplos se toma en cuenta que no todas las personas pueden llegar a los puntos de reunión y se quedan en los puntos alternativos.

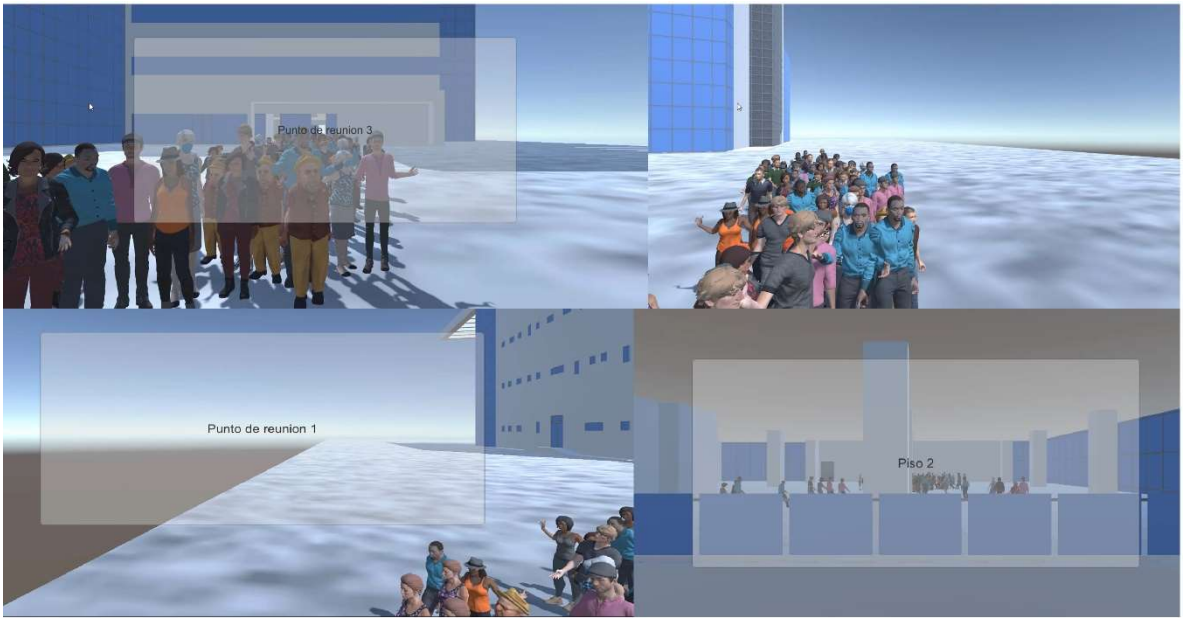


Fig. 6.4.Escenario 3 y 4.

Como se ve en la imagen casi todas las personas estan en los puntos de reunión excepto los del piso 2 debido a que se encuentran bloqueadas las salidas y como se ve en las tablas anteriores los tiempos de los escenarios 3 y 4 son muy parecidos debido a que las configuraciones son muy parecidas sólo cambia la puerta de salida, gracias a eso se pueden llegar a varias conclusiones sobre qué pasaría si alguna puerta en la parte baja no funcionara.

6.5. Escenario 5.

Este caso los resultados son un poco diferentes a los anteriores debido a que esta prueba fue diseñada para que todas las personas salieran, pero algunas sólo por caminos alternos y no las originales, pero todas llegaron a su punto de reunión.

Fig. 6.5. Escenario 5.

Como se ve en la imagen los de la planta 3 y la 1 se van a salir por las escaleras de emergencia del piso 2, esto da una idea que los de la planta 3 no van a la 4 a pesar de que pueden, debido que el camino más corto es por el piso de abajo.

Y cómo se puede ver todos llegan a sus puntos por lo cual se muestra información sobre qué pasaría si todos lograran salir por caminos alternos a los previstos.

Al finalizar la simulación se observa que todos los personajes salen, pero por caminos alternos, algunos personajes tardan más, por eso esta es la simulación con mayor tiempo en todos los casos.

7. Conclusiones.

Desarrollar una simulación requiere de la unión de varias disciplinas y áreas, las cuales se unen para la creación de un videojuego o representación de la vida. Para este trabajo, cada área hace su aportación en cada una de las fases de la creación de la simulación; desde la creación de un algoritmo, que le da inteligencia a nuestro personaje para poder escoger el mejor camino entre varios posibles, hasta la creación de los objetos y personajes en 3D. De acuerdo a la situación a representar, y el ambiente o escenario donde los personajes se desenvolverán, estos tienen ciertas características que dan como resultado una aplicación funcional y que cumple con los objetivos propuestos.

Estudiar e implementar PRM sirvió como apoyo para entender la importancia y la complejidad de utilizar un algoritmo de creación de caminos, y cómo éste puede mejorar si se toman en cuenta factores como el estilo de programación propio de Unity. Dicho algoritmo se puede mejorar comprendiendo su funcionamiento. Como se puede observar en los resultados, el ahorro de nodos no siempre es lo mejor para la aplicación, así como tampoco es beneficioso tener demasiados; el objetivo es llegar a un estado donde el *roadmap* pueda generar caminos sin demasiados nodos.

Las herramientas para el desarrollo de simulaciones son cada vez más sofisticadas y permiten mejorar el realismo, esto debido a sus facilidades para el manejo de modelos con un alto grado de detalle. Cada herramienta está enfocada en su propia área y brinda opciones únicas para la creación de contenido, ya sea para creación entornos *online* reales, como Second Life que

permite el manejo de avatares de forma fácil, o Unity que permite la creación de videojuegos sorprendentes a través de la integración de diversas disciplinas.

El hecho de poder ver una simulación y cómo se desarrolla, así como ver que los resultados que no siempre son los esperados, permite saber que si la gente se comportara como es debido tendría una mayor probabilidad de salvaguardar sus vidas, es algo increíble.

Finalmente, el haber trabajado con esta simulación me hizo comprender la complejidad que esto implica, ya que es una tarea complicada describir situaciones con gran fidelidad; en este caso un comportamiento humano en caso de incendio. El hecho de poder apreciar con detalle el comportamiento de personas virtuales en situaciones recreadas de forma controlada y con cierto nivel de realismo (que con el paso del tiempo irá mejorando), me lleva a pensar cómo será ese futuro cercano donde no se sabrá si lo que estamos viendo es una simulación o una situación real.

8. Trabajo a futuro.

Al ser este trabajo la integración de muchos elementos cada detalle se puede mejorar y agregar más contenido dando un mejor resultado tanto en vista como en configuración, debido a que cada vez las herramientas mejoran y por lo tanto las simulaciones se pueden mejorar en cualquiera de sus apartados, algunas de esas mejoras serian.

- La simulación requiere de más animaciones ya que en este caso solo se trabajó con las indispensables, pero para dar más realismo se requiere de más y mejores.
- Trabajar con nuestro propio algoritmo de obtención de un camino el cual ya lleva un gran avance, pero para la implementación requiere de comportamientos que lo aligeren y le permita la simulación con una gran cantidad de personajes a la vez.
- Mejorar la integración con diferentes accesorios como lo son los Oculus rift para una mejor inmersión del usuario final.
- Integrar diversos complementos al escenario como muebles, decoraciones, etc.
- Integrar diversas situaciones no sólo de incendio.
- Integrar los modelos a otros proyectos como juegos , otras simulaciones ,etc.

9. Bibliografía

[1] G. A. V. N. ,. C. G. F. Paola Mercado Lozano, «Mundos Virtuales, nuevas generaciones y nuevas formas de socialización,» Universidad de Guadalajara, Guadalajara, 2013.

[2] S. R. M. Daniel Thalmann, Crowd Simulation, London: Springer, 2007.

[3] S. I. L. L. Minutella Darío, «Animación de avatares en un Ambiente Virtual Inmersivo Interactivo,» Facultad de Informática Ciencias de la Comunicación y Técnicas Especiales , Morón, Buenos Aires, Argentina, 2016 .

[4] D. unity, «https://docs.unity3d.com/es/current/Manual/Glossary.html,» Unity, 2018.3.

[5] X. H. Velázquez, «Animación de grupos de entidades autónomas utilizando técnicas PRM,» Facultad de Ciencias de la Computación BUAP, Puebla, Pue. , 2016.

[6] B. P. C. J. L. e. L. F. S. F. M. M. D. R. J. C. S. I. Altube Alejandro, «Desarrollo de Planes de Evacuación, utilizando un Ambiente Virtual Inmersivo Interactivo,» Facultad de Informática Ciencias de la

Comunicación y Técnicas Especiales Universidad de Morón , Morón, Buenos Aires, Argentina, 2015.

[7] D. N. Ramírez, «MOVE – BUAP # Una herramienta multipropósito para la animación de personajes virtuales,» Facultad de Ciencias de la Computación BUAP, Puebla, Pue., 2008 .

[8] S. O. RODRIGUEZ, ROADMAP-BASED TECHNIQUES FOR MODELING GROUP BEHAVIORS IN MULTI-AGENT SYSTEMS, Texas: Texas A&M University , 2012.

[9] A. S. L. Juan Carlos Conde Ramírez, «Una arquitectura para el modelado cognitivo para apoyar la adaptación en tiempo real y las respuestas motivacionales en el videojuego,» MOVIS BUAP, Puebla, Pue, 2013.

[10] J. C. C. Ramírez, «Modelado Cognitivo en Videojuegos,» Facultad de Ciencias de la Computación BUAP, Puebla, Pue, 2013.

[11] G. I. M. PINTLE, «Propuesta de un motor de comportamientos grupales para el desarrollo de video juegos,» FCC BUAP, Puebla, Pue., 2016.