Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

"MOVE – BUAP # Una herramienta multipropósito para la animación de personajes virtuales"

Tesis que presenta

David Núñez Ramírez

Para obtener el titulo de

Ingeniero en Ciencias de la Computación

Director de tesis

Dr. Abraham Sánchez López

Puebla, Pue. ***Otoño 2008***

A mis padres, por su confianza.

Agradecimientos

Quiero agradecer…

… a mis padres, que siempre han estado junto a mí y me han guiado con sus sabios consejos.

… a mi familia y las personas que me quieren, por su apoyo y comprensión.

… a mi asesor, por su paciencia y verdadera amistad.

… a mis amigos, porque siempre están ahí, para hacer este mundo más divertido.

… a la Facultad de Ciencias de la Computación y el VIEP, por la enseñanza, el soporte e infraestructura que hicieron posible la culminación de este proyecto.

Resumen

La animación por computadora y la creación de videojuegos representa una actividad comercial muy exitosa en el ambiente computacional. En cualquier hogar es usual encontrar consolas de videojuegos o DVDs de películas con efectos especiales asombrosos.
El mayor desafío en la animación de personajes virtuales es la naturalidad en los movimientos de las estructuras. En general, la animación desarrollada en la actualidad se basa en tres métodos: captura de movimientos, especificación de escenas clave y comandos de procedimiento.

Cada método satisface diferentes objetivos en la animación, pero en general el conjunto de las tres genera las escenas con mayor naturalidad y mejor desempeño. Los modelos animados han sido desarrollados desde los noventa, pero la simulación de la dinámica humana no solucionó el problema de manera satisfactoria. Razón por la que los científicos se han interesado en la investigación de la combinación de movimientos existentes.

Para la mayoría de las aplicaciones, la captura de movimientos debe ser modificada para crear una variedad de animaciones específicas que puedan tomar en cuenta al ambiente sintético.

El objetivo de este proyecto es el desarrollo de una herramienta de animación tridimensional MOVE – BUAP # en el lenguaje C# utilizando OpenGL como motor gráfico y ODE como gestor de física. Esta aplicación cuenta con un controlador con capacidad para combinar y extrapolar con naturalidad, movimientos capturados del personaje virtual. Dicha solución muestra además la animación de un personaje que navega desde una posición inicial a una final dentro de un escenario tridimensional mediante el empleo de comandos de alto nivel.

Índice general

Resumen ... 1
Índice general ... 2
Índice de figuras ... 4
Capitulo 1. Consiguiendo la locomoción ... 5
1.1 Introducción ... 5
1.2 Modelización y control del actor ... 6
1.2.1 Modelo del personaje virtual ... 6
1.2.2 Cinemática directa ... 7
1.2.3 Cinemática Inversa ... 8
1.2.4 Las posiciones claves ... 9
1.2.5 El control de los métodos cinemáticas ... 10
1.2.6 Conclusión ... 11
1.3 Animación e imitación ... 11
1.3.1 Las técnicas de captura de movimiento ... 11
1.3.2 Edición de las capturas de movimiento ... 13
1.3.2.1 Adaptación morfológica ... 13
1.3.2.2 Restricciones espacio – temporales ... 14
1.3.2.3 Deformación cinemática ... 14
1.3.2.4 Mezcla de captura de movimientos ... 15
1.3.3 Control de las técnicas de edición de capturas ... 16
1.3.4 Conclusión ... 16
1.4 Animación y simulación ... 16
1.4.1 Principio ... 17
1.4.2 Control dinámico del personaje virtual ... 18
1.5. Planificación de la locomoción ... 18
1.5.1 Planificación y robótica ... 18
1.5.2. Planificación y animación de personajes ... 20
1.5.3. Movimientos reactivos ... 22
1.6. Nuestro enfoque ... 23
1.6.1 Entradas y salidas ... 24
1.6.2 Módulos ... 24
1.6.3 Organización del documento ... 25
Capitulo 2. Modelización del personaje y manejo de capturas de movimiento. .. 26
2.1 Mecánica del actor virtual ... 26
2.2 Aspecto visual del personaje virtual ... 27
2.3 Miembros activos y reactivos. ... 28
2.4 Las capturas de movimiento ... 29
2.4.1 Archivos de esqueleto (.asf) ... 29
2.4.2 Archivos de captura de movimiento (.amc) ... 30
2.5 El modelo matemático de locomoción del personaje ... 30
2.6 El controlador cinemático del personaje. ... 32
Capitulo 3. Planificación de la trayectoria ... 35
3.1 Introducción ... 35

3.2 Planificación basada en árboles aleatorios de exploración rápida 35
3.2.1 El planificador RRT Connect 36
3.3 Alisado del camino planificado 37
3.3.1 Subdivisión del camino planificado 38
3.3.2 Optimización del camino planificado 38
3.3.3 Optimización bi-direccional del camino planificado 38
3.4 Aproximación del camino global con curvas de Bézier 39
Capítulo 4. Resultados 42
4.1 El entorno de trabajo 42
4.1.1 Compilado principal 43
4.1.2 Compilado herramientas 49
4.1.3 Compilado importador3DS 49
4.1.4 Compilado importadormovimiento 50
4.1.5 Compilado física 50
4.1.6 Compilado Hmat 50
4.1.7 Compilado Formas 50
4.2 Pruebas y resultados 51
4.2.1 Prueba 1. El estacionamiento 51
4.2.2 Prueba 2. El interior de una casa 52
4.2.3 Prueba 3. El corral con ovejas 54
Capítulo 5. Conclusiones y trabajos futuros 56
5.1 Conclusiones 56
5.2 Trabajos futuros 56
Bibliografía 58

Anexo. El motor de física ODE 64
A.1 Introducción 64
A.1.1 Características 64
A.1.2 Particularidades 64
A.2 Instancias básicas de ODE 65
A.2.1 Concepto de mundo (world) 65
A.2.2 Concepto de cuerpos (body) y masa (mass) 66
A.2.3 Concepto de geometría (geom) 67
A.2.4 Concepto de espacio (space) 67
A.2.5 Concepto de unión (joint) 68
A.2.6 Concepto de unión de contacto (contact joints) y colisiones 68
A.2.7 Concepto de fuerza (force) 69
A.3 ODE.NET 69
A.3.1 Descargar ODE.net 70
A.3.2 Instalar ODE.net 70
A.3.3 Ejemplo de ODE.net 71

Índice de figuras

Figura 1.1: Ejemplo de modelo cinemático 6
Figura 1.2: Arquitectura del sistema 24
Figura 2.1: Estructura cinemática del personaje 27
Figura 2.2: Vestimenta del personaje virtual. 27
Figura 2.3: Forma semi-envolvente y grados activos y reactivos del personaje 28
Figura 2.4: Esqueleto jerárquico (archivo .asf) 29
Figura 2.5: Movimiento contenido en un archivo (.amc) 30
Figura 2.6: Disco que representa al personaje en el controlador de locomoción 31
Figura 2. 7: Comportamiento de Warping 34
Figura 3.1: Comportamiento del planificador RRT Connect 37
Figura 3.2: El camino planificado (izquierda) y el alisado (derecha). 39
Figura 3.3: Diferentes curvas de Bézier y su composición. 40
Figura 3.4: Camino generado con curvas de Bézier 41
Figura 4. 1: Compilados del sistema MOVE - BUAP # 43
Figura 4. 2: Interfaz gráfica de usuario de MOVE - BUAP # 44
Figura 4. 3: Menú de control de MOVE - BUAP # 44
Figura 4. 4: Pestaña “General” de MOVE - BUAP # 45
Figura 4. 5: Pestaña "Escenario" de MOVE - BUAP # 45
Figura 4. 6: Pestaña "Iluminación" de MOVE - BUAP # 46
Figura 4. 7: Pestaña "Personajes" de MOVE - BUAP # 46
Figura 4. 8: Pestaña "Editor de personajes" de MOVE - BUAP # 47
Figura 4. 9: Pestaña "Editor de posturas" de MOVE - BUAP # 47
Figura 4. 10: Pestaña "Editor de movimientos" de MOVE - BUAP # 48
Figura 4. 11: Pestaña "Problema" de MOVE - BUAP # 48
Figura 4. 12: Pestaña "Animación" de MOVE - BUAP # 49
Figura 4. 13: Prueba 1. Posición inicial y final 51
Figura 4. 14: Prueba 1. Trayectoria planificada 52
Figura 4. 15: Prueba 1. Movimiento reactivo efectuado 52
Figura 4. 16: Prueba 1. Animación generada 52
Figura 4. 17: Prueba 2. Posición inicial y final 53
Figura 4. 18: Prueba 2. Trayectoria planificada 53
Figura 4. 19: Prueba 2. Movimiento reactivo efectuado 53
Figura 4. 20: Prueba 2. Animación generada 54
Figura 4. 21: Prueba 3. Posición inicial y final 54
Figura 4. 22: Prueba 3. Trayectoria planificada 55
Figura 4. 23: Prueba 3. Movimiento reactivo efectuado 55
Figura 4. 24: Prueba 3. Animación generada 55

Figura A. 1: Comprimidos de ODE para descargar 70
Figura A. 2: Sección de referencias del proyecto 71
Figura A. 3: Ventana para agregar referencias de proyecto 71
Figura A. 4: Captura 1 del ejemplo 74
Figura A. 5: Captura 2 del ejemplo 75

Capitulo 1. Consiguiendo la locomoción

1.1 Introducción

Actualmente la cinematografía y el desarrollo de videojuegos representan una parte muy lucrativa dentro de la industria del software. En particular la participación de personajes virtuales dentro de estos entornos se ha incrementado gracias a la investigación realizada en el campo de la cinemática humana. Un actor ideal es aquel que realiza las acciones de manera independiente a sus decisiones. Para tal objetivo, éste debe considerar las restricciones que representa su entorno, así como las propias. El objetivo primordial de esta investigación es que el personaje virtual sintetice una animación de manera autónoma que represente el movimiento de un ser humano de manera realista mediante el uso de los comandos de alto nivel proporcionados por el usuario. El objetivo es dirigir al personaje virtual como un director lo hace con un actor real en la filmación de una película.

La locomoción, del latín locus (lugar) y motio (movimiento), se define como la capacidad de los seres humanos de cambiar de lugar. Específicamente la locomoción del personaje virtual representa una parte importante de la resolución del problema planteado. Por tanto el objetivo de la planificación es encontrar las trayectorias que lleven de un punto a otro al personaje virtual y que éste elabore la animación del desplazamiento de manera realista y autónoma. Para llevar a cabo tal tarea se debe considerar un conjunto de restricciones que modificaran la trayectoria como puede ser la presencia de obstáculos.

El reto de la locomoción autónoma conlleva al problema de la decisión del camino a seguir, pero además del realismo del movimiento de desplazamiento a lo largo de este camino. Este enfoque de dos niveles es constante en la literatura, y las dos capas comportamentales aparecen claramente en la concepción de las diferentes soluciones existentes. Por otro lado, estas soluciones presentan numerosas dificultades técnicas como el control de las articulaciones, de los miembros, del cuerpo en su integralidad, pero además la gestión de las interacciones con el ambiente o de otros personajes en la escena. El resultado final de estas técnicas es una animación, una serie de imágenes, que ejecutadas a una velocidad satisfactoria, reproducen un efecto visual de movimiento. La dificultad de este ejercicio consiste en generar en los espectadores un sentimiento de convencimiento del realismo de la animación.

El personaje es representado por un modelo que contiene dos secciones principales: el aspecto geométrico y el comportamiento físico. La posición del actor virtual esta condicionada por las entradas que se proporcionen al modelo mencionado. El usuario puede manipular directamente las entradas para obtener cinemática directa del personaje o elaborar técnicas que proporcionen capacidad de cinemática inversa al actor virtual. El establecimiento de leyes y restricciones por parte del animador puede ser usado para generar un controlador que automatice la generación del proceso de animación.

Un realismo convincente se obtiene por medio de las técnicas de captura de movimientos. Tal cual estas técnicas ofrecen poca flexibilidad, pero las capturas realizadas pueden ser editadas y sus características adaptadas a un contexto o a una situación precisa. Existen muchos métodos. Todos realizan un compromiso entre la importancia de las modificaciones posibles y la preservación del realismo de las capturas.

1.2 Modelización y control del actor

1.2.1 Modelo del personaje virtual

El personaje virtual representa un cuerpo humano. Éste refleja su apariencia geométrica y su comportamiento mecánico. Un modelo óptimo representaría los huesos, músculos, articulaciones, tejidos, cabello, piel, nervios y otros elementos propios del cuerpo humano. Sin embargo lo anterior encierra una complejidad difícil de controlar. Para la tarea de locomoción se toma en cuenta únicamente los miembros más importantes del esqueleto. Los otros elementos se moverán en conjunto con la estructura del esqueleto.

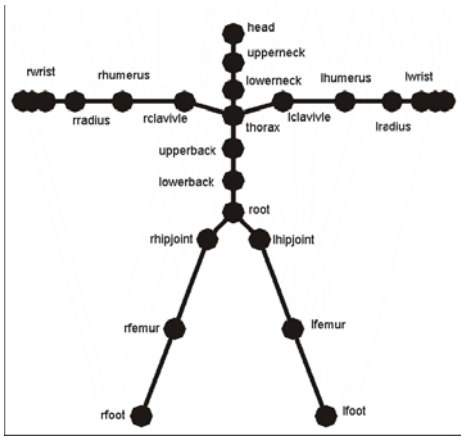

Figura 1.1: Ejemplo de modelo cinemático

El esqueleto es representado por una cadena cinemática abierta, es decir un conjunto de cuerpos rígidos articulados entre ellos, con un nodo raíz y un extremo en cada extremidad. Se establece también como restricción los limites angulares que pueden alcanzar el movimiento de cada una de las articulaciones. La figura 1.1 muestra un ejemplo simplificado de esqueleto donde se identifica cada una de las articulaciones controladas. La raíz del personaje se representa como el nodo ubicado en las caderas. Éste especifica la posición global del personaje en el entorno virtual. Los otros miembros están orientados relativamente a su predecesor en la cadena cinemática que implica cinco ramas abiertas: 2 hasta los pies, 2 hasta las manos y una hasta la cabeza. En este tipo de modelo se establece un conjunto de suposiciones que incrementan la facilidad en el control de la animación:

La cantidad de huesos en el esqueleto virtual es menor al del esqueleto real (hasta 32 para el modelo en comparación con los 200 de los humanos)

Las articulaciones están modeladas como articulaciones esféricas o de pivotes, por tanto las colisiones con el medio se comportan de manera aproximada.

La raíz de la cadena cinemática se encuentra ubicada en el centro de la cuenca, hecho que modifica el comportamiento real de las articulaciones humanas donde las uniones se encuentran en los extremos de ésta.

Los límites angulares de los miembros se consideran independientes, hecho que no se lleva a cabo en el cuerpo humano. Un ejemplo de este fenómeno se da cuando el pie no tiene el mismo alcance con la pierna tensa que con la pierna doblada.

El modelo satisface un buen número de requerimientos de la animación realista por computadora. Además para alcanzar un realismo apreciable se debe también considerar el alto detalle de la geometría del personaje.
La apariencia visual del personaje esta dada por un conjunto de formas geométricas que adquieren la posición que les corresponde en el cuerpo humano y son ubicados en el futuro por la salida del modelo. Pueden representar los miembros del cuerpo de manera estática o dinámica, dependiendo del realismo que se quiera proporcionar a la animación. Las formas geométricas pueden incluir músculos y tejido así como ropa y accesorios. La investigación en el ámbito de simulación de tejidos tiene dos corrientes en la actualidad: los modelos de tipo físico [GMTT89, CZ92, NT98, AT01] y los de tipo geométrico [TSC96, SPCM97, Wil97]. El diseño y uso de esta metodología de recubrimiento mejora de manera considerable el realismo de la animación.

Badler et al, en [BPW92] proponen un conjunto de objetivos que debe satisfacer un modelo de personaje. A medida que el actor virtual incrementa su complejidad, el realismo en su animación se ve beneficiado. El control y manipulación del personaje va ligado de manera directa con la complejidad de las restricciones y automatizaciones del modelo. A partir de esta idea se han creado modelos cuya complejidad varía en función a la proximidad o alejamiento del punto de vista y el personaje. De manera que el nivel de detalle aumenta entre más cerca esté el personaje a la cámara [Nou98, ABT00].

1.2.2 Cinemática directa

En gráficos 3D, la cinemática directa se define como una técnica para calcular la posición de las partes de una estructura articulada a partir de sus componentes fijos y las transmisiones inducidas por las articulaciones. En nuestro caso consiste en definir la orientación y posición de cada articulación del esqueleto para posicionar al personaje virtual. Para llevar a cabo la manipulación en el posicionamiento de las partes del actor, es necesario tomar en cuenta los siguientes puntos:

- La definición de sistemas locales ligados a cada miembro del personaje, y la utilización de matrices de transformación entre estos sistemas para situarlos unos en relación de otros. Las matrices formadas de esta manera permiten

visualizar inmediatamente los sistemas locales, pero sus composiciones son costosas y no son adaptadas a la interpolación cartesiana.

- Los ángulos de Euler, que componen tres rotaciones alrededor de los ejes de los mismos sistemas locales: *Ōx, Ōy y Ōz,* siendo *O* el origen de la articulación del miembro considerado. Esta representación es más intuitiva pero posee problemas de singularidad (en especial en las interpolaciones) y de continuidad en caso de conversión desde otra representación (donde tiene soluciones diferentes).

- Los cuaterniones que consisten en definir por un lado un vector de rotación unitario cuya base está en el centro de la articulación y por otro lado de un escalar que cuantifica la rotación del cuerpo alrededor de este eje. Estos están adaptados a la interpolación y son más intuitivos.

- Los mapas exponenciales donde el principio se deriva de los cuaterniones, pero en este caso la norma del vector de rotación proporciona su amplitud. Estos ofrecen las mismas ventajas que los cuaterniones, no se trata de una representación menos intuitiva, más bien es una representación más reducida, y por lo tanto menos compleja desde un punto de vista algorítmico.

El animador puede definir el método de orientación que más se adecue a su problema. Dado que los modelos del esqueleto virtual tienen muchos grados de libertad, el posicionamiento se torna largo y complicado. De manera que la complejidad de las instrucciones del usuario condiciona directamente la calidad de la animación mostrada.

El objetivo de la creación de un motor de movimientos es que el usuario no defina la posición del personaje para cada instante que compone la secuencia, sino que sea éste quien calcule las posiciones desconocidas mediante interpolación. Algunos autores han desarrollado interfaces (materiales y software) para simplificar el posicionamiento del personaje [PB88, Tha93]. A continuación mencionaremos otra técnica que auxilia al usuario en la generación de la animación, la cinemática inversa.

1.2.3 Cinemática Inversa

La cinemática inversa calcula las orientaciones de los segmentos de la cadena que permiten a cada extremidad ser llevada a una posición determinada, dependiendo de sus articulaciones, límites angulares y posición de su base. Esta técnica es muy utilizada en la resolución de problemas de manipulación. En el caso de la mano, el pecho permanece fijo y la cinemática inversa calcula automáticamente la orientación adecuada de los huesos del brazo. Dos tipos de solución existen para este problema.

- Métodos numéricos: proponen algoritmos de resolución que permiten considerar sistemas con restricciones y muchos grados de libertad.

- Métodos analíticos: están adaptados a las cadenas cinemáticas simples, y tienen la ventaja de ser más robustos.

El problema se formula de manera clásica de la siguiente forma:

$$\Delta X = J\Delta\Theta \tag{1.1}$$

Donde ΔX representa la diferencia entre la posición actual y la posición deseada, J es la matriz jacobiana del sistema, y $\Delta\Theta$ es el desplazamiento angular solución. En el caso de un sistema bien formado para el cual la matriz J es cuadrada y no singular, el cálculo de su inversa es suficiente para la resolución del problema.

En el caso del esqueleto del personaje virtual se prefiere emplear los métodos numéricos dado que el actor es una cadena cinemática compleja y redundante ($\dim(\Theta)>\dim(X)$). Particularmente, la ecuación anterior tiene infinitas soluciones. Para resolver el problema, se calcula la pseudo-inversa de la matriz jacobiana y la introducción de una restricción. Las restricciones pueden considerar la minimización de la energía, la minimización de la divergencia de postura respecto a una referencia, la evasión de las singularidades de la cadena [SS87], evitar las colisiones [Bai86], considerar los límites angulares de las articulaciones [GM85], o mantener el equilibrio [BMT96].

En el caso del control parcial del esqueleto se prefiere una solución analítica. Los brazos y las piernas tienen similitudes en los modelos comunes: 3 grados de libertad en cada extremidad y un central (rodillo o codo). La cinemática inversa a menor costo para estos conjuntos existe en la literatura [LS99, TGB00, SLG01], y la solución de Tolani es objeto de una biblioteca informática.

El usuario controla las extremidades de los miembros de su actor para darle vida. Fenómeno que simula la acción de un marionetista sobre un muñeco. Aprovecha un número más reducido de entradas que en el caso de la cinemática directa, y la definición de las dificultades subyacentes participan en la obtención de un resultado realista.

1.2.4 Las posiciones claves

Una animación esta compuesta por un conjunto de fotogramas que reproducen la sensación de movimiento. En el caso de personajes virtuales, estos pueden ser animados posicionando al actor personalmente durante todos los instantes que dura la animación. Dada la frecuencia de visualización de las imágenes (25 imágenes por segundo para la percepción humana), y el gran número de grados de libertad de los actores, esta solución no es factible para animaciones de larga duración. Para resolver esta tarea, existe un método que consiste en definir las trayectorias articulares como funciones definidas por pedazos [BW71, Stu86]. El usuario solo debe definir las posiciones del personaje en momentos claves de la animación (imágenes clave, posiciones clave, key - frames). El movimiento resulta de la

interpolación entre las posiciones claves realizada por el controlador de movimientos.

Existen algunas plataformas en el mercado que integran este tipo de técnica de animación, entre ellos destacan 3D Studio Max, Blender y Maya, más el método no es sensible en los casos de:

- Elección del tipo de interpolación
- La elección de la posición de imágenes clave en la animación.
- Las posiciones definidas por el actor.

Por lo tanto, su calidad depende de las capacidades artísticas del animador.

1.2.5 El control de los métodos cinemáticos

Un controlador de la locomoción calcula, en función de un comando de entrada, la posición del actor para cualquier instante de la animación. Este comando puede ser un camino a seguir o un juego de directivas de alto nivel (dirección, velocidad a respetar). En [MFCD99] encontramos un ejemplo de controlador de locomoción para personajes virtuales. El autor proporciona una reseña de las técnicas sobre las cuales está basado el controlador del actor virtual. Basado en la cinemática directa, el controlador sintetiza las trayectorias articulares para el esqueleto en vinculo con el control proporcionado. Basado en la cinemática inversa, el controlador sintetiza las trayectorias de los miembros efectores (por ejemplo, los cuerpos rígidos a las terminaciones de las cadenas cinemáticas o extremidades). El criterio de calidad de un controlador depende del realismo del movimiento obtenido y del nivel de abstracción de las directivas de entrada que simplifican sin empobrecer, la intervención del animador.

La biomecánica se ha encargado de la observación y descripción de la locomoción bípeda [IRT81], y la robótica se ha centrado en aplicar los conocimientos en los robots bípedos [GF74], en conjunto se ha desarrollado una buena cantidad de controladores [CC78, MTT87, Zs89, BMTT90b, BMTT90c, Mul98].

Badler et al [BPW92] proponen al problema de la síntesis de un controlador cinemático como la definición de la función f tal que:

$$\Theta = f(\Lambda, \Sigma, t) \tag{1.2}$$

Donde Θ es el vector del estado del personaje (el conjunto de las orientaciones de las articulaciones), Λ es el conjunto de las articulaciones del personaje, y Σ caracteriza la marcha (posición y orientación de cada pie, frecuencia y longitud de los pasos, etc.). f no puede generalmente ser explicita y la definición del controlador se da mediante la solución de Θ en función de los mismos parámetros.

La cinemática inversa interviene esencialmente en una fase de corrección de trayectorias. Se considera entonces una animación previamente calculada que se modifica en vías de considerar ciertas restricciones: límites angulares de las articulaciones, no contacto del pie con el suelo, etc. [BMTT90a, Chu00]. Además, se puede utilizar también para adaptar las leyes de los controladores cinemáticos para los nuevos personajes (donde varíen las dimensiones).

1.2.6 Conclusión

Existen diversas maneras de implementar un controlador de movimientos de acuerdo a nuestras necesidades. La cinemática directa e inversa nos proporcionan un amplio conjunto de herramientas para el diseño de un controlador que satisfaga nuestras necesidades de posicionamiento y animación. En la actualidad se empiezan a presentar controladores de alto nivel para el usuario, más aún existen limitaciones tales como la necesidad de un conocimiento profundo de la locomoción y personalización de la animación.

Podemos destacar que entre los investigadores existen dos corrientes principales: los que tratan de imitar el movimiento del mundo real mediante la captura de movimientos de actores reales y los que se enfocan en la generación de movimientos a partir de reglas y restricciones mediante la modelización y simulación dinámica.

1.3 Animación e imitación

La captura de movimientos es una disciplina muy extendida en la actualidad entre las tecnologías de generación de animación tridimensional. Su principio se basa por un lado en la medida y grabado del gesto humano, ya sea la evolución de la posición y de la orientación de los miembros de un actor real en el transcurso del tiempo. Por otro lado, los datos capturados se adaptan al modelo (cuya morfología es cercana a la del actor real): este modelo puede de esta manera reproducir los movimientos capturados. Al utilizar instrumental de precisión, los movimientos capturados permiten obtener animaciones fieles a las ejemplificadas e intrínsecamente realistas.

Desafortunadamente al ser considerada de manera aislada, esta técnica no es flexible: el gesto capturado y el gesto que se reproduce en la animación son parecidos, por lo tanto todos los movimientos que aparecerán en la animación deben ser previamente capturados. Los métodos de edición de estas capturas de movimiento reducen esta limitación y permiten la modificación y adaptación del movimiento.

1.3.1 Las técnicas de captura de movimiento

En la actualidad existen diversas metodologías para capturar el movimiento humano, entre ellos destacan:

- **Los sistemas de adquisición mecánicos**
  Se necesita que el actor real porte un exoesqueleto para medir los movimientos. Se estima la posición total del actor en base a la información proporcionada por los sensores colocados en las articulaciones. Las condiciones del ambiente de la captura no afectan las mediciones, sin embargo existen algunas restricciones importantes: el exoesqueleto usado por el actor reduce la libertad de movimiento y la posición global en el ambiente debe percibirse mediante un sistema externo.

- **Los sistemas de adquisición de video**
  Realizan una estimación del movimiento efectuado por un actor filmado: existen diversos algoritmos para la estimación del movimiento a partir de imágenes 2D. Su principio se basa en los modelos de movimiento [Roh94], sobre el video inverso [GBUP95], o aún en la estereoscopia que como el humano utiliza dos puntos de vista para extraer una información 3D de un par de imágenes 2D. El actor no necesita equipo especial, el material de trabajo no es costoso, una cámara y una unidad de cálculo para el tratamiento de las secuencias. No obstante es importante tomar en cuenta la existencia de una parte del cuerpo que se encuentra oculta. Además el análisis de la secuencia es complejo, necesita de la interacción con el usuario durante el tratamiento.

- **Los sistemas de adquisición sonora**
  Los actores son equipados con múltiples sensores de tres micrófonos, que detectan la señal de un emisor ultrasonoro. Conociendo la velocidad de propagación de la onda sonora en el aire, los sensores estiman su posición y orientación mediante triangulaciones geométricas. Este método de captura presenta dificultades relacionadas con las imprevisiones del entorno, como sonidos externos, obstáculos entre el emisor y los receptores y aun cuestiones difíciles de determinar como la temperatura del aire. Además ofrece un pequeño volumen de trabajo.

- **Los sistemas de adquisición magnética**
  En este caso, los sensores son sensibles al campo electromagnético modulado a base de frecuencias [JFOBBH00]. Son particularmente rápidos, dado que realizan un tratamiento pesado de los datos brutos y no sufren problemas de ocultamiento. Sin embargo no soportan la presencia de objetos metálicos en el medio ambiente.

- **Los sistemas de adquisición opto-electrónicos**
  Se caracterizan por tener múltiples cámaras y marcadores fácilmente localizables en la escena. Gracias a los múltiples puntos de vista, es más sencillo reconstruir la escena de manera tridimensional. Las fases de identificación y reconstrucción son complejas y costosas en tiempo de calculo, mas esta metodología ofrece mayor precisión, posibilidad de captura a gran frecuencia de muestreo, ausencia de restricción mecánica sobre los movimientos del actor, un campo de adquisición importante (si se cuenta con muchas cámaras y éstas están bien espaciadas).

El siguiente paso consiste en filtrar y modificar los movimientos medidos para adaptarlos a un esqueleto virtual articulado. Este modelo es entonces el modelo del actor real. En una película de animación es necesario que un actor real sea capturado y produzca todos los movimientos necesarios en el escenario. Existen métodos de adaptación de grabado: su principio es modificar las características, sin desvirtuarlos ni degradar su realismo.

1.3.2 Edición de las capturas de movimiento

Para adaptar las capturas de movimiento a un contexto preciso, sin necesidad de re-grabar una secuencia es necesario utilizar métodos de edición para modificar los detalles. Estas metodologías permiten:

- Modificar la morfología del esqueleto al que se le habían destinado inicialmente las capturas.
- Dadas las restricciones de un movimiento capturado, ser tomadas en cuenta y realizar un filtrado.
- Modificar las capturas hasta posturas determinadas de manera arbitraria por el animador.
- Interpolar las posturas para construir una animación a partir de diferentes posiciones.

1.3.2.1 Adaptación morfológica

Al tratar los datos y adaptarlos al modelo del actor real, el movimiento es fielmente reproducido en el proceso de la captura de movimiento. Por tanto es de alta prioridad personalizar la morfología del modelo. Los valores angulares deben ser modificados por el usuario. Los investigadores han desarrollado diferentes soluciones de adaptación morfológica.

El principio de restricciones espacio-temporales ha sido utilizado por Gleicher et al. [GL98, Gle98] para la adaptación del movimiento. Las características dinámicas del movimiento inicial son conservadas debido a las restricciones impuestas. La solución técnica más cercana es la desarrollada por Komura et al [KSK00, KS01] donde se agregan las capacidades musculares virtuales al modelo. Entonces seria posible una adaptación enriquecida con criterios fisiológicos.

Un esqueleto intermediario es utilizado para la adaptación morfológica por Monzani et al. [MBBT00]. La actualización entre las articulaciones del modelo viejo y el nuevo debe ser especificada por el usuario. Se realiza un seguimiento de las nuevas articulaciones en referencia con las anteriores de manera que se observe claramente la evolución del esqueleto. Las características del movimiento inicial son

conservadas debido a la especificación realizada por las restricciones cinemáticas (resueltas mediante cinemática inversa).

La descomposición de trayectorias articulares en splines jerárquicos propuesta en [LS99] aporta una adaptación en tiempo real. La solución permite desacoplar la adaptación de movimientos de alta y baja frecuencia. Choi y Ko en [CPK99, CK00] proponen otra solución en tiempo real al basarse en cinemática inversa y considerar las posiciones de ciertas partes del esqueleto como las manos y los pies. El nuevo movimiento se ve condicionado por una minimización de variaciones angulares.

Existen restricciones al modificar las proporciones del esqueleto, tales como la penetración de los pies en el suelo o el desplazamiento del pie de apoyo. Las violaciones de las restricciones son corregidas por Kovar et al en [KGS02] donde el estudio se enfoca principalmente en la adaptación de los movimientos de la base del cuerpo.

Algunas técnicas de edición se enfocan en modificar el contenido y otras sin embargo, se centran en la adaptación morfológica y tratan de preservar al menos el aspecto del movimiento inicial.

1.3.2.2 Restricciones espacio – temporales

El método de restricciones espacio temporales es general y es introducido por Witkin et al en [WK88]. El juego de restricciones definido por el usuario se utiliza como criterio de optimización del movimiento proporcionado en la entrada. Entre las restricciones podemos encontrar: respetar el equilibrio estático o de otras leyes de la física, disminuir la cantidad de energía utilizada, etc. El movimiento final es el resultado de la combinación de las restricciones establecidas por el usuario y las restricciones que preservan el contenido y el realismo inicial.

La necesidad de tratar el movimiento en bloque y la complejidad de los cálculos limitan la generación de una aplicación de restricciones en tiempo real. Sin embargo, el usuario define las zonas donde la deformación interviene, en caso de ser muy grandes, la búsqueda de solución es difícil y si son pequeñas restringen el espacio de soluciones.

1.3.2.3 Deformación cinemática

Witkin en [WP95] introdujo la técnica de deformación cinemática (motion warping) que permite deformar un movimiento en función de restricciones impuestas. Aquí, las restricciones están definidas por la modificación de la postura del actor virtual en momentos claves de la animación, elegidos por el animador. La postura original $\theta(t')$ es deformado hacia una postura $\theta'(t')$ por una transformación espacial $\theta'(t) = f(\theta,t)$ y temporal $t = g(t')$. Estas dos transformaciones deforman progresivamente el conjunto de posturas y permiten al personaje adaptar el movimiento modificado por el usuario

en la animación. Éste método también ha sido utilizado por otros autores como Bruderlin y Williams [BW95].

1.3.2.4 Mezcla de captura de movimientos

La interpolación entre diferentes capturas de movimientos (motion blending) posibilita la reutilización de capturas de movimientos existentes entre diferentes contextos o escenarios. Se pueden sintetizar nuevos movimientos al considerar simultáneamente diferentes capturas e interpolarlas.

Para lograr tal objetivo se debe clasificar el conjunto de miembros que interviene en la captura de movimientos. Entonces se identifica a cada captura de una manera distinta y se unen utilizando distintos métodos de interpolación. Por ejemplo, si el movimiento del brazo del modelo es dirigido por la captura "saludar" mientras que los otros movimientos están dirigidos por la captura "caminar", se puede obtiene entonces el movimiento "caminar saludando".

De hecho, cada articulación del personaje puede ser influenciada por otras capturas. En el siguiente bloque se exponen diversos tipos de interpolación que pueden ser adaptados a nuestra problemática.

Guo et al [GR96] extraen posiciones claves de las capturas candidatas a la interpolación. Las capturas son sincronizadas entre ellas por deformación temporal. Cuando la duración de los ciclos varía, ésta técnica es adaptada a la locomoción humana. Por ultimo, las capturas pueden ser mezcladas de manera lineal.

En [WH97], se propone una proyección de los datos capturados en un espacio apropiado en la interpolación. Se comienza construyendo un conjunto de capturas de movimientos. Se clasifican por tipos, por ejemplo, diversos movimientos de manipulación. Cada movimiento debe tener una característica distinta que solucione un problema específico, como diferentes distancias o alturas. Con los diferentes movimientos se pueden generar nuevas animaciones. Se selecciona el movimiento que se aproxima más a la condición deseada y se realiza una ligera interpolación para modificar el movimiento hasta que se encuentra en la situación que deseamos.

En [BBET97] se definen tres modos para cada tipo de acción, inicialización, ejecución y terminación. Las acciones son elegidas por el animador según el contexto, después se ejecutan de manera concurrente. Estas son seleccionadas según un orden de prioridad establecido, dadas las circunstancias, algunas acciones pueden inhibir a otras, o varias ser ponderadas e interpoladas, más aun ser ejecutadas unas después de otras.

Este tipo de interpolación necesita la consideración de captura inicial, dicha posturas se establecen como posición inicial, para la sincronización y correspondencia de animaciones posteriores.

1.3.3 Control de las técnicas de edición de capturas

Los controladores de movimiento representan la solución para la edición de capturas. El objetivo de los controladores de movimientos es la síntesis de animaciones con alto nivel de naturalidad basadas en la utilización de comandos de alto nivel. El modelo ejecuta de manera automática todos los comandos internos involucrados en la resolución del problema planteado. A continuación mencionaremos algunos trabajos realizados en el área de controladores de movimiento.

Las características generales y naturaleza del movimiento permanecerán intactas en el problema de la adaptación morfológica. Por lo tanto solo es necesario redefinir el esqueleto animado. Komura [KSK00, KS01] considera otros criterios que tienen influencia en el movimiento, como los fisiológicos en la ocurrencia.

La modificación de los detalles del movimiento al adaptarlo al ambiente [WK88, Gle97, GL98] o al corregir los defectos [PW99] es permitida por las restricciones espacio - temporales. Aunado a la tarea inicial, es posible integrar tareas secundarias al movimiento principal, como puede ser la manipulación.

Algunos autores han optado por la elaboración de múltiples animaciones y su agrupación en las llamadas bibliotecas de movimientos. En [WH97, Ros99] se describen diferentes estructuras de biblioteca que determinan los comandos posibles para sintetizar la animación. Dado un conjunto de tareas a realizar, se emplea la compleja estructura de biblioteca que permite la dirección de la animación.

1.3.4 Conclusión

Se puede sintetizar nuevas animaciones a partir de las existentes, mediante las técnicas de edición de capturas de movimientos. El usuario puede apreciar animaciones realistas al usar un controlador de movimiento que sintetice las posturas del personaje de manera automática. Las técnicas de reutilización de capturas de movimiento existentes reducen los costos de producción de animación de alta calidad. Sin embargo existe otra tendencia en la animación de personajes que considera las características físicas del entorno y los actores virtuales.

1.4 Animación y simulación

Esta corriente de la animación se basa en la creación de un motor de simulación de comportamiento físico para cada entidad. El motor cuenta con métodos dinámicos que se apoyan sobre las leyes de la física para calcular los movimientos resultantes de las acciones sobre el modelo. Dado que el motor calcula las trayectorias nuevas en base a las fuerzas aplicadas, se obtiene un realismo importante. El modelo se vuelve más complejo a medida que aumenta la cantidad de cuerpos que componen la simulación.

1.4.1 Principio

El conjunto de reglas que rigen el comportamiento físico de los modelos dinámicos se resume a continuación:

- Un modelo mecánico que regula las ecuaciones de movimiento
- Un juego de fuerzas y de pares aplicados a la entrada del modelo
- Un sistema donde el estado se calcula a la salida del modelo.

Existen dos formalismos importantes que contienen las ecuaciones del movimiento. Los principios fundamentales de la dinámica de los cuerpos sólidos están contenidos en el formalismo de Newton – Euler (Ecuaciones 1.3 y 1.4):

$$\sum_{i=0}^{n} \vec{f}_i = m\vec{a} \tag{1.3}$$

$$\sum_{i=0}^{m} \vec{c}_i = I\vec{\omega} + \vec{\omega} \wedge I\vec{\omega} \tag{1.4}$$

Donde $\vec{f}_i$ son las fuerzas y $\vec{c}_i$ los pares aplicados al sistema, m es su masa e I la matriz de inercia, $\vec{a}$ es el vector aceleración del sistema y $\vec{\omega}$ el vector velocidad angular.

Otro formalismo basado en la expresión de la energía cinética y potencial del sistema es el planteamiento de Lagrange, además incluye restricciones mecánicas como articulación de los cuerpos e inercias.

$$Q_i = \frac{d}{dt}\left(\frac{\vartheta L}{\vartheta q_i}\right) - \frac{\vartheta L}{\vartheta q_i} - C_i \tag{1.5}$$

Donde q_i es el i-ésimo parámetro del sistema, Q_i son las acciones generalizadas relativas al i-ésimo parámetro, L es el lagrangiano del sistema y C_i representa las restricciones holonómicas y no – holonómicas del sistema.

La respuesta al problema de locomoción de actores virtuales está representada por el conjunto de fuerzas que se deben aplicar al esqueleto para que éste adquiera la postura deseada. La solución a este planteamiento está propuesta en la creación de controladores dinámicos.

1.4.2 Control dinámico del personaje virtual

Algunos autores se han basado en la locomoción para crear su controlador dinámico [RH91, KB96]. También se han enfocado en los gestos deportivos [HWBJFO95], y otros en los movimientos de los brazos [Mul98].

El realismo se incrementa de manera considerable en aquellas animaciones basadas en modelos y simulación dinámica. Sin embargo su complejidad restringe la interactividad. Otro tópico que debe resolverse es la dificultad ligada a la colocación de la capa de comportamiento del actor (como son el mecanismo de interacción y los controladores). Al filtrar las violaciones hechas a las leyes de la física, los modelos dinámicos son utilizados para calcular las deformaciones cinemáticas de los movimientos capturados [SYN01].

1.5. Planificación de la locomoción

El problema de la planificación de la locomoción para los personajes virtuales se describe como: dada una posición inicial y final en un ambiente que contiene obstáculos, se debe calcular el movimiento de desplazamiento realista del personaje uniendo estas dos posiciones sin colisión con el ambiente.

1.5.1 Planificación y robótica

Comúnmente el problema de la planificación de movimientos es conocido en la robótica como el problema de mover el piano. Y en el se establece un ambiente tridimensional compuesto por obstáculos y se refiere al paso sin colisión de un cuerpo de una posición de partida a una posición objetivo.

T. Lozano Pérez [LP83] en los años 80s formuló el problema, de manera que el problema de mover un robot en su espacio de trabajo se convirtió en el movimiento de un punto en el espacio de configuraciones del sistema. El movimiento de un cuerpo sólido cualquiera necesita ser razonado en un espacio de seis dimensiones (tres parámetros de posición y tres de orientación). En este modelo, el problema de planificación de mover un cuerpo en 3D, se convierte en un problema de planificación de un punto en el espacio de configuraciones de los cuerpos.

Para resolver dicho problema, se tiende a explorar las componentes conexas del espacio de configuraciones admisible (sin colisionar) [Lat91]. La investigación en este tópico ha tomado dos tendencias que se detallan a continuación.

Los equipos de robótica, particularmente en manufactura renuncian a una representación exacta del espacio de configuraciones admisibles para un brazo de manipulación. Se enfocan sobre el estudio de algoritmos del cálculo de la distancia entre cuerpos en R^3 y establece estructuraciones aproximadas del espacio de configuraciones admisible. Aparecen rápidamente limitaciones debidas a la potencia

de cálculo disponible. Los espacios que son posibles de construir y explorar son los de baja dimensión (inferior a 4).

De manera paralela, los equipos de matemáticos y especialistas en geometría computacional se interesan en el problema e intentan desarrollar soluciones exactas. Debido a la complejidad del cálculo, estos métodos solo permiten resolver problemas sencillos que están fuera de las necesidades reales.

Al mismo tiempo, se presentaron métodos locales de evasión de obstáculos. En [Kha86] se introduce el método de potenciales donde el objetivo a alcanzar ejerce un potencial atractivo sobre el sistema, los obstáculos representan un potencial repulsivo. De manera que al seguir el gradiente del campo potencial, se genera un movimiento que evita los obstáculos. Sin embargo, existen mínimos locales dentro de estos métodos que deben ser tratados por un método de planificación.

Durante los noventas, los investigadores dejan a un lado el estudio formal de la estructura del espacio de configuraciones admisible y se enfocan en la exploración de la capacidad incremental de los medios de cálculo. No se busca más construir representaciones explicitas del espacio de configuraciones admisibles; se busca explorar apoyándose en las representaciones implícitas.

Los primeros éxitos retoman la idea de los métodos de potencial y la aplican en una representación implícita del espacio de configuraciones bajo la forma de una rejilla. Cuando un mínimo local se alcanza en el espacio de configuraciones, el algoritmo produce una marcha aleatoria durante un tiempo aleatorio [BL91]. Es muy difícil de analizar el comportamiento del algoritmo pues encontrará una solución (si esta existe) en un tiempo de calculo donde la esperanza es finita.

Por otro lado, en [KSL096] se elabora un roadmap, que está definido como un grafo compuesto por puntos muestreados de manera aleatoria en el espacio de configuraciones. Al aplicar cálculos de distancia en el espacio, se puede determinar si el punto corresponde a una configuración admisible o no. Una vez concluida la fase de aprendizaje (creación del grafo), se puede resolver el problema agregando los arcos de partida y llegada (fase de consulta). La complejidad de este método es completa en probabilidad y tiene un buen desempeño en la implementación. Hasta la actualidad, se han desarrollado numerosas variantes, como pueden ser: muestreo cerca de los bordes del espacio de configuraciones admisibles [BOvdS99], evaluación perezosa de la admisibilidad de un camino local [BK00, S03], optimización del número de nodos [SLN00].

Los métodos de difusión son inspirados por los métodos que usan marchas aleatorias y los basados en muestreo. Estos son aplicados en los contextos donde solo se debe resolver un problema, en este caso, la fase de aprendizaje no se aplica, solo el punto de partida y el objetivo son los datos de entrada del algoritmo. Se difunde progresivamente la búsqueda a partir de la posición de inicio haciendo muestreos aleatorios en la vecindad. De manera que la posición inicial se convierte en el primer nodo del roadmap. Las configuraciones admisibles son generadas de manera aleatoria en la vecindad de este nodo y permiten crear nuevas ramas. Dichas

configuraciones se convertirán en los nodos terminales del árbol (hojas). El proceso de generación se centra en la vecindad de las hojas. Las hojas pendientes al objetivo determinan las direcciones privilegiadas de crecimiento del árbol y establecen la condición de salida [HLM97, KL00].

1.5.2. Planificación y animación de personajes

A continuación se describen diversos métodos de planificación que han sido aplicados con éxito en la animación de personajes, con el objetivo de aumentar su autonomía de movimiento. Las soluciones de desplazamiento son propuestos a partir de un estado dado y un objetivo. El nivel de control propuesto de esta manera es por lo general muy abstracto. Las soluciones a los problemas de planificación de la locomoción se basan en una doble elección técnica: la del método de planificación y la del controlador de locomoción para el personaje.

Kuffner propuso una extensión de los métodos de planificación a personajes animados en [Kuf98]. El problema se compone por un ambiente (terreno plano) y un personaje. Al personaje, se le liga con un objetivo que debe alcanzarse así como la captura de movimiento de un único ciclo de marcha. En una segunda etapa, el ambiente se modela como un plano 2D (vista superior), con el fin de determinar los obstáculos y zonas libres para un cilindro que engloba al personaje en marcha. Para evitar colisiones, los obstáculos se hacen crecer de acuerdo al radio del cilindro, y posteriormente se traza una rejilla al plano. A cada celda de la rejilla se le asigna una etiqueta que indica si esta libre u ocupada. Para alcanzar el objetivo se realiza una búsqueda en la rejilla utilizando algoritmos clásicos de programación dinámica (Dijkstra o A*). A lo largo del camino solución, el personaje es animado en base a la captura de movimiento elegida. El método es suficientemente efectivo para considerar un ambiente dinámico (obstáculos móviles) y para interacción en tiempo real con el usuario.

Cuando un obstáculo móvil bloquea el paso del personaje, el camino solución anterior se invalida y entonces se realiza nuevamente la planificación con un plano 2D actualizado. Cabe destacar que la ausencia de una solución provoca una animación de estado de espera del personaje. Y dado el punto de vista humano, este tipo de animación con estados de espera carece de naturalidad convincente.

Choi et al en [CLS03] proponen una planificación de la locomoción basada en los métodos probabilistas. Se propone la generación de un roadmap representado por el grafo donde los nodos definen las colocaciones para los pies del personaje. Para insertar un arco, se debe considerar la siguiente condición: la deformación de un movimiento capturado para establecer las posiciones de los pies (definidos por los nodos) debe estar limitada. Esta solución incrementa considerablemente el realismo en la síntesis de la locomoción, y permite tratar los casos de terrenos accidentados. Sin embargo la implementación no considera la evasión de obstáculos en tres dimensiones: si un obstáculo se encuentra en la parte superior, el personaje no se agachará para esquivarlo.

Salomón et al [SGLM03] se interesan en los métodos probabilistas. Dichas soluciones son aplicadas en el caso de la planificación de la locomoción en ambientes de grandes dimensiones. La solución muestra un nuevo ejemplo de aplicación de los métodos robóticos de planificación a la navegación de personajes virtuales. Actualmente, no se ha propuesto algún acoplamiento a un controlador de personajes, solo se produce una trayectoria a seguir.

En [RAL00] se propone un método de planificación de locomoción en diversas etapas para el personaje virtual "Virtual Robot". En primer término, se planifica una trayectoria al hacer un modelo simplificado del ambiente. La trayectoria mencionada se ejecuta en el ambiente original y se emplea un conjunto de sensores simulados para tomar en cuenta los obstáculos que no se habían considerado en el inicio. Para evitar la colisión con los objetos no previstos se emplea una estrategia que depende de la naturaleza de los obstáculos encontrados. Esta metodología muestra un comportamiento reactivo, lo que permite el tratamiento de un ambiente dinámico. Sin embargo presenta un conjunto de debilidades: se requiere información sobre los objetos de la escena para utilizar la estrategia de evasión de colisiones y a menudo se tiene que ejecutar nuevamente la rutina de planificación inicial.

Al mezclar diferentes tipos de locomoción, Shiller et al en [SYN01] proponen la elaboración de planes de navegación. La rutina de planificación inicia al discretizar el ambiente en celdas que forman una rejilla. Para cada tipo de locomoción, se asigna una caja que engloba el movimiento de desplazamiento. Al tomar en cuenta sucesivamente cada caja envolvente y las celdas del ambiente, se construye un grafo de navegación para cada tipo de locomoción. Dado que todos los grafos están basados en la misma rejilla, es posible superponerlos. Si un mismo punto de la rejilla es válido en dos grafos de navegación diferentes, puede ser cambiado el tipo de locomoción en la celda correspondiente. Se pueden entonces elegir entre estas dos estrategias:

- Tomar un borde de un grafo de navegación dado, correspondiente en la animación a un desplazamiento entre dos celdas utilizando el tipo de locomoción asociado al grafo

- Pasar de un grafo a otro permaneciendo en el mismo punto de la rejilla, lo que en la animación corresponde a un movimiento de transición de un tipo de locomoción.

Al ser definido un problema de planificación, el algoritmo de Dijkstra devuelve la solución óptima que combina estas dos clases de movimiento. El control del personaje que realiza este tipo de planificación es posible gracias a diversas técnicas de edición de capturas de movimiento, filtradas de manera dinámica para hacer el resultado más coherente. Una contribución de este método es la posibilidad de combinar diversos tipos de locomoción, y encontrar pasajes en ambientes relativamente restringidos.

1.5.3. Movimientos reactivos

En el caso de ambientes estructurados, el problema de planificación de movimiento de robots móviles va desde la planificación de trayectorias hasta el control de trayectoria. Al ingresar en ambientes desconocidos o mal modelados, se induce la necesidad de tomar en cuenta eventos no planeados o dinámicos. De manera que el personaje simulado reaccione como lo haría un ser vivo. Por lo tanto, el comportamiento reactivo representa un papel importante cuando el robot tiene que moverse en un ambiente no estructurado o dinámico.

Los reflejos artificiales diseñados para robots móviles se definen como la habilidad de reaccionar cuando un evento no planeado ocurre, por ejemplo el desplazamiento en ambientes desconocidos o dinámicos. El problema de diseñar sistemas artificiales de acciones reflejas ha sido parcialmente resuelto al establecer relaciones entre las entradas (estímulos) y las salidas (acciones) a través de máquinas de estado [Bro89, And90]. Dichas máquinas son programadas con lógica difusa, redes neuronales, máquinas de estado deterministas, etc. Otra propuesta de diseño considera la simulación de sensores que envían señales de información a la unidad de control del robot. El método más difundido de este tipo es el método de potencial desarrollado por O. Khatib en los años ochentas [Kha86]. En la primera propuesta, la estimulación es traducida en fuerzas externas virtuales que son sencillamente sumados a la fuerza de atracción de la meta, cuando hay una; todas estas fuerzas producen una acción, moviendo el sistema y modificando su posición en el mundo. Por otro lado, los sensores estereoceptibles proporcionan la información generada por las fuerzas externas virtuales que auxilian en la evolución del sistema, posteriormente se realiza una comparación con las fuerzas de atracción de la meta predefinida.

Durante los últimos 15 años, los investigadores en robótica móvil se han interesado en el problema de los modelos reactivos para la evasión de los obstáculos [Zap91, ZLT94]. Dentro de esta tendencia, los trabajos apuntan a proporcionar un planificador práctico que considere acciones reflejas. Profundizando en esta propuesta de control mediante métodos reactivos, destaca la investigación realizada recientemente por René Zapata et al [ZLT94] donde se expone la zona virtual deformable (DVZ). Dicha propuesta propone la suposición de que el robot esta rodeado por una DVZ que depende geométricamente de la cinemática del robot sin ninguna modelación de los obstáculos, y las deformaciones son debidas a la intrusión de obstáculos en la zona de seguridad, así los sensores generan información referente a la proximidad de los obstáculos dentro del espacio del robot. La metodología usada por la DVZ consiste en minimizar estas deformaciones no controladas debidas al ambiente actuando directamente sobre los controles del vehículo. La finalidad de dicho proceso es obtener una zona cuya forma dependa únicamente del móvil, modificando de manera local el vector control (aceleración, orientación, etc.) [ZLT94].

1.6. Nuestro enfoque

En nuestro trabajo consideramos como objetivo primordial la autonomía de locomoción del personaje animado. Nuestros objetivos secundarios nos han condicionado la elección de los métodos previos de la resolución del problema. Entre ellos destacan:

- La obtención de una planificación eficaz. La metodología de planificación se relaciona directamente con la complejidad de los métodos de locomoción autónoma. Dicha planificación debe considerar ambientes restringidos, complicados o de muchas dimensiones.

- Una cinemática realista. La aplicación desarrollada tiene como objetivo primordial a los animadores. La calidad de nuestros resultados esta directamente relacionada con el aspecto visual que muestran. Dicho realismo es representado por la trayectoria descrita y la secuencia de movimientos efectuados por el personaje.

- Una arquitectura flexible. Para dotar de continuidad al trabajo realizado, se planteo una estructura modular en la que se pueden integrar nuevos elementos en el futuro.

- Editor de capturas de movimiento y apariencia. Para dotar de un mayor realismo se incluye la capacidad de creación y modificación de capturas de movimiento, escenarios y vestimentas del personaje.

La planificación de la locomoción se resuelve al elegir dos metodologías, la que corresponde a la planificación y otra que representa la cinemática del personaje. El muestre aleatorio realizado en los ambientes virtuales, nos provee la información necesaria para realizar la planificación de la trayectoria.

En lo concerniente a la animación del personaje, se han planteado diversas metodologías para modificar capturas de movimiento que han mostrado resultados con alto nivel de realismo. Gracias al diseño del controlador cinemático, se puede acoplar de manera sencilla la animación del personaje a la trayectoria realizada. Dicha metodología se basa en los trabajos de [Kuf98, PLS03] en lo concerniente a la evasión de obstáculos en 3D. Al igual que en estos estudios previos, nuestra solución considera la planificación y la animación en dos cuestiones separadas.

Nuestra aplicación tiene características modulares, de manera que en tiempo de ejecución pueden ser agregados, eliminados o editados diferentes elementos que forman parte de la animación. Por lo tanto hemos identificado rigurosamente las entradas y salidas de cada componente. Y se han establecido estructuras de datos similares para los bloques semejantes.

En la figura 1.2 se detalla la organización de los módulos que forman nuestra solución. Se establecen dos niveles principales, la planificación esta representada por el nivel superior y la animación en tiempo real por el inferior. La columna de la

izquierda son los datos internos que restringen el funcionamiento del sistema como son: el ambiente, el personaje, y la biblioteca de movimientos. En la columna central se describen los principales módulos desarrollados. Finalmente la columna de la derecha indica el tipo de datos que los módulos utilizan como entrada y salida.

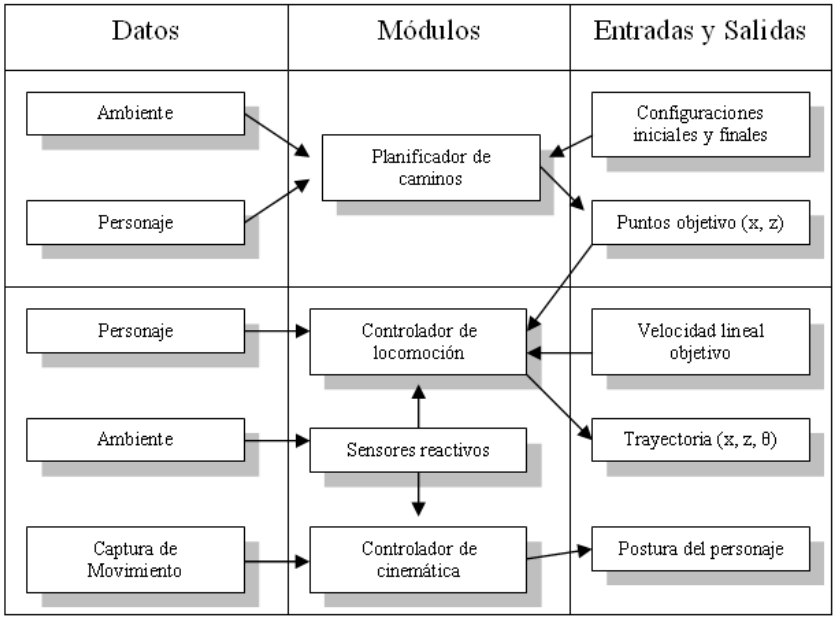

Figura 1.2: Arquitectura del sistema

1.6.1 Entradas y salidas

Para obtener una animación terminada, es necesario proporcionar al sistema un conjunto de datos de entrada que se describe a continuación. En primer lugar, se asigna un escenario prediseñado, se establecen las restricciones del personaje (posición inicial y final, tamaño y apariencia), se asignan las capturas de movimiento que serán modificadas y la velocidad lineal máxima que puede adquirir el personaje durante la animación.

Cada modulo del sistema proporciona una salida distinta, que puede ser reingresada para enriquecer las otras partes del sistema como pueden ser, las señales internas proporcionadas por los sensores simulados, las velocidades y cambios de trayectoria obtenidos por el controlador de locomoción y la modificación en las posturas y capturas de movimiento generadas por el controlador de cinemática; finalmente la salida más importante es representada por la animación construida en tiempo real por el modulo de animación.

1.6.2 Módulos

Se han desarrollado e integrado cuatro módulos principales. El primero en consultarse es el planificador (primera etapa), que toma como parámetro de entrada el ambiente virtual, y las restricciones de personaje, como son el tamaño, la posición

inicial y posición final. La salida del módulo viene representada por un conjunto de puntos objetivo (x, z) que el personaje tiene que alcanzar para llegar a su meta. Dichos puntos marcan una trayectoria preliminar que el personaje recorrerá posteriormente.

Los siguientes componentes son consultados durante la segunda etapa del proceso. Al llevarse a cabo la animación en tiempo real, se integró un módulo de sensores reactivos que informan al personaje de la situación de su entorno.

Dado que el sistema considera aspectos físicos como la velocidad lineal, velocidad angular y aceleración, los sensores reactivos informan al controlador de locomoción sobre la proximidad de los obstáculos de su entorno. El controlador de locomoción es quien modifica la velocidad lineal y dirección del personaje en base a lo requerido por el usuario y a las señales recibidas por los sensores reactivos.

Además el controlador de cinemática modifica la postura y capturas de movimiento en presencia de obstáculos no planificados como los superiores (Warping). Dicho controlador modifica las capturas de movimiento en base a la información referente a la velocidad lineal y angular proporcionada por el controlador de locomoción.

1.6.3 Organización del documento

En el capitulo 2 se detallan los elementos que componen a nuestro personaje, es decir, su estructura de datos, su modelo de cinemática y la composición de su apariencia.

El método de planificación, optimización y adaptación de las trayectorias para personajes virtuales, se explican en el capítulo 3. Destacamos que el método desarrollado es RRT bi-balanceado, se aplica un doble método de optimización recursivo a la trayectoria previa planificada y finalmente se realiza un ajuste con curvas de Bézier para dotar de naturalidad humana al conjunto de puntos objetivo.

En el capitulo 4 se muestran los resultados obtenidos, además describe el sistema diseñado y las pruebas realizadas.

Finalmente en el capítulo 5 se presentan las conclusiones y posibilidades de trabajo futuro.

Capitulo 2. Modelización del personaje y manejo de capturas de movimiento.

Para alcanzar una animación realista, nuestra solución utiliza tanto el método de planificación basada en árboles aleatorios de exploración rápida como la edición y combinación de capturas de movimiento. El modelo del personaje debe ser compatible con estas técnicas. Para esto se toman en cuenta los siguientes elementos:

- El esqueleto cinemático: éste proporciona las dimensiones y la articulación de los cuerpos rígidos entre ellos.
- La vestimenta del actor virtual: este elemento es la expresión geométrica de los cuerpos rígidos.
- Las capacidades de movimiento: éstas se describen bajo la forma de una biblioteca de captura de movimientos y posturas.

2.1 Mecánica del actor virtual

El sistema tiene la capacidad de utilizar cualquier tipo de estructura cinemática que sea definida en un archivo de esqueleto. Por cuestiones de estandarización comúnmente se emplea un esqueleto cinemático compuesto por cuerpos rígidos, que cuenta con 52 grados de libertad: 6 grados de posicionamiento de la raíz y 46 grados articulares para la orientación de los cuerpos que forman la estructura (Figura 2.1).

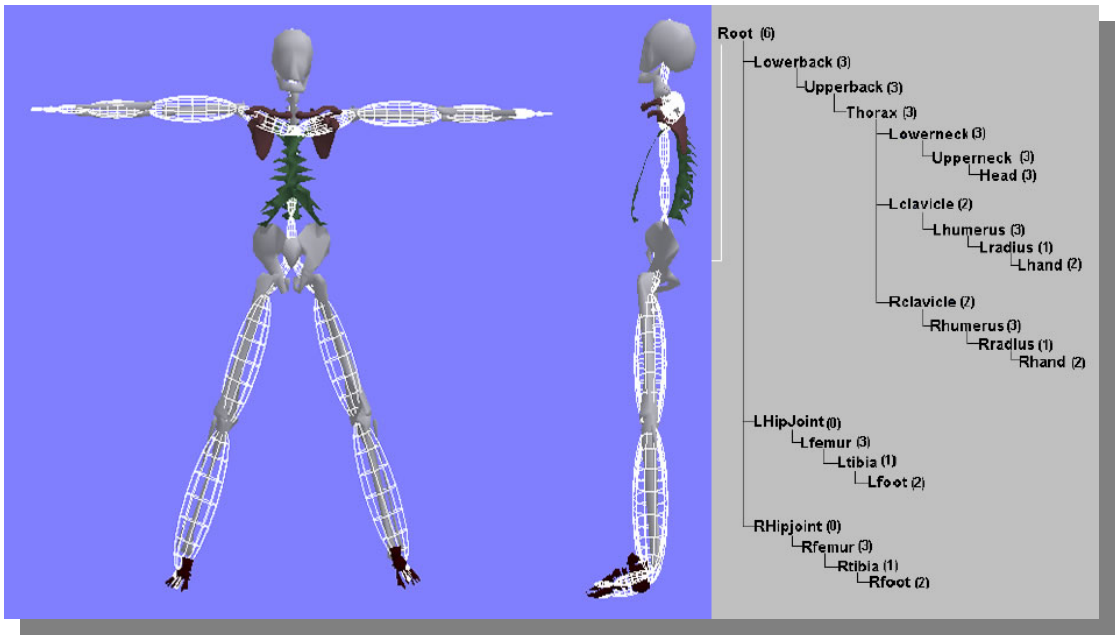

Figura 2.1: Estructura cinemática del personaje

En este tipo de modelo, la raíz de la cadena cinemática esta ubicada a nivel de la pelvis. Desde ahí se extienden 5 ramas abiertas: dos hacia las piernas, dos hacia los brazos y una hacia la cabeza. La pelvis cuenta con 6 grados de libertad de posicionamiento: 3 de traslación y 3 de rotación. Las otras articulaciones son esféricas con 3 grados de libertad. Dado que se establecen como sólidos rígidos, solo cuentan con capacidad de rotación. La estructura se adaptó al problema de la planificación de movimientos de personajes virtuales. Solo se tomaron en cuenta los grupos estructurales de principal tamaño para reducir la complejidad del cálculo.

2.2 Aspecto visual del personaje virtual

El aspecto físico del personaje viene definido por la asociación de un conjunto de entidades geométricas tridimensionales en cada cuerpo rígido de la estructura cinemática. Dicha asociación se lleva a cabo mediante la manipulación del editor de vestimentas del personaje.

El sistema cuenta con una herramienta de importación de entidades tridimensionales generadas por diversos programas independientes. Estas entidades son alojadas en una estructura de datos denominada biblioteca de entidades de personaje. Después de definir un esqueleto cinemático, es posible asignar una cantidad variable de entidades tridimensionales a cada parte del modelo. Las formas geométricas del personaje, al igual que las del escenario están definidas por un conjunto de listas de facetas triangulares no deformables.

Hemos definido varias vestimentas para el personaje, que son ilustradas en la figura 2.2. Las vestimentas se mueven en conjunto con las rotaciones de las articulaciones del modelo cinemático. El posicionamiento de las entidades en el modelo es empírico dado que no se cuenta con la localización precisa de los sensores sobre el actor real después de la generación de la captura de movimientos.

Figura 2.2: Vestimenta del personaje virtual.

2.3 Miembros activos y reactivos.

Para incrementar el realismo de la animación, se han separado los miembros del personaje vinculados a la locomoción. De manera que al momento de la planificación de los puntos objetivo, se busca solamente una trayectoria despejada para los miembros locomotores. Posteriormente la animación se genera a partir de dichos puntos. Si los otros miembros entran en colisión, el personaje modificará su postura sin intervenir con los miembros encargados de locomoción con el objetivo de evadir el obstáculo. Por lo tanto, los grados de libertad del actor virtual son clasificados en activos y reactivos. En este caso, los miembros locomotores forman parte de los grados activos.

La etapa de planificación se lleva a cabo antes de la animación. Para reducir la complejidad del cálculo de colisiones al momento de la planificación, se representa la parte activa del personaje como un cilindro solidó y no deformable (Figura 2.3). El tamaño del cilindro es especificado por el usuario tanto en radio como en altura y el modelo cinemático y visual del personaje adquieren sus características a partir de las del cilindro. Por tanto el personaje puede adquirir diferentes tamaños al momento de llevar a cabo la animación.

Los cuerpos restantes se etiquetan como reactivos. Dichos miembros se muestran también en la figura 2.3. Para incrementar la naturalidad de los movimientos del personaje virtual, se agrupan todas las partes reactivas en un solo grupo. La utilidad del grupo reactivo viene dada al momento de una posible colisión, de manera que el modelo cinemático puede modificar la postura del grupo para evitar el obstáculo. Este movimiento de evasión se busca entonces para todo el grupo cinemático.

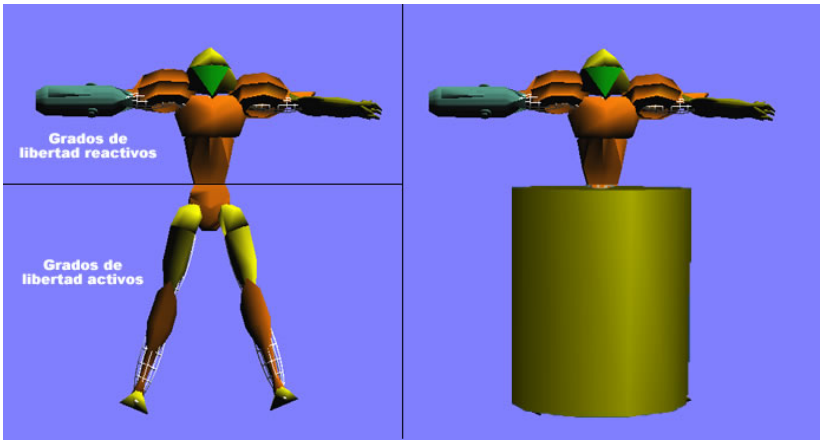

Figura 2.3: Forma semi-envolvente y grados activos y reactivos del personaje

La clasificación de miembros activos y reactivos en el modelo del personaje virtual depende del tipo de locomoción que se realice. En particular la distribución elegida coincide con los movimientos de un humano erguido. Sin embargo los miembros activos y reactivos cambiarían en el caso de una marcha a cuatro piernas y la forma semi-envolvente seria de un aspecto diferente.

2.4 Las capturas de movimiento

La animación del personaje se basa en la interpolación de capturas de movimiento. El principio general de estos métodos es modificar movimientos capturados y de esta manera obtener una nueva animación. La captura de movimiento original y la editada difieren en tanto a los valores de las rotaciones de los miembros del modelo cinemático, sin embargo conservan el sentido. Un conjunto de capturas de movimiento define entonces las capacidades de locomoción del personaje.

Las capturas de movimiento son alojadas en una biblioteca de movimientos. Esta biblioteca clasifica los movimientos en fijos, caminar y correr. El sistema automáticamente varía la frecuencia y valor de las posturas en base a la velocidad que el usuario indique.

2.4.1 Archivos de esqueleto (.asf)

En el archivo .asf (Acclaim's skeleton file) se define un punto base que representa el nodo raíz de la información de movimiento. Cada segmento del archivo contiene información acerca de la manera en que el segmento será dibujado así como la información que puede ser usada por programas de física dinámica o de cinemática inversa. En este tipo de modelos, sólo existe una raíz y los otros miembros se ajustan a ésta. El documento esta organizado de manera sencilla, se trata de información alojada en formato de texto en un árbol jerárquico. A cada miembro del modelo cinemático se asigna un identificador de hueso (figura 2.4) y sus capacidades de movimiento (restricciones en los grados de libertad).

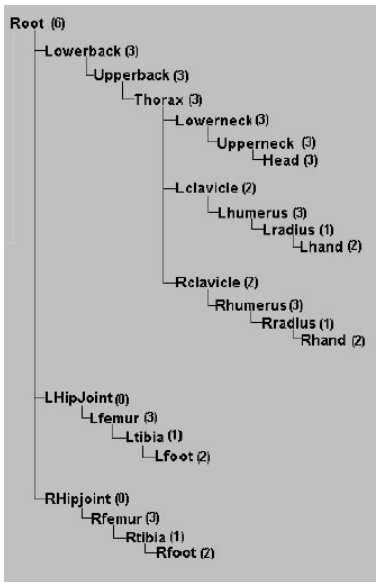

Figura 2.4: Esqueleto jerárquico (archivo .asf)

2.4.2 Archivos de captura de movimiento (.amc)

Acclaim es una de las compañías con más experiencia en el desarrollo de videojuegos, y ha realizado importantes investigaciones en el área de captura de movimientos. Dicha compañía estableció el formato de alojamiento que posteriormente acogería Metrics, la compañía creadora del sistema de captura de movimiento llamado Vicon, para preservar sus movimientos capturados.

Acclaim's motion capture (.amc) son los datos de captura de movimiento (figura 2.5). A cada archivo de captura de movimiento esta asignado una estructura cinemática similar a la mostrada por la figura 2.4.

Al igual que en el archivo de esqueleto, el formato de los archivos de movimiento es muy sencillo. Los datos se agrupan por recuadros, y contienen el nombre de cada miembro del esqueleto y posteriormente los valores asociados a cada grado de libertad. Los nombres de cada parte del personaje están organizados de la misma manera que en la especificación del esqueleto.

Figura 2.5: Movimiento contenido en un archivo (.amc)

2.5 El modelo matemático de locomoción del personaje

La locomoción humana se caracteriza por ser frontal, no en reversa ni lateral. Particularmente, la dirección de la visión del sujeto típicamente coincide con el movimiento en el espacio. De manera que al igual que Kuffner en [Kuf99] hemos adoptado el modelo matemático basado en un disco orientado para diseñar la locomoción. El centro del disco corresponde al punto proyectado del centro de la geometría del personaje sobre el plano horizontal. La orientación del disco viene representada por la dirección frontal del personaje como se presenta en la figura 2.6. La velocidad lineal del personaje esta condicionada a coincidir con la dirección de visión del personaje. Esto responde a la habilidad del personaje para caminar y correr. El giro del modelo viene representado al considerar la velocidad angular del círculo sobre su centro.

Para una mejor exposición denotaremos los elementos del modelo de la siguiente manera.

- P_t Posición (x_t, z_t) del centro del disco
- θ_t Orientación (dirección frontal del modelo)
- υ_t Velocidad lineal hacia la dirección θ_t
- ω_t Velocidad angular respecto a P_t

La tupla $(P_t, \theta_t, \upsilon_t, \omega_t)$ representa el estado simulado del personaje en el tiempo t respecto a su locomoción.

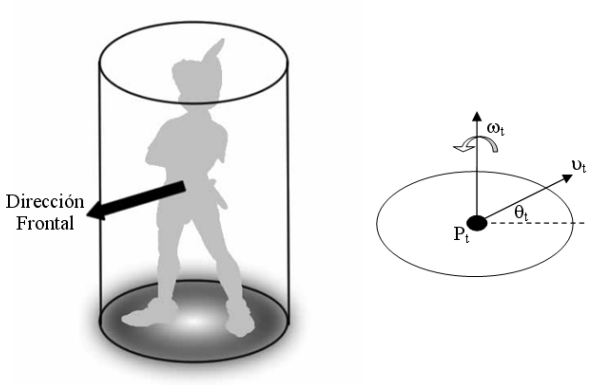

Figura 2.6: Disco que representa al personaje en el controlador de locomoción

En el modelo también fueron considerados los siguientes elementos:

- a_t Aceleración lineal
- α_t Aceleración angular

El usuario puede definir la velocidad lineal máxima que el personaje puede alcanzar. El control de locomoción automáticamente ajusta los valores de la aceleración tanto lineal como angular para alcanzar la dirección y velocidad deseada. Estos elementos controlan las cuatro acciones básicas del controlador de locomoción: acelerar, frenar, girar a la izquierda y girar a la derecha. Acelerar viene representado por el valor positivo de a_t y frenar por el negativo. Al igual que los valores de α_t los valores positivos corresponden a girar a la izquierda y los negativos a la derecha.

En cada fracción de tiempo, el controlador cinemático actualiza el estado del personaje utilizando las siguientes ecuaciones (2.1):

$$\begin{aligned} x_{t+\Delta t} &= x_t + (\upsilon_t \cos\theta_t)\Delta t \\ z_{t+\Delta t} &= z_t + (\upsilon_t \sin\theta_t)\Delta t \\ \theta_{t+\Delta t} &= \theta_t + \omega_t \Delta t \end{aligned} \qquad (2.1)$$

$$\upsilon_{t+\Delta t} = \upsilon_t + a_t \Delta t$$
$$\omega_{t+\Delta t} = \omega_t + \alpha_t \Delta t$$

El proceso de simulación procede de esta manera continuamente. Mientras el valor del incremento de tiempo (Δt) sea razonable, el movimiento será suave y continuo. En nuestro caso elegimos 1/25 de segundo dado que cada segundo cuenta con 25 fotogramas.

Para la actualización de los parámetros del controlador de locomoción, se hacen las siguientes consideraciones particulares aplicando el siguiente conjunto de ecuaciones (2.2):

- La posición objetivo (x_t', z_t') corresponde a la siguiente en el camino P.
- La dirección objetivo está determinada por la ecuación:
$$\theta_t' = atan2(x_t' - x_t, z_t' - z_t) \qquad (2.2)$$
- La velocidad objetivo (υ_t') está determinada por el usuario.
- La aceleración lineal objetivo está determinada por la siguiente ecuación:
$$a_t = k_\upsilon (\upsilon_t' - \upsilon_t)$$
- La aceleración angular objetivo está determinada por la siguiente ecuación:
$$\alpha_t = k_\omega (\omega_t' - \omega_t)$$

Después de que los controles se calculan, los valores son actualizados en el siguiente lapso de tiempo (Δt). El siguiente estado representa la nueva posición y dirección del disco del personaje virtual. Para animar las articulaciones, el controlador cinemático toma como parámetro de entrada la velocidad υ_t actual.

2.6 El controlador cinemático del personaje

La naturalidad en el movimiento del personaje no esta determinada únicamente por su locomoción sino también por la cinemática del cuerpo. Por esta razón nos hemos dado a la tarea de especificar un controlador cinemático que dote de realismo al personaje. Para cumplir con este objetivo, nos hemos basado en algunos aspectos del controlador presentado por Pettre y Laumond en [PL05], donde se relaciona la velocidad lineal y angular del personaje virtual con la cinemática de éste.

El controlador de nuestra solución requiere de tres capturas de movimiento: la posición fija, el movimiento de caminar y el movimiento de correr (con esta propuesta es posible usar tres tipos de movimiento cualesquiera en el entendido de que debe existir una correspondencia). Estas animaciones pueden ser obtenidas de dos maneras: mediante la manipulación de los ángulos de cada articulación del esqueleto virtual en el editor de posturas o a través de un archivo de captura de movimientos (.amc).

La tarea de diseñar un movimiento de varios segundos a través del editor de posturas para obtener cada fotograma resulta tediosa debido a la gran cantidad de articulaciones con las que cuenta el modelo. Sin embargo se ha pensado en un conjunto de herramientas que facilita la síntesis de capturas de movimiento mediante interpolación lineal.

Esta metodología se describe de la siguiente manera.

Sea Post la postura del personaje, dicha postura tiene N valores Rot correspondientes a los ángulos de rotación de los N huesos del modelo.

Interpolar ($Post_{inicial}$,$Post_{final,}$ t)

```
1    Post ←Nueva_Postura;
2    para i=0 hasta N hacer
3        Post.Rot(i) = (1-t)*Post_inicial.Rot (i)+ Post_final.Rot (i);
4    devuelve Post;
```

Sea $Post_{inicial}$ la postura inicial y sea $Post_{final}$ la configuración final del personaje virtual. El método devuelve una postura con valores intermedios de rotación Rot entre la postura inicial y final, depende del valor de *t* que puede variar entre [0,1] en valor real. Para obtener una postura intermedia central entre la postura inicial y final, el valor de *t* debe ser de 0.5.

Para incrementar el realismo de la cinemática del personaje, se destinó una parte del controlador para manejar la velocidad lineal y otra para la velocidad angular.

El modulo que toma la velocidad lineal como entrada, establece un conjunto de rangos que van desde 0 hasta los 7 m/s. De acuerdo a la velocidad a la que se desplace el personaje será la captura de movimiento que se mostrará. Para cada captura de movimiento se asigna también una subdivisión de rangos, de manera que se muestre la misma captura de diferente manera a medida que se incrementa la velocidad. El controlador se encarga de realizar interpolaciones automáticas cuando se cambia la captura de movimiento mostrada.

El módulo que toma la velocidad angular como entrada, controla directamente las articulaciones superiores de las piernas del personaje. Dependiendo del valor de las rotaciones que se llevaran a cabo se decide entre actuar en una u otra pierna. El valor del ángulo se incrementará ligeramente en el eje Y en base a un índice calculado en conjunción al valor absoluto de la velocidad angular.

Finalmente, cabe destacar la participación en conjunto con el controlador de locomoción en el aspecto reactivo del personaje (Warping). Ya que el controlador de locomoción cuenta con los sensores radiales, es éste quien establece la comunicación con el controlador de cinemática cuando un obstáculo superior imprevisto es detectado.

Figura 2. 7: Comportamiento de Warping

Mientras el sensor superior detecte algún obstáculo, los ángulos superiores e inferiores de la espalda se incrementaran hasta llegar a un valor objetivo (Figura 2.7). Si el sensor superior no detecta colisión alguna, los valores de rotación de los huesos de la espalda regresarán a su valor original.

Capitulo 3. Planificación de la trayectoria

3.1 Introducción

Para resolver el problema de la planificación de la trayectoria, se necesitan como entradas la posición inicial y final del personaje y la definición tridimensional del ambiente. El módulo se limita a obtener la planificación en un ambiente plano. La salida del planificador viene representada por un conjunto de puntos que el personaje tendrá que recorrer que llevan de la posición inicial a la final sin colisionar con obstáculos describiendo un conjunto de curvas que dotan de naturalidad al movimiento. Hasta este punto del proceso de animación, esta aun no esta completa, sin embargo para realizar el muestreo se utiliza la forma semi – envolvente. El proceso de planificación garantiza el paso de los cuerpos activos sin colisión una vez que se calcula la animación. Dado que el método no se enfoca en las articulaciones, solamente se calcula la evolución de los parámetros de posición x, z. El resultado final de éste módulo es un conjunto de puntos $P_i(t_i) = (x_i, z_i)$ que el personaje deberá recorrer durante la animación.

El proceso de planificación se divide en tres partes. La primera corresponde al elemento que conecta la posición inicial a la final. El siguiente muestra la manera en que se optimiza el camino encontrado y finalmente se describe la manera en que se ajusta la trayectoria a un conjunto de curvas de Bézier para incrementar la naturalidad del camino.

3.2 Planificación basada en árboles aleatorios de exploración rápida

El planificador adoptado para nuestra solución esta basado en la metodología propuesta por Kuffner en [Kuf99]. Describiremos el funcionamiento de los algoritmos y tipos de datos usados para llevar a cabo la tarea. Se realizaron algunos ajustes a los procesos para obtener los resultados deseados.

Steven M. LaValle introdujo el concepto de árboles aleatorios de exploración rápida (RRTs) en [LaV99], un esquema de muestreo aleatorio originalmente diseñado para planificación de movimiento no holonómica. Los RRT también han sido aplicados en problemas de planificación cinemática en espacios de configuración de hasta 12 dimensiones [LK99]. El método muestra muchas propiedades favorables tanto en problemas holonómicos como no holonómicos. Al igual que en el planificador de [HLM97], el objetivo es construir progresivamente un árbol de configuraciones de manera que la expansión del árbol este altamente guiada hacia regiones inexploradas del espacio.

El algoritmo utilizado aquí se denomina planificador RRT Connect y se define como un método de planificación basado en dos árboles aleatorios de exploración rápida balanceados, que se generan utilizando un detector de colisiones para alcanzar una rápida convergencia [KL00].

Este planificador fue seleccionado con el objetivo de obtener un rendimiento semejante a la complejidad del problema, de manera que si la solución del problema es muy sencilla, el algoritmo encuentre la solución rápidamente. Sin embargo si el problema es complejo (por ejemplo que involucre caminos estrechos), el planificador debe ser capaz de resolverlo a través de mayor tiempo de cálculo si es requerido.

3.2.1 El planificador RRT Connect

Al planificador se le proporciona $q_{inicial}$ como configuración de inicio del personaje y q_{final} como configuración objetivo dentro del espacio de configuraciones C (en este caso el plano X, Z del entorno tridimensional). La tarea del planificador es encontrar un camino libre de colisiones que conecte $q_{inicial}$ y q_{final} en C_{libre} (sea C_{libre} el subconjunto de configuraciones de C libres de colisión). Para su funcionamiento, el algoritmo requiere de la especificación de una función que devuelva una distancia entre dos configuraciones de C (métrica). También se emplea un método que realice un muestreo aleatorio uniforme y una función (detector de colisiones) para discriminar configuraciones no deseadas (en colisión). Para llevar a cabo la planificación, se emplean dos estructuras de datos de tipo árbol.

RRT – Connect ($q_{inicial}$, q_{final})

```
1   Ta ← inicializa (qinicial);
2   Tb ← inicializa (qfinal);
3   para i =1 hasta K hacer
4       qaleatorio ← Configuración_Aleatoria ();
5       qcercano ← Cercano (Ta, qaleatorio);
6       si Arista (qcercano, qaleatorio) ∈ Clibre entonces
7               Agrega_Rama (Ta, qcercano, qaleatorio);
8               q'cercano ← Cercano(Tb, qaleatorio);
9               si Arista (q'cercano, qaleatorio) ∈ Clibre entonces
10                      Agrega_Rama(Tb, q'cercano, qaleatorio);
11                      devuelve SOLUCION;
12      si |Tb|>|Ta| entonces Intercambia (Ta, Tb);
13  devuelve FALLO;
```

El T_a (árbol a) tiene $q_{inicial}$ como nodo raíz, T_b (árbol b) tiene q_{final} como nodo raíz. K es un número establecido por el usuario que indica el número de veces que se intentara la conexión. La configuración $q_{aleatorio}$ representa una posición aleatoria muestreada por el método Configuración_Aleatoria. La configuración $q_{cercano}$ representa el nodo del árbol T_a más cercano a $q_{aleatorio}$ y es determinado por el proceso Cercano. El método Arista devuelve una línea recta entre sus parámetros. El método Agrega_Rama tiene la capacidad de asignar un nodo hijo a un elemento de

un árbol determinado. La configuración $q'_{cercano}$ corresponde al nodo del árbol T_b más cercano a $q_{aleatorio}$. El método Intercambia, intercambia los contenidos de los árboles proporcionados, dicha función se utiliza para balancear el crecimiento de los árboles. En caso de encontrar un conjunto de configuraciones que vayan de $q_{inicial}$ a q_{final}, el algoritmo devolverá SOLUCION en caso contrario, devolverá FALLO.

En la figura 3.1 se ejemplifica el funcionamiento del planificador RRT Connect en un escenario tridimensional donde ambos cilindros representan la posición inicial y final del personaje.

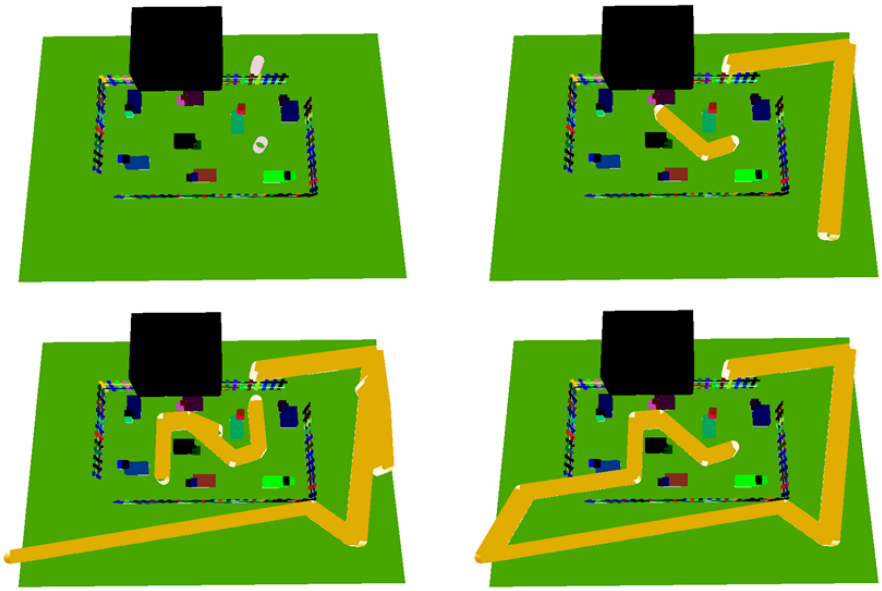

Figura 3.1: Comportamiento del planificador RRT Connect

3.3 Alisado del camino planificado

Los caminos generados por los planificadores de muestreo aleatorio típicamente contienen segmentos de trayectoria excedentes (en ocasiones ciclos). Dichos caminos no pueden ser utilizados de manera inmediata y requieren una optimización local o alisado.

Para obtener un camino óptimo, algunos autores han optado por planificar un camino libre de colisiones y posteriormente deformarlo de manera iterativa. En [Hsu00] se busca un camino en diferentes niveles de resolución, y a cada sección se le sustituye por un camino más corto de manera iterativa. Esta solución fue la adoptada para llevar a cabo nuestro alisado.

El proceso de alisado, esta basado en dos métodos, la subdivisión y la optimización. La combinación de ambas técnicas de manera iterativa nos devuelve un camino más corto y con una menor cantidad de puntos dentro de la trayectoria.

3.3.1 Subdivisión del camino planificado

Para lograr un mejor alisado, es necesario subdividir los segmentos de la trayectoria sin modificar ésta. Entre mayor sea el número de segmentos de la trayectoria, mejor funcionará el alisado y se alcanzará una trayectoria más corta. Nuestro enfoque se basa en un método que divide cada segmento del camino en dos partes. Después del concluir el proceso de subdivisión se continúa con la optimización.

3.3.2 Optimización del camino planificado

El algoritmo de optimización es de tipo recursivo, y tiene como objetivo reducir la trayectoria que tendrá que recorrer nuestro personaje. Posee como parámetro de entrada un camino generado por un planificador aleatorio, y está representado por una lista ordenada de puntos (x, z) que el personaje virtual tendrá que recorrer. El primer elemento de la lista corresponde a la posición inicial del personaje y el último es la posición final. Este proceso devuelve otro camino de estructura similar, con una cantidad menor o igual de puntos objetivo. El método de optimización utiliza algunos métodos auxiliares utilizados también por el planificador como son Arista y el detector de colisiones.

Optimizar (P)

```
1   si P ≠ Ø entonces
2       Agrega (P^Optimo, P_0);
3       para i = 0 hasta |P|-1 hacer
4           si Arista (P_0, P_|P|-i) ∈ C_libre entonces
5               P^Resto ← Rango (P_|P|-i, P_|P|);
6               Optimizar (P^Resto);
```

P es el camino que nos proporcionó el planificador, P^{Optimo} es la salida del método que es actualizado por referencia. Al finalizar el método, P^{Optimo} tiene menor o igual numero de puntos que P. P_0 es el primer elemento del camino P. El proceso Agrega, añade un elemento a un camino. Arista es el método que genera una arista entre dos elementos del camino (puntos x, z). C_{Libre} representa las configuraciones válidas (puntos del espacio libres de colisión). Rango es un método que devuelve un rango de elementos de un camino. P_{Resto} son los elementos de P que aun no han sido optimizados.

3.3.3 Optimización bi-direccional del camino planificado

Para reducir el camino a su expresión más corta, se planteo la aplicación de los métodos en ambas direcciones, de manera que el algoritmo de alisado quedo configurado de la siguiente manera.

Alisado (P, $P^{Alisado}$)

```
1    para i = 0 hasta K hacer
2        Subdividir (P);
3    Optimizar (P, P^Optimo);
4    Invertir (P^Optimo);
5    para i = 0 hasta K hacer
6        Subdividir (P^Optimo);
7    Optimizar (P^Optimo, P^Alisado);
8    Invertir (P^Alisado);
```

El método de alisado nos devuelve, $P^{Alisado}$ que es el camino optimizado en ambas direcciones. Se usa la directiva Invertir para realizar el recorrido en ambas direcciones. Finalmente se obtiene un camino considerablemente mejor para ser recorrido como se muestra en la figura 3.2.

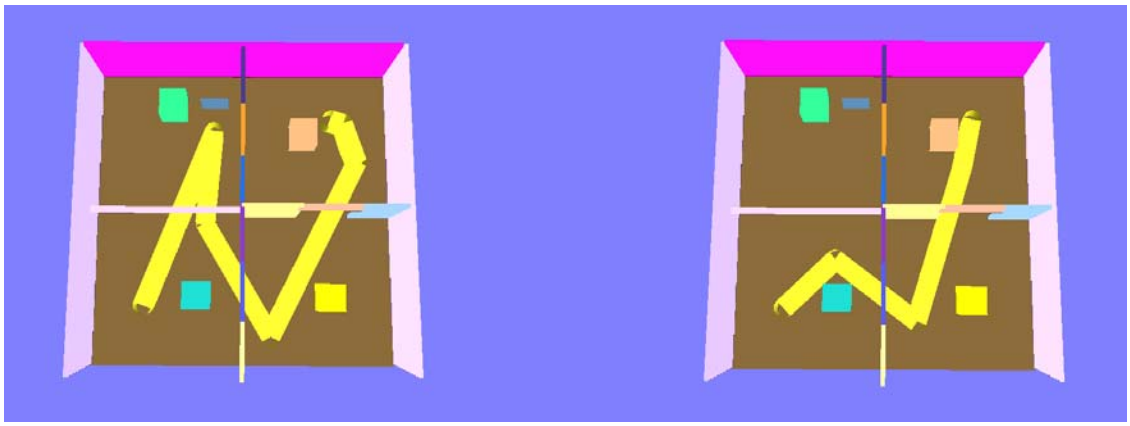

Figura 3.2: El camino planificado (izquierda) y el alisado (derecha).

3.4 Aproximación del camino global con curvas de Bézier

Hemos escogido modelar el camino global por medio de un conjunto de curvas de Bézier de tercer grado. Las entradas de cálculo de cada segmento son dos posiciones (x, z) del personaje y el punto de control de la curva anterior. Es conveniente encontrar un camino que una estas dos posiciones. Las curvas de Bézier nos permiten garantizar que las extremidades del camino consideren progresivamente las direcciones θ (más adelante se presentan algunas propiedades de las curvas de Bézier) que posteriormente adoptará el personaje. En general, consideramos que un personaje se desplaza entre dos puntos de la manera siguiente: se dirige globalmente en línea recta de la posición de partida a la de llegada, por otro parte se reorienta a las extremidades de su trayecto. En el caso de la navegación en presencia de obstáculos, los cambios de orientación intervienen en proximidad de los obstáculos dado que el personaje los rodea. Las propiedades del planificador y de las curvas de Bézier permiten simular correctamente este comportamiento. En la parte que corresponde a los resultados experimentales se mostrará la importancia de este nuevo método. Las

curvas de Bézier permiten producir caminos alisados, es decir diferenciables en todo punto. Estas son analíticamente descritas por una ecuación parametrizada (ecuación 3.1), donde las variables son las coordenadas (x, z) de cuatro puntos de control (es decir, una curva de Bézier de grado N se construye con N+1puntos de control).

$$B(u) = \Sigma_{k=0}^{3} \frac{P_k 3!}{K!(3-k)!u^k(1-u)^k} \tag{3.1}$$

Donde $0 \leq u \cdot \leq 1$ es el parámetro de la curva, y P_k la posición del k-ésimo punto de control.

Consideremos el cálculo de un camino local del punto $A = (x_A; z_A)$ hacia $B = (x_B; z_B)$. El posicionamiento de los cuatro puntos de control (denotados como P_0, P_1, P_2, P_3 respectivamente definidos por (x_0, z_0), (x_1; z_1), (x_2; z_2), (x_3; z_3)) se calcula como sigue:

- P_0 coincide con A: $(x_0; z_0) = (x_A; z_A)$,
- P_1 se obtiene por traslación del punto A, según la orientación θ_A sobre una distancia D: $(x_1; z_1) = (x_A + D\cos(\theta_A), z_A + D\sin(\theta_A))$. Empíricamente, se escoge D de acuerdo a la altura del personaje.
- P_2 se obtiene por traslación del punto B siguiendo $\theta_B+\pi$ sobre la distancia D. $(x_2; z_2) = (x_B+D\cos(\theta_B+\pi), z_B+D\sin(\theta_B+\pi))$.
- P_3 coincide con B: $(x_3; z_3) = (x_B; z_B)$,
- Caso particular: si $\| AB \| < 2D$ entonces $D = \| AB \|/2$.

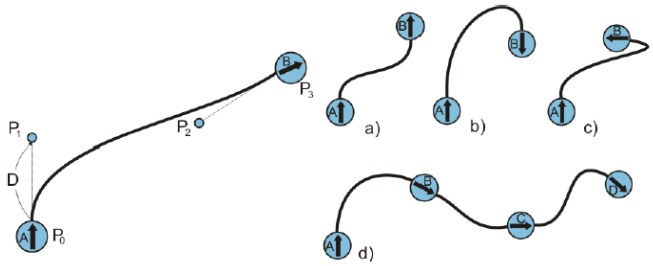

Figura 3.3: Diferentes curvas de Bézier y su composición.

A la izquierda de la figura 3.3, se muestra un ejemplo de colocación de los puntos de control de la curva de Bézier, así como la curva resultante. Dicha curva posee ciertas propiedades remarcables:

- La curva pasa por sus puntos de control extremos. Es decir por los puntos P_0 y P_3 (o A y B). Los dos puntos de control intermedios se denominan externos a la curva.

- La curva queda contenida en la envolvente convexa de los puntos que la controlan.

- En sus extremidades, la curva es tangente a los segmentos definidos por los dos primeros y los dos últimos puntos de control (respectivamente en $[P_0 P_1]$ y en $[P_2 P_3]$).

- La curva y su derivada son continuas.

Recordemos que el camino solución es una composición de curvas de Bézier. Aquí las propiedades de tangencia de las curvas de Bézier toman un mayor interés. Se puede observar una composición de curvas en la parte inferior derecha de la figura 3.3. Una composición así construida de curvas de Bézier es por definición una B-Spline. Las propiedades de las curvas de Bézier permiten asegurar una continuidad de clase $C1$ de la B-Spline.

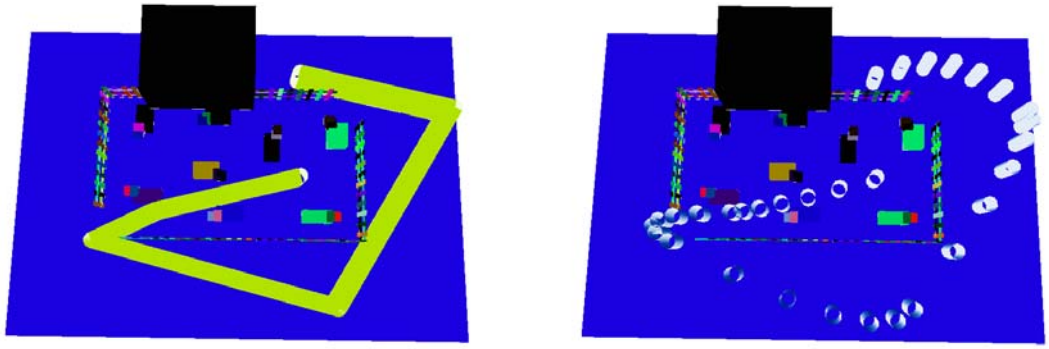

Figura 3.4: Camino generado con curvas de Bézier

Dado que las curvas de Bézier generan un desfase con el camino rectilíneo, se tiene que recurrir al detector de colisiones. Los puntos que coinciden con un obstáculo son removidos de la trayectoria (Figura 3.4).

Capítulo 4. Resultados

El software desarrollado tiene como nombre MOVE - BUAP # y es el resultado de un conjunto de investigaciones previas que conciernen sus diferentes capacidades. Fue diseñado e implementado en el laboratorio de movilidad y visión de la facultad de ciencias de la computación de la Benemérita Universidad Autónoma de Puebla (FCC - BUAP). Se investigó de manera separada la integración de objetos tridimensionales diseñados en programas independientes dentro de aplicaciones de escritorio, el uso de motores de física tanto para simulación como para detección de colisiones, además de la interacción con esqueletos y movimientos capturados.

4.1 El entorno de trabajo

El software se desarrollo en un principio en Microsoft Visual Studio 2005, y se finalizó en su edición 2008. El lenguaje de programación empleado fue la especificación de C# usando la familia de lenguajes .NET 2.0. Se optó por C# debido a su gran capacidad en el manejo de objetos abstractos.

La solución cuenta con dos motores externos que proveen la funcionalidad de renderizado y manejo de física (detección de colisiones).

El renderizador utilizado es OpenGL (Open Graphic Library) en su versión contenida en el Tao Framework. Dicho entorno de trabajo es una compilación de motores tanto gráficos, como de audio y simulación especializados en el desarrollo de videojuegos y soluciones multimedia utilizando la familia de lenguajes de programación .NET.

Como motor de física se utilizó ODE (Open Dynamics Engine) utilizando un adaptador para .NET llamado ODE.NET. Éste entorno de trabajo provee todas las herramientas necesarias para la simulación física de nuestra aplicación. Principalmente fue utilizado su detector de colisiones dado que es muy preciso.

El sistema fue desarrollado en una PC con procesador de dos núcleos AMD de 64 bits, corriendo a 1.70 GHz y 1 GByte de RAM. Las pruebas fueron realizadas bajo las mismas condiciones.

El sistema fue ideado en diferentes compilados (.dll) para incrementar la velocidad de compilación. En un principio se diseño una aplicación pequeña para cada uno de los módulos que se emplearon y finalmente se prosiguió a la unión de ellos para la generación de MOVE – BUAP # como se detalla en la siguiente sección.

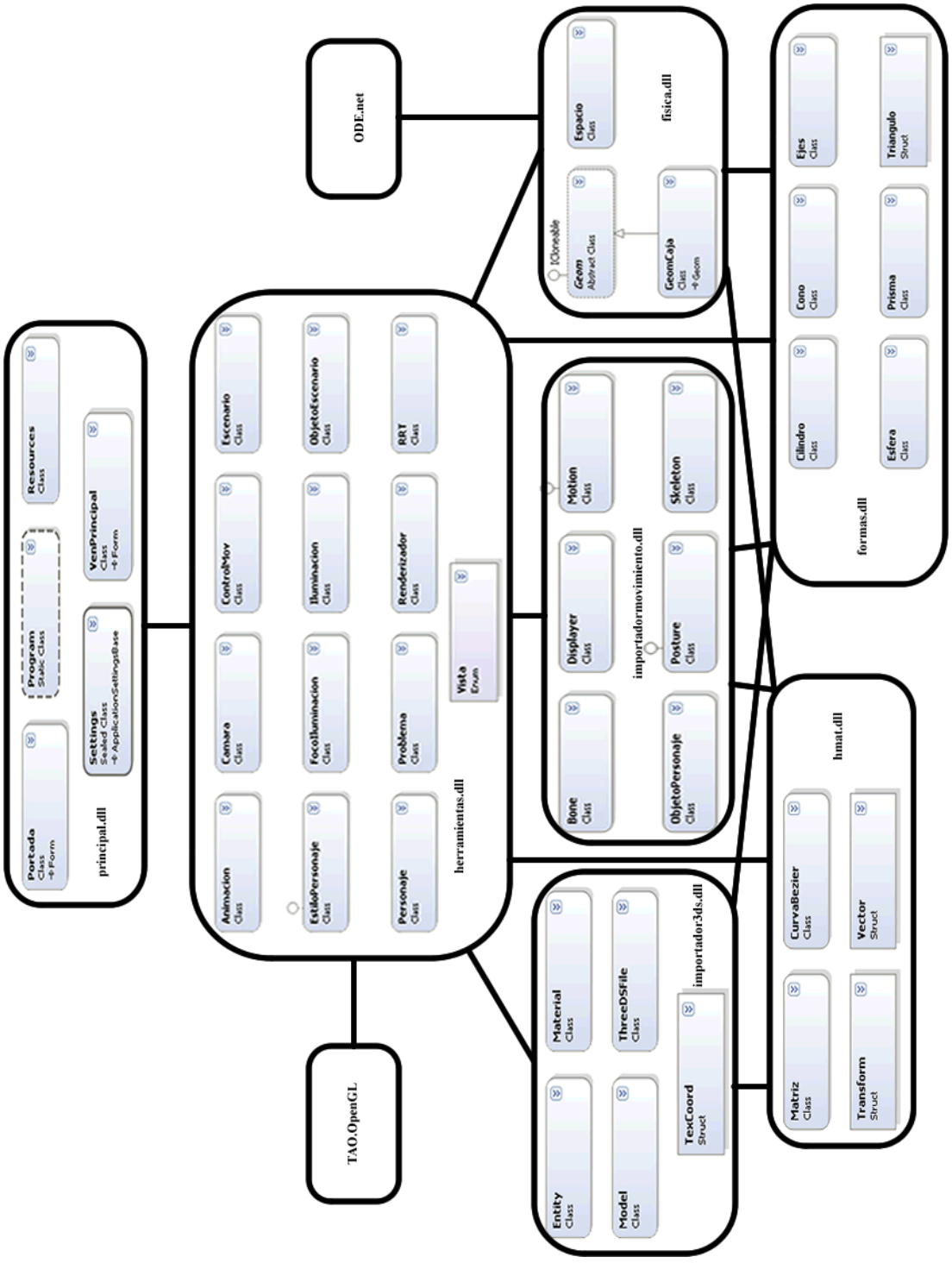

Figura 4. 1: Compilados del sistema MOVE - BUAP #

4.1.1 Compilado principal

Tal como se detalla en la figura 4.1 este es el módulo que se encuentra a más alto nivel del programa, y en este se diseñó la interfaz gráfica. La interfaz gráfica cuenta con nueve apartados que sirven para el manejo integral de MOVE - BUAP #. El

sistema está basado en la existencia de proyectos, que pueden contener toda la información necesaria para reproducir, manipular y generar nuevas animaciones. La tecnología utilizada para el alojamiento de la información en archivos binarios se denomina serialización dentro de la tecnología de .NET (C#). La interfaz gráfica fue planteada de la manera más intuitiva para facilitar su uso por parte del usuario (figura 4.2).

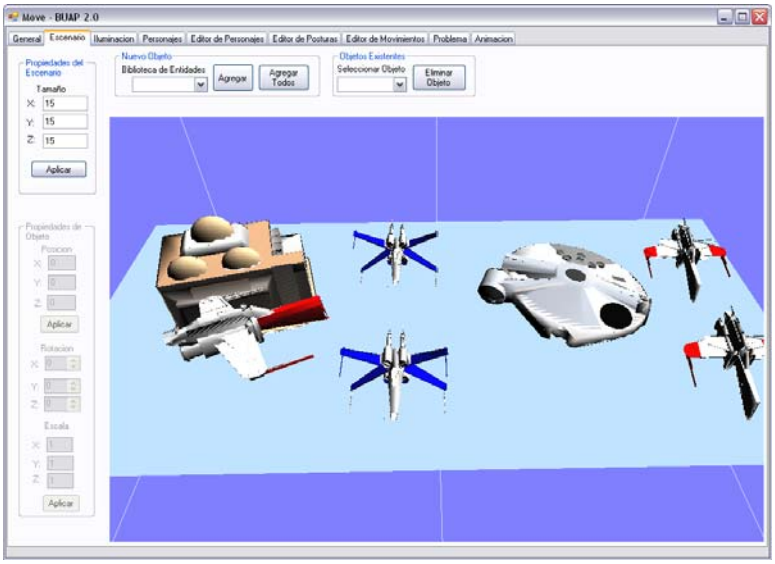

Figura 4. 2: Interfaz gráfica de usuario de MOVE - BUAP #

La serialización permite el alojamiento de objetos lógicos dentro de un flujo de datos conservando los apuntadores que éstos poseen. Por lo tanto la cantidad de información alojada en los archivos binarios se optimiza. A través del menú de pestañas superior, se puede controlar cada aspecto referente a la animación que será diseñada como se puede observar en la figura 4.3.

Figura 4. 3: Menú de control de MOVE - BUAP #

El funcionamiento de cada pestaña se describe brevemente como sigue:

- General
  En el se pueden crear, cargar o guardar los proyectos de MOVE – BUAP #, sirve para alojar las herramientas de importación y manejo de entidades tridimensionales (archivos .3ds) para el escenario o para las vestimentas de los personajes (figura 4.4).

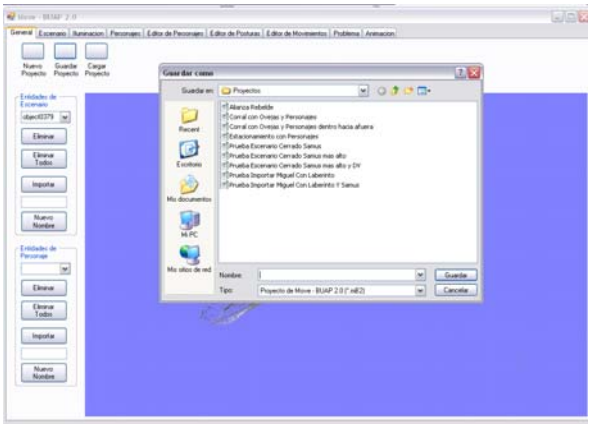

Figura 4. 4: Pestaña "General" de MOVE - BUAP #

- Escenario
  En el se establece el tamaño del escenario y cuenta con un conjunto de controles especializados para el diseño de éste. Cada entidad del escenario tiene la capacidad de ser trasladado, rotado y escalado a voluntad del usuario (figura 4.5).

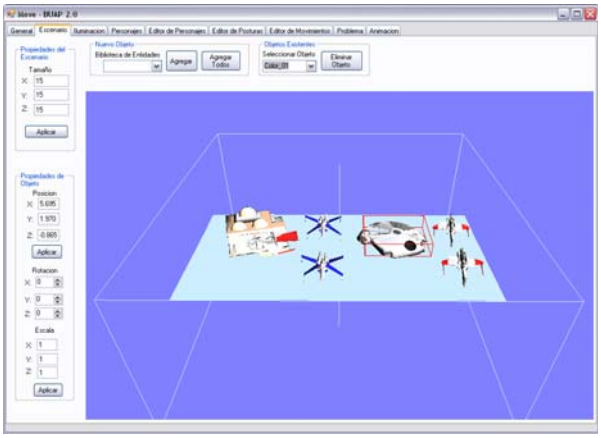

Figura 4. 5: Pestaña "Escenario" de MOVE - BUAP #

- Iluminación
  En esta pestaña se controla todo lo referente a la iluminación de la escena, se pueden especificar hasta ocho puntos de iluminación (restricción de OpenGL). A cada punto se puede asignar una posición y valores tanto de luz especular como ambiente y difusa (figura 4.6).

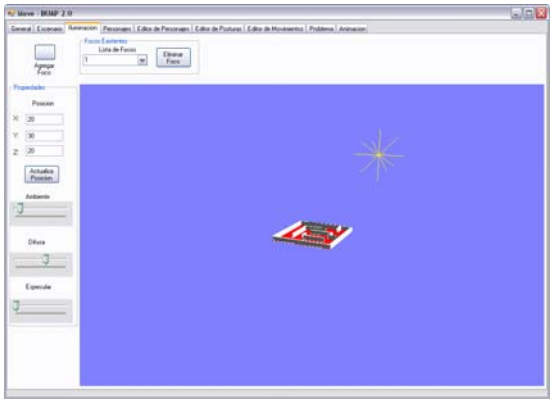

Figura 4. 6: Pestaña "Iluminación" de MOVE - BUAP #

- Personajes
  En este apartado, el usuario puede determinar cuantos personajes quiere mostrar en la animación y dotarles de sus características como son: posición inicial y final, altura, radio aproximado y estilo (vestimenta y cinemática). La posición inicial del personaje se muestra como un cilindro plano y la final como un cilindro de alambre (figura 4.7).

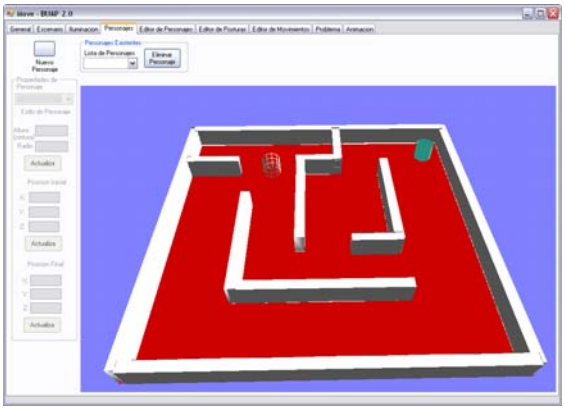

Figura 4. 7: Pestaña "Personajes" de MOVE - BUAP #

- Editor de personajes
  Este módulo se encarga del manejo de los estilos de personaje que serán usados en nuestra animación. A cada estilo se le asigna un esqueleto cinemático mediante un importador (archivos .asf). Dicho estilo se caracteriza por poseer un conjunto de vestimentas específicas para cada miembro del esqueleto. Cada entidad tridimensional que forma parte de la vestimenta puede ser trasladada, rotada y escalada a voluntad del usuario (figura 4.8).

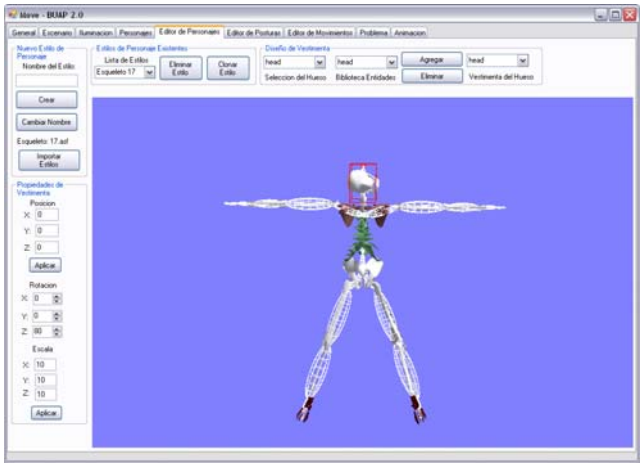

Figura 4. 8: Pestaña "Editor de personajes" de MOVE - BUAP #

- Editor de posturas
  El usuario puede generar y alojar una cantidad ilimitada de posturas. Éstas son creadas al manipular los valores de rotación de cualquier articulación estilo de personaje deseado. Dichas posturas pueden ser transmitidas al editor de movimientos para su empleo futuro (figura 4.9).

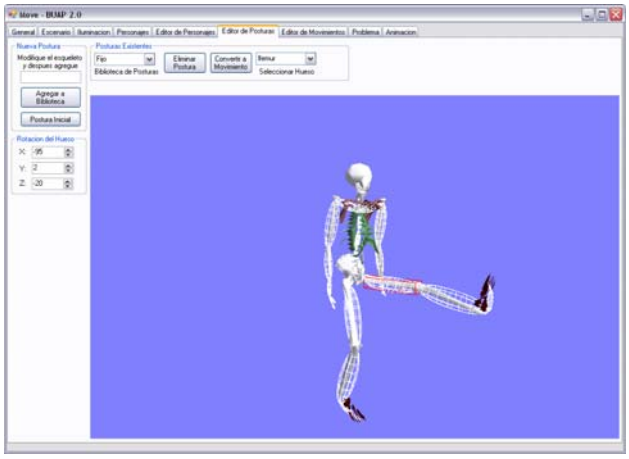

Figura 4. 9: Pestaña "Editor de posturas" de MOVE - BUAP #

- Editor de movimientos
  Particularmente esta es una de las secciones más interesantes de la solución. El usuario tiene las herramientas necesarias para importar capturas de movimiento alojadas en archivos (.amc). Estas capturas son agregadas a una biblioteca. Además pueden generarse movimientos nuevos a partir de las posturas existentes utilizando herramientas de extrapolación, copia y corte. El software cuenta también con un conjunto de controles para hacer variar la velocidad de los movimientos. En esta pestaña se debe especificar la posición fija, y los movimientos de caminar y correr que el personaje utilizará para expresar la animación final. Cada recuadro de la animación puede ser

apuntado mediante una barra de desplazamiento, que además sirve para seleccionar las posturas que serán editadas (figura 4.10).

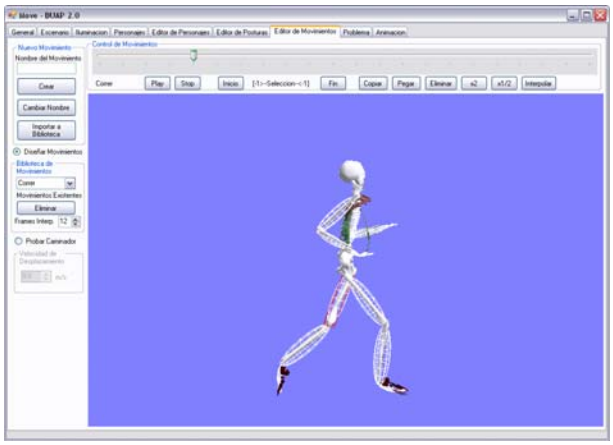

Figura 4. 10: Pestaña "Editor de movimientos" de MOVE - BUAP #

- Problema
  Esta pestaña permite al usuario activar el mecanismo de planificación. Debe especificarse el número de nodos que serán tomados como muestra usando el planificador RRT. Posteriormente, puede activarse el alisado y la expresión de la trayectoria como un conjunto de curvas de Bézier, para incrementar la naturalidad del movimiento (figura 4.11).

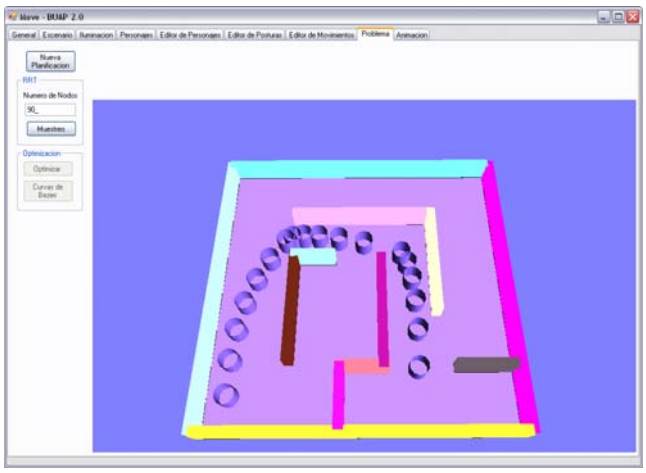

Figura 4. 11: Pestaña "Problema" de MOVE - BUAP #

- Animación
  Este marco muestra la animación final, en esta sección se puede activar el comportamiento reactivo (warping) del personaje y la forma de visualización. Es necesario especificar la velocidad objetivo que el personaje deberá alcanzar en el recorrido de la trayectoria (figura 4.12).

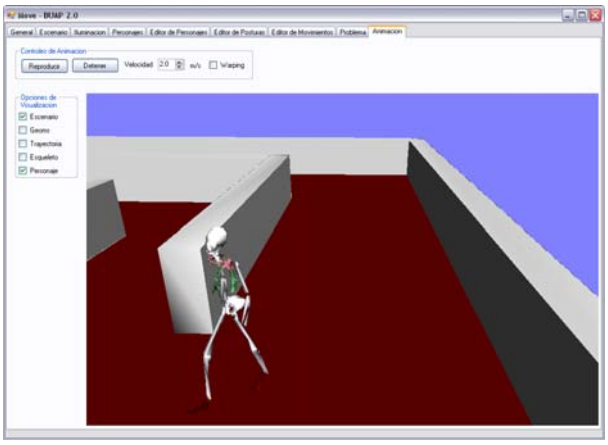

Figura 4. 12: Pestaña "Animación" de MOVE - BUAP #

4.1.2 Compilado herramientas

El módulo herramientas se encarga de establecer un vínculo entre los motores de bajo nivel (OpenGL, ODE, herramientas matemáticas, importadores de geometrías tridimensionales y capturas de movimiento) y la interfaz gráfica de usuario. Se encarga de alojar los ajustes que se han realizado en la interfaz gráfica y gestionar la automatización de las tareas cíclicas (ajuste automático del tamaño del escenario, cálculo de tamaño de entidades tridimensionales, etc.). Además organiza los elementos propios del funcionamiento de MOVE – BUAP # como las bibliotecas de entidades de escenario y vestimenta, estilos de personaje, etc. En este módulo se realiza la planificación (RRT) y optimización de las trayectorias a nivel lógico. Además contiene tanto el controlador de locomoción como el cinemático que serán utilizados para especificar la animación. Una de las tareas más importantes del compilado "herramientas" es la coordinación del renderizado del controlador de OpenGL de acuerdo con los valores actuales alojados en las clases concernientes a la visualización (cámara, iluminación, etc.).

4.1.3 Compilado importador3DS

Este compilado contiene las clases necesarias para la integración de objetos tridimensionales especificados en archivos de tipo 3DS. Cuenta con la capacidad de traducir el contenido de los archivos en colecciones de triángulos llamados entidades y modelos. Efectúa el cálculo de las normales de manera automática y mapea las texturas respecto a las posiciones relativas de las caras. Las texturas quedan alojadas en sus tres valores de respuesta a la luz: ambiente, especular y difusa. Para trazar los dibujos recurre al compilado "formas" que contiene la especificación de la estructura triangulo.

4.1.4 Compilado importadormovimiento

Este conjunto de clases esta provisto de las funciones necesarias para traducir archivos de esqueleto cinemático (.asf) y captura de movimientos (.amc) a estructuras de datos conocidas por MOVE - BUAP #. Cuenta además con un tipo de objeto capaz de reproducir las capturas de movimiento importadas. Se encarga de realizar las tareas de extrapolación y edición de posturas. Además provee las funcionalidades para generar nuevas animaciones y posturas a partir de la información recibida por el compilado "herramientas". Para su funcionamiento recurre al compilado "hmat", que es un conjunto de herramientas matemáticas genérico.

4.1.5 Compilado física

El módulo de física establece un vinculo entre el lenguaje de programación de ODE.NET y las entidades de MOVE - BUAP #. Se encarga de la automatización de tareas de difícil realización como la traducción de matrices de transformación en valores independientes (necesarios para el renderizado). El módulo de física coordina el funcionamiento del detector de colisiones. Asigna un comportamiento diferente a cada tipo de colisión detectada por este mecanismo. Contiene además un conjunto de directivas utilizadas para facilitar el manejo de las geometrías físicas (encontrar ángulos, buscar el más cercano, generar geometrías intermedias, etc.). Para su funcionamiento recurre a los compilados de "hmat" (herramientas matemáticas) y formas.

4.1.6 Compilado Hmat

Hmat es una manera abreviada de herramientas matemáticas, y es un compilado que contiene entidades matemáticas de uso frecuente. La gran mayoría de módulos de MOVE – BUAP # utilizan éste conjunto de funciones. En ella encontramos clases tan importantes para la graficación como son los vectores, matrices, transformaciones y curvas de Bézier (necesarios para el alisado final de la trayectoria planificada).

4.1.7 Compilado Formas

MOVE – BUAP # utiliza formas geométricas de uso frecuente como son los cilindros, prismas rectangulares, ejes, conos, esferas y los antes mencionados triángulos. Estas clases se encargan de alojar los valores principales de cada una de las figuras además de su posición y valores de rotación en los diferentes ejes. De esta manera se reduce la cantidad de memoria utilizada dado que se recurre a un solo modelo en vez de múltiples cada vez que se realiza el renderizado.

4.2 Pruebas y resultados

A continuación se describen tres pruebas de diferente dificultad realizadas en MOVE – BUAP # y se evalúan diferentes aspectos del funcionamiento de éste. Estos son algunos de los aspectos que se obtuvieron al realizar las evaluaciones.

Escenario	Nodos Explorados	Tiempo de planificación	Tiempo de optimización	Tiempo de Bézier	Tiempo total
Estacionamiento	37	230.33 ms	80.11 ms	30.04 ms	340.48 ms
Interior de una casa	44	20.02 ms	40.05 ms	12.01 ms	72.08 ms
Corral con ovejas	299	240.34 ms	270.38 ms	20.02 ms	530.74 ms

4.2.1 Prueba 1. El estacionamiento

El ambiente es de tamaño grande, su dificultad es baja dado que no cuenta con una gran cantidad de pasajes estrechos. Sin embargo existe una zona de obstáculos superior producida por el árbol que puede producir una modificación cinemática por warping. En este caso se está evaluando la eficiencia de los grados de libertad reactivos.

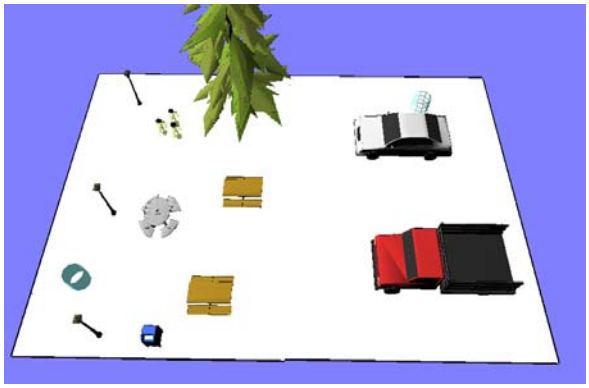

Figura 4. 13: Prueba 1. Posición inicial y final

Figura 4. 14: Prueba 1. Trayectoria planificada

Figura 4. 15: Prueba 1. Movimiento reactivo efectuado

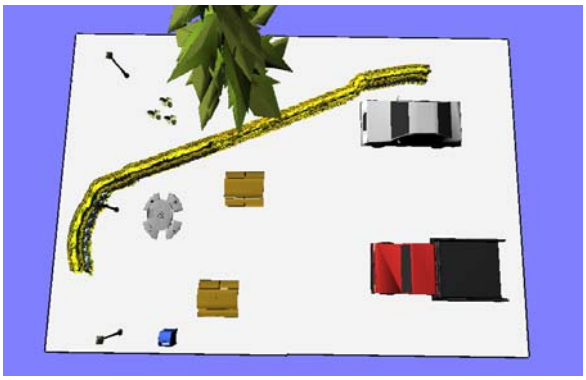

Figura 4. 16: Prueba 1. Animación generada

4.2.2 Prueba 2. El interior de una casa

El ambiente es de tamaño mediano, su dificultad se incrementa respecto al anterior dado que cuenta con pasajes más estrechos que el estacionamiento. Cuenta con una buena cantidad de pasajes en los que se activará el módulo reactivo. El personaje tiene múltiples posibilidades de alcanzar su objetivo mediante el paso de dichos

pasajes. Se evalúa la confiabilidad del conjunto formado entre grados activos y reactivos.

Figura 4. 17: Prueba 2. Posición inicial y final

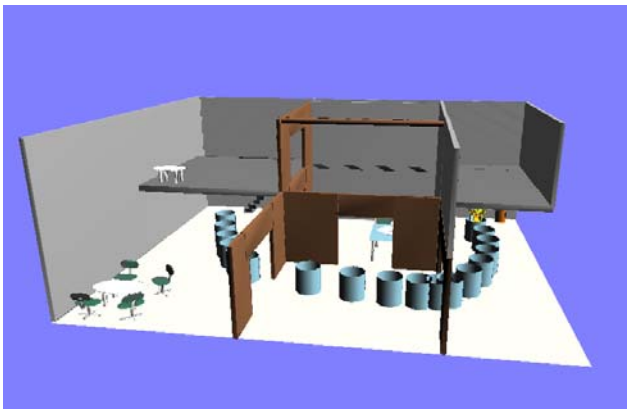

Figura 4. 18: Prueba 2. Trayectoria planificada

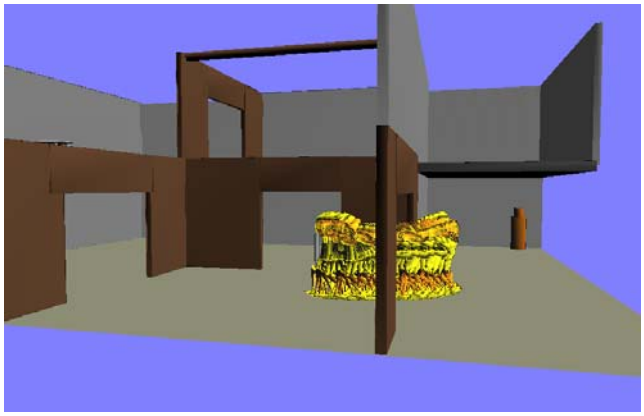

Figura 4. 19: Prueba 2. Movimiento reactivo efectuado

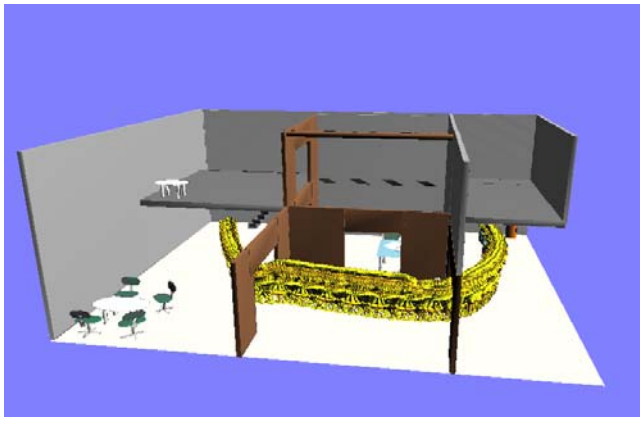

Figura 4. 20: Prueba 2. Animación generada

4.2.3 Prueba 3. El corral con ovejas

En este caso, nos encontramos con una prueba clásica en los métodos de planificación y animación de personajes virtuales. El corral con ovejas se mantiene como un experimento de buen grado de dificultad debido la cantidad de caminos estrechos no uniformes con los que cuenta. Además contiene un pasaje en el que se debe realizar warping bajo el árbol.

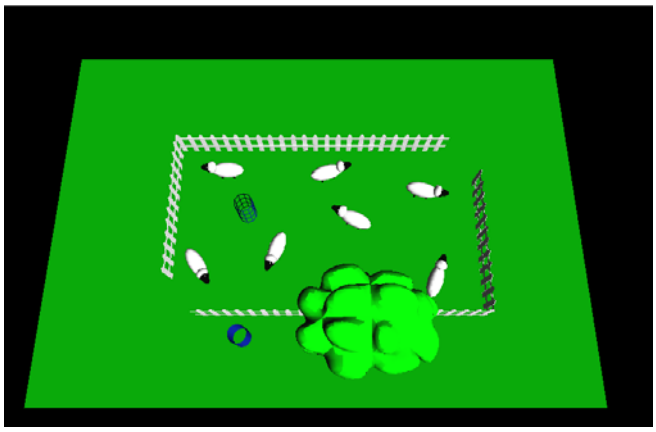

Figura 4. 21: Prueba 3. Posición inicial y final

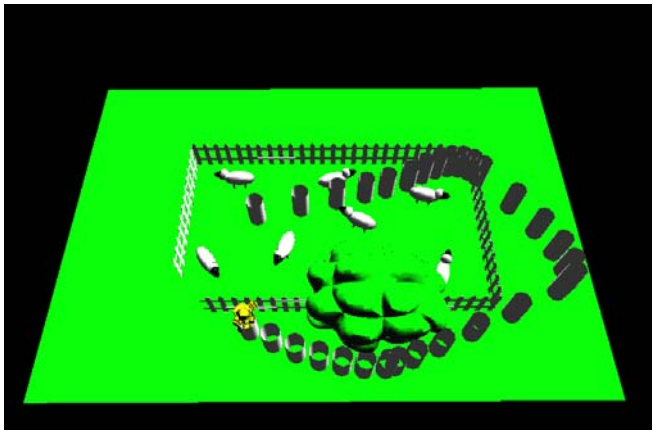

Figura 4. 22: Prueba 3. Trayectoria planificada

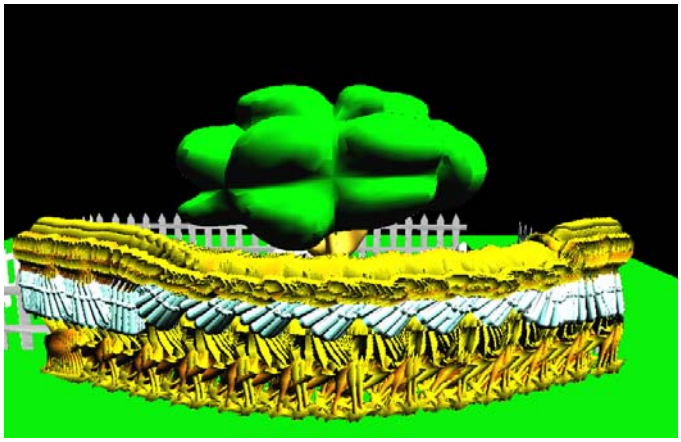

Figura 4. 23: Prueba 3. Movimiento reactivo efectuado

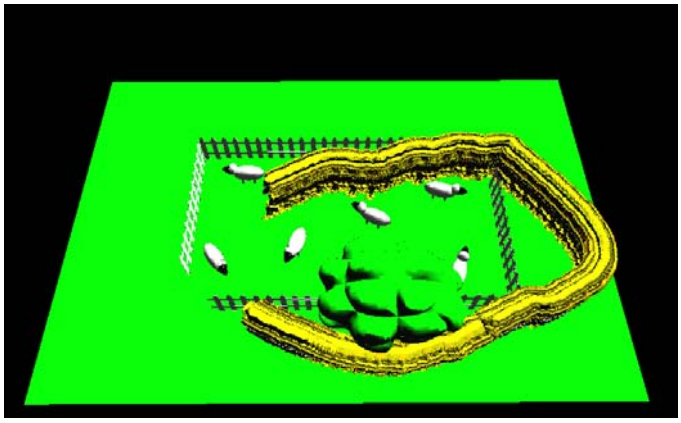

Figura 4. 24: Prueba 3. Animación generada

Capítulo 5. Conclusiones y trabajos futuros

5.1 Conclusiones

El ser humano por naturaleza tiene la capacidad de desplazarse en el espacio que le rodea. Esta habilidad incluye la evasión y reacción ante obstáculos imprevistos. Debido al gran incremento en la aparición de entidades humanoides en los videojuegos y en la industria cinematográfica. El principal objetivo perseguido en esta investigación fue el incremento en la naturalidad de los movimientos de los personajes virtuales.

La generación de animaciones que incluyen locomoción dentro de ambientes no estructurados requiere de la resolución de problemas de planificación con diversas restricciones. Así mismo se sintetizan trayectorias humanamente naturales mediante el alisado basado en curvas de Bézier.

Nuestra solución provee una herramienta sólida para el desarrollo de animaciones realistas en ambientes tridimensionales. Destaca la velocidad en el proceso de planificación de la trayectoria del actor virtual. Además, las posibilidades de visualización son ilimitadas gracias a los editores de vestimenta y ambiente construidos. La autonomía del personaje respecto a su entorno asegura el recorrido de la trayectoria a diferentes velocidades mostrando diferentes animaciones.

Los obstáculos imprevistos conciernen a los grados de libertad reactivos del modelo cinemático. La percepción de obstáculos por parte del personaje lleva a la activación de rutinas de evasión para incrementar el nivel de realismo visual en el comportamiento del mismo.

La demostración de las funcionalidades que ofrece nuestra aplicación se muestra mediante un conjunto de ejemplos. Hasta este punto se concluye que se logró el objetivo de crear una aplicación de esta naturaleza.

5.2 Trabajos futuros

En lo referente al futuro de la animación de personajes virtuales, la visión va dirigida hacia los comandos de alto nivel. Los proyectos futuros deben considerar seriamente la existencia de métodos que faciliten al usuario establecer comportamientos complejos de uno o múltiples personajes. Se debe además agregar la posibilidad de realizar tareas conjuntas con objetivos múltiples.

Para tales propósitos considero que se debe plantear un sistema de animación completamente basado en un motor de física. De manera que se incremente el realismo de la escena por parte de los elementos activos y reactivos del entorno y los personajes.

Finalmente se debería integrar el concepto de comportamiento basado en el entorno, con lo que las acciones de los personajes se verían condicionadas por el momento y el lugar en que se encuentren.

Bibliografía

[And90] T. L. Anderson, Autonomous robots and emergent behaviors: A set of primitive behaviors for mobile robot control. *IROS'90,* 1990.

[AT01] A. Aubel y D. Thalmann. Interactive modeling of the human musculature. *Proceedings of Computer Animation 2001*, 2001.

[Bai86] J. Baillieul. Avoiding obstacles and resolving kinematic redundancy. *Proceedings of 1986 International Conference on Robotics and Automation (ICRA'86),* 1986.

[BBET97] R. Boulic, P. Becheiraz, L. Emering y D. Thalmann. Integration of motion control techniques for virtual human and avatar real – time animation. *Proceedings of ACM International Symposium VRST'97,* 1997.

[BK00] R. Bohlin y L. Kavraki. Path planning using lazy PRM. *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'00),* 2000.

[BL91] J. Barraquand y J. C. Latombe. Robot motion planning> a distributed representation approach. *IEEE International Journal of Robotics Research, vol 10 (6),* 1991.

[BMT96] R. Boulic, R. Mas y D. Thalmann. A robot approach for the center of mass position control with inverse kinematics. *Journal of Computers and Graphics, vol. 20, no. 5,* 1996.

[BMTT90a] R. Boulic, N. Magnenat-Thalmann y D. Thalmann. Coachtrainee: A new methodology for the correction of predefined motions. *Proceedings of Eurographics Workshop on Animation and Simulation,* 1990.

[BMTT90b] R. Boulic, N. Magnenat-Thalmann y D. Thalmann. A global human walking model with real-time kinematic personification. *The visual computer.* 1990.

[BMTT90c] R. Boulic, N. Magnenat-Thalmann y D. Thalmann. Human free walking model for a real-time interactive design of gaits. *Computer animation'90.* 1990.

[BPW92] N. Badler, C. B. Phillips y B. L. Webber. Simulating humans: Computer graphics, animation and control. *Oxford University Press,*

University of Pennsylvania, Philadelphia, 1992.

[Bro89] R. A. Brooks, Robot beings. 1989.

[BW71] N. Burtnyk y M. Wein. Computer-generated key-frame animation. Journal of SMPTE, 1971.

[CC78] T.W. Calvert y J. Chapman. Notation of movement with computer assistance. *Proceedings of ACM Annual Conference,* 1978.

[Chu00] S. K. Chung. Interactively responsive animation of human walking in virtual environments. *These de l'Universite George Washington,* 2000.

[CK00] K. J. Choi y H. S. Ko. Online motion retargeting. *The Journal of Visualization and Computer Animation,* 2000.
.

[CLS03] M. G. Choi, J. Lee y S. Y. Shin. Planning biped locomotion using motion capture data and probalistic roadmaps. *ACM transactions on Graphics, Vol. 22 (2),* 2003.

[CPK99] K. J. Choi, S. Park y H. Ko. Processing motion capture data to achieve positional accuracy. *The Journal of Graphical Models and Image Processing,* 1999.

[CZ92] D. Chen y D. Zeltzer. Pimp it up: Computer animation for a biomechanically based model of muscle using the finite element method.

[GBUP95] L. Goncalves, E. Di Bernardino, E. Ursella y P. Perona. Monocular tracking of human arm in 3D. *Proceedings of International Conference on Computer Vision,* 1995.

[GF74] V. S.Gurfinkel y S. V. Formin. Biomechanical principles of constructing artificial walking systems. *Theory and Practice of Robots and Manipulators,* 1974.

[GL98] M. Gleicher. Motion editing with space-time constraints. *Proceedings of SIGGRAPH'98,* 1998

[Gle98] M. Gleicher. Retargeting motion to new characters. *Proceedings of ACM SIGGRAPH'85,* 1985.

[GM85] M. Girald y A. A. Maciejewski. Computational Modeling for the computer animation of legged figure. *Proceedings of ACM SIGGRAPH'85,* 1985.

[GMTT89] JP. Gourret, N. Magnenat-Thalmann y D. Thalmann. Simulation of

objects and human skin deformations in grasping task. *Proceddings of the ACM SIGGRAPH'89,* 1989.

[GR96] S. Guo y J. Roberge. A high-level mechanism for human locomotion based on parametric frame interpolation space. *Computer Animation and Simulation'96.* 1996.

[HLM97] D. Hsu, J. C. Latombe y R, Motwani. Path planning in expansive configuration spaces. *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'97),* 1997.

[Hsu00] D. Hsu. Randomized single-query motion planning in expansive spaces. *PhD Thesis Stanford University.* 2000

[IRT81] V. T.Inman, H. J. Ralston y F. Todd. Human walking. *Baltimore, Williams and Wilkins,* 1981.

[KB96] H. Ko y N. Badler. Animating human locomotion in real time using inverse kinematics. *IEEE Computer Graphics and Applications.* 1996.

[KGS02] L. Kovar, M. Gleicher y Schreiner. Footstake cleanup for motion capture. *Proceedings of ACM SIGGRAPH Symposium on Computer Animation (SCA'02),* 2002.

[Kha86] O. Khatib, Realtime obstacle avoidance for manipulators and mobile robots. *International Journal of Robotic Research, Vol. 5 (1),* 1986.

[KL00] J. Kuffner y S. LaValle. RRT-Connect: An efficient approach to single-query path planning. *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'00), 2000.*

[KS01] T. Komura y Shinagawa. Attaching physiological effects to motion-capture data. *Graphics Interface, 2001.*

[KSK00] T. Komura, Y Shinagawa y L. Kinii. Creating and retargeting motion by the musculosketal human body model. *The Visual Computer,* 2000.

[KSL096] L. Kavraki, P. Svetska, J.C. Latombe y M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Proceedings of IEEE Transactions on Robotics and Automation,* 1996.

[Kuf98] J. Kuffner. Goal-directed navigation for animated characters using real-time path planning and control. *CAPTECH,* 1998.

[Kuf99] J. Kuffner. Autonomous Agents for Real Time Animation. *PhD*

Thesis Stanford University. 1999.

[Lat91] J. C. Latombe. Robot motion planning. *Boston: Kluwer Academic Publishers,* 1991.

[LK99] S. M. LaValle y J. Kuffner. Randomized kiodinamic planning. *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'99),* 1999.

[LP83] T. Lozano-Perez. Spatial Planning: A configuration space approach. *IEEE Transactions on Computers C-32(2),* 1983.

[LS99] J. Lee y S. Y. Shin. A hierarchical approach to interactive motion editing for human-like figures. *Proceedings of ACM SIGGRAPH'99,* 1999.

[MBBT00] J. S. Monzanni, P. Baerlocher, R. Boulic y D. Thalmann. Using an intermediate skeleton and inverse kinematics for motion retargeting. *Proceedings Eurographics 2000,* 2000.

[MFCD99] F. Multon, L. France, M. P. Cani y G. Debunne. Computer animation of human walkings: a survey. *The Journal of Visualization and computer Animation,* 1999.

[MTT87] N. Magnenat-Thalmann y D. Thalmann. The direction of synthetic actors in the film rendez-vouz a Montreal. *IEEE Computer Graphics and Applications,* 1987.

[Mul98] F. Multon, Controle du movement des humanoides de sythese. *These, Universite de Rennes I,* 1998.

[NT98] J. P. Nedel y D. Thalmann. Real time muscle deformation using mass-spring systems. *Proceedings of the CGI'98,* 1998.

[PB88] Cary B. Phillips y N. Badler. Jack: a toolkit for manipulating articulated figures, *Proceedings of the 1st annual ACM SIGGRAPH symposium on User Interface Software,* 1998.

[PL05] J. Pettre y J. P. Laumond, A motion capture-based control-space approach for walking mannequins. *Computer Animation and Virtual Worlds Vol. 16,* 2005

[RAL00] D. Raulo, J. M. Ahuactzin y C. Laugier. Controlling virtual autonomous entities in dynamic environments using appropiate sense-plan-control paradigm. *Proceedings of IEEE/RS International Conference on Intelligent Robots and Systems (IROS'00),* 2000.

[RH91] M. H. Raibert y J. K. Hodgins. Animation of dynamic legged

locomotion. *Proceedings of ACM SIGGRAPH,* 1991.

[Roh94] K. Rohr. Towards model-based recognition of human movements in image sequences. *CVGIP: Image Understanding,* 1994.

[Ros99] C.F. Rose. Verbs and adverbs: Multidimensional motion interpolation using radial basis functions. *These de l'Universite de Princeton,* 1999.

[S03] A. Sanchez. Contributionà la planification de mouvements en robotique: Approches probabilistes et approaches deterministes. *PhD Thesis, Universite Montpellier II,* 2003

[SGLM03] B. Salomon, M. Garber, M. Lin y D. Manocha. Interactive navigation in complex environments using path planning. *ACM S. on Interactive 3D Graphics,* 2003.

[SLG01] H. J. Shin, J. Lee, M. Gleicher y S, Y. Shin. Computer puppetry: an importance-based approach. *ACM Transaction on Graphics, Vol. 20, No. 2,* 2001.

[SPCM97] F. Scheepers, R. Parent, W. Carlson y S. May. Anatomy-based modeling of the human musculature. *Proceedings of ACM SIGGRAPH'97,* 1997.

[SS87] L. Sciavicco y B. Siciiliano. A dynamic solution to the inverse kinematic problem of redundant manipulators. *Proceedings of IEEE International Conference on Robotics and Automation. (ICRA'87),* 1987.

[Stu86] D. Sturman. Interactive keyframe animation of 3D articulated models. *Graphic Interface'86, Tutorial on Computer Animation,* 1996.

[SYN01] Z. Shiller, K. Yamane y Y. Nakamura. Planning motion patterns of human figures using a multi-layered grid and the dynamics filter. *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'01),* 2001.

[TGB00] D. Tolani, A. Goswami y N. Badler. Real-time inverse kinematic techniques for anthropomorphic limbs. *Graphical Models 62 (5),* 2000.

[Tha93] D. Thalmann. Using virtual reality techniques in the animation process. *Virtual Reality Systems (Earnshaw R. Gigante M, Jones H eds),* 1993.

[TSC96] D. Thalmann, J. Shen y E. Chauvineau. Fast human body

deformations for animation and vr applications. *Proceedings of CGI'96,* 1996.

[WH97] D. Wiley y H. Hann. Interpolation synthesis for articulated figure motion. *Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS'97), 1997.*

[Wil97] J. Wilhelms, Animals with anatomy. *IEEE Computer Graphics and Applications, Vol. 17, No. 3,* 1997.

[WK88] A. Witkin y Z. Popovic. Motion warping. *Proceedings of SIGGRAPH'95*, 1995.

[Zap91] R. Zapata, Quelques aspects topologiques de la planification de mouvements et des actions reflexes en robotique moile. *These dEtat, University of Montpellier II.* 1991.

[ZLT94] R. Zapata, P Lepinay y P. Thomson. Reactive behaviors of fast mobile robots. *Journal of Robotic Systems, Vol. 11, No. 1,* 1994.

[Zs89] D. Zeltzer y K. Sims. A figure editor and gait controller to task level animation. *SIGGRAPH'88 Tutorial Notes on Synthetic Actors: The Impact of A. I. and Robotics on Animation,* 1989.

Anexo. El motor de física ODE

A.1 Introducción

Open Dynamics Engine (ODE) es una biblioteca libre de calidad industrial para simular la dinámica de cuerpo rígidos independientes o articulada. Es apropiada para la simulación de los vehículos de tierra, patas de criaturas y los objetos que se mueven en entornos de realidad virtual. Es rápido, flexible y robusto, además le ha sido incorporado un sistema de detección de colisiones.
ODE está siendo desarrollado por Russell Smith con la ayuda de varios colaboradores.

A.1.1 Características

- ODE es efectivo en la simulación de estructuras rígidas articuladas de un cuerpo. Una estructura articulada se crea cuando cuerpos rígidos de varias formas se conectan por uniones de distintos tipos. Ejemplos de ello son los vehículos de terreno (en el que están conectados a las ruedas del chasis), patas de criaturas (donde las piernas se conectan al cuerpo), o un conjunto de objetos.

- ODE ha sido diseñado para ser utilizado en simulaciones de tiempo real. Especialmente para la simulación de objetos en movimiento. Esto se debe a que es rápido, sólido y estable, y el usuario tiene plena libertad para modificar la estructura del sistema, incluso mientras la simulación está ejecutándose.

- ODE utiliza un integrador muy estable. Hace hincapié en la estabilidad de la velocidad y la exactitud física.

- ODE incorpora un sistema de detección de colisiones. Sin embargo se puede hacer caso omiso de el y hacer su propia detección de colisiones si así se desea. Las actuales primitivas de colisión son esfera, caja, cilindro, plano, rayos, y mallas triangulares. El sistema de colisión de ODE proporciona identificación rápida de los objetos, a través del concepto de “espacios”.

A.1.2 Particularidades

- Tipos de Uniones: bola-y-socket, bisagra, deslizante (prismáticos), bisagra-2, fija, angular de motor, universal.

- Primitivas de Colisión: esfera, caja, cilindro, plano, rayos, y malla triangular.

- Espacios de Colisión: Árbol Quad, espacio hash y simple.

- Método de Simulación: Las ecuaciones de movimiento se derivan de un multiplicador de Lagrange.

- Un integrador de primer orden se comienza a usar. Es rápido, pero no lo suficientemente preciso para la ingeniería cuantitativa aún. Se espera el desarrollo de integradores de orden superior.

- Modelo de fricción y Contacto: Esto se basa en "Dantzig LCP solver" descrito por Baraff, aunque ODE implementa una aproximación más rápida al modelo de fricción de Coloumb.

- Tiene una interfaz nativa C (aunque ODE es principalmente escrito en C++).

- Tiene una interfaz de C++ construido sobre C y otra de C# construida sobre C++.

- Optimizaciones específicas de plataforma.

A.2 Instancias básicas de ODE

Una vez terminado de configurar el entorno de trabajo, y de haber ejecutado el programa "Hola mundo de ODE", veremos a continuación aspectos básicos de ODE.

A.2.1 Concepto de mundo (world)

El mundo contiene todos los objetos que actualmente existen en el mundo de ODE. Como en el mundo real, tiene su propia línea de tiempo. Los objetos del mundo se moverán de acuerdo a las fuerzas que se estén aplicando en ellos, cada vez que el tiempo avance.

En esta parte del tutorial se usaran dos conceptos fundamentales: el de cuerpo (body) y geometría (geom). En realidad un cuerpo es una entidad con una masa (mass), pero no tiene una forma. Dicha forma es la geometría. Esta distinción es requerida, debido a que las geometrías se usan para la detección de colisiones, y las masas se emplean en la simulación física. Una geometría no necesita tener masa de manera obligatoria. Así que este apartado se dividirá en 2 partes: como crear cuerpos y como crear geometrías.

El mundo (world) se emplea como entorno de simulación física, y puede contener cuerpos. Dicho mundo es adicionado con un conjunto de condiciones que lo caracterizan. A continuación se ejemplifica la manera en que se inicializa un mundo.

```
//Identificación del mundo a crear
dWorldID world;

// Inicializa el mundo
world = dWorldCreate();
dWorldSetGravity(world, 0.0, -9.8, 0.0);
```

Como se observa en el apartado anterior, el mundo puede contar con un una gravedad establecida.

A.2.2 Concepto de cuerpo (body) y masa (mass)

A diferencia de los cuerpos físicos, los cuerpos de ODE no ocupan espacio y no tienen concepto de masa, el cuerpo solo es un contenedor que posee la información referente a la posición, rotación angular y velocidad lineal.
Dado que a cada cuerpo debe agregársele una masa, es útil agrupar ambos conceptos en un solo tipo de dato objeto.

```
typedef struct {
        dBodyID body;
        dMass mass;
} Object;
```

Como se observa, tenemos un identificador de cuerpo, y también una masa, la cual indica la cantidad de masa, el sistema provee además la capacidad para establecer el centro de gravedad e inercia. Para realizar lo mencionado, se procede como se expresa a continuación:

```
/* Generación de los identificadores */
enum { oCube, oSphere, oComplex, objsCount };
Object objs[objsCount];
dReal masses[objsCount] = {1.0, 2.0, 4.0};

// Inicializa objetos
for (int i = 0; i < objsCount; ++i) {
        // Crea un objeto en el mundo
        objs[i].body = dBodyCreate(world);
        //  Crea una masa con la distribución de una esfera
        dMassSetSphereTotal(&objs[i].mass, masses[i], 1.0);
        //Aplica la masa al cubo
        dBodySetMass(objs[i].body, &objs[i].mass);
        // Traslada el objeto
        dBodySetPosition(objs[i].body, (i - 1) * 4.0, 0.0, 0.0);
        // Ajusta una velocidad inicial
        dBodySetLinearVel(objs[i].body, (i - 1) * -1.0, 0.0, 0.0);
}
```

Hasta este punto, los objetos caerán al vacío sin importar que pongamos un piso, dado que el programa no tiene implementado la detección de colisiones. El uso de geometrías y colisiones se detallara a continuación.

A.2.3 Concepto de geometría (geom)

Una geometría representa una forma rígida y es la base del sistema de detección de colisiones.
Definir una geometría es como crear un "cuerpo duro" en la realidad, por ejemplo una mesa, un libro o algo que se puede tocar.
Cuando las geometrías colisionan producen puntos de contacto, los cuales pueden ser usados para determinar aspectos particulares, como rotar el objeto para que siga la pendiente de la superficie por la cual se desliza.

Las geometrías son colocables si es que se pueden mover, o no-colocables en caso contrario. Cada geometría posee un identificador dGeomID, y se puede destruir dicha geometría mediante dGeomDestroy(dGeomID). Las geometrías en realidad son objetos en el código, son instancias de clases tales como *sphere, cube, cylinder*. Es posible implementar nuestros propios tipos de geometría o modificar las clases existentes.

A.2.4 Concepto de espacio (space)

El concepto de espacio es parecido al de mundo, pero un espacio puede contener múltiples geometrías en vez de cuerpos, permitiendo colisiones internas entre ellas. Esto es importante por que permite tener pequeños subsistemas de geometrías u otros espacios, además incrementa la velocidad de cómputo en la simulación.

Primero se define el espacio y se crea una geometría para el plano y las esferas. Posteriormente las esferas colisionarán entre si.
Para dicho objetivo añadiremos la geometría a la estructura, que habíamos creado anteriormente:

```
typedef struct {
        dBodyID body;
        dMass mass;
        dGeom geom;
} Object;
```

Ahora, se tendrá que definir un espacio. Existen diferentes tipos de ellos, pero para nuestra simulación se ajusta perfectamente un espacio simple.

```
dSpaceID space;
...
space = dSimpleSpaceCreate(0);
```

Se pueden establecer jerarquías de espacios. Así que se puede especificar el espacio padre cuando se crea uno nuevo. En este caso, este es la raíz (y el único espacio en nuestro mundo), así que emplearemos el 0 como parámetro.
Finalmente podemos inicializar las geometrías, a continuación se muestra el código de inicialización:

```
for (int i = 0; i < objsCount; ++i) {
        //Crea un objeto en el mundo
        objs[i].body = dBodyCreate(world);
```

```
            //Crea una masa con la distribución de un cubo
            dMassSetBoxTotal(&objs[i].mass, masses[i], 1.0, 1.0, 1.0);
            // Aplica la masa al cubo
            dBodySetMass(objs[i].body, &objs[i].mass);
            // Traslada el cuerpo
            dBodySetPosition(objs[i].body, (i - 1) * 4.0, 0.0, 0.0);
            // Añade una velocidad inicial
            dBodySetLinearVel(objs[i].body, (i - 1) * -1.0, 0.0, 0.0);
            // Crea una geometría
            objs[i].geom = dCreateSphere(space, (i / 3.0) + 1.0);
            dGeomSetData(objs[i].geom, names[i]);
            dGeomSetBody(objs[i].geom, objs[i].body);
}
```

Para definir un objeto se hace lo siguiente:

- Se crea un cuerpo en el mundo.
- Se crea una masa con características.
- Se asigna la masa al cuerpo creado.
- Se establecen las características del cuerpo.
- Se crea una geometría
- Se establecen las características de la geometría
- Se asigna un cuerpo a la geometría creada.

Como se detalla, *dCreateSphere* incluye la creación de la forma y la geometría de una esfera.

A.2.5 Concepto de unión (joint)

Las uniones permiten a dos o más cuerpos establecer posiciones y rotaciones condicionadas por el tipo de unión establecida. Esta relación esta determinada por el tipo de bisagra que se especifica.

A.2.6 Concepto de unión de contacto (contact joints) y colisiones

Sin embargo, las uniones también pueden ser de contacto, éstas se forman cuando dos cuerpos chocan entre sí. Se trata de una asociación temporal que describe las características físicas de los cuerpos, como son la elasticidad y la fricción.

Para manejar las colisiones, necesitamos implementar una función *callback*, esta función será llamada cada vez que dos objetos colisionen. A continuación se muestra el código de dicha función.

```
void nearCallback(void *unused, dGeomID o1, dGeomID o2) {
            // Cuerpo 1 involucrado en la colisión
```

```
            dBodyID body1 = dGeomGetBody(o1);
            // Cuerpo 2 involucrado en la colisión
            dBodyID body2 = dGeomGetBody(o2);
            // Conjunto de puntos de contacto entre las geometrías
            dContact contact[MAX_CONTACTS];
            for (int i = 0; i < MAX_CONTACTS; i++)
            {
                        // Caracteristicas de la superficie
                        contact[i].surface.mode = dContactBounce; // Estilo de rebote
                        contact[i].surface.bounce = 0.5; // Indice de Rebote
                        contact[i].surface.mu = 100.0; // Indice de Fricción
            }


            int collisions = dCollide(o1, o2, MAX_CONTACTS, &contact[0].geom, sizeof(dContact));
            if (collisions) {
                        for (int i = 0; i < collisions; ++i) {
                                    dJointID c = dJointCreateContact(world, contactGroup,
                                                contact[i]);
                                    dJointAttach(c, body1, body2);
                        }
            }
}

dJointGroupEmpty(contactGroup);
```

Los valores de los parámetros pueden ser modificados para observar el comportamiento físico del objeto y así entender de mejor manera el funcionamiento de la simulación.

A.2.7 Concepto de fuerza (force)

Sin fuerzas, ninguno de los cuerpos se movería a ningún lado. Una vez especificada una fuerza, los cuerpos se moverán continuamente debido a la inercia y no se detendrán hasta que sufran un frenado vía fricción.

A.3 ODE.NET

ODE.NET es un motor de simulación física que permite a los programadores imitar interacciones de la vida real en ambientes virtuales utilizando la familia de lenguajes de programación .NET.

ODE.NET permite a los usuarios el uso del famoso motor de física ODE (Open Dynamics Engine) mediante un adaptador. Dado que ODE solo se encarga de la simulación. Los programadores deben transferir los valores de ODE y mostrarlos mediante un renderizador independiente para hacer visibles los resultados. Además ODE no restringe el uso de diversos detectores de colisión dado que cuenta con uno propio pero permite el uso de una herramienta de detección independiente.

En ODE, se tiene dGeom, dWorld, dMass, dBody, dJoint, dJointGroup al igual que sus identificadores de referencia como son dGeomId, dWorldId, etc. Sin embargo, ODE.NET es un adaptador y como tal es incapaz de reproducir estas clases individualmente de manera efectiva. Lo que se tiene en ODE.NET son las referencias genéricas y ambiguas, IntPtr que pueden significar desde dGeomId hasta dBody.

Dado que IntPtr es tan confuso como tal, es útil definir una clase personalizada que actúe como contenedor para organizar entre los diferentes tipos de IntPtr.

A.3.1 Descargar ODE.NET

Conectarse a http://sourceforge.net/projects/opende/, que es la página que ODE tiene reservada en SourceForge.net, y ve a la sección de descargas.

Obtén un paquete de la última construcción de ODE. Descarga tanto la especificación de lenguaje como el adaptador de ODE.NET.

Figura A. 1: Comprimidos de ODE para descargar

El paquete que contiene ODE.NET, esta listo para usarse, si desea hacer algún ajuste en el código de ODE, es necesario abrir el código fuente, modificarlo, y recompilarlo por separado.

A.3.2 Instalar ODE.NET

Para usar ODE.NET en un proyecto creado, es necesario agregar la referencia (ya sea en Visual Studio, SharpDevelop o Mono) como se ilustra a continuación, al seleccionar ODE.net.dll.

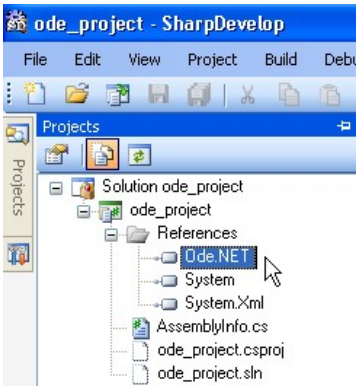

Figura A. 2: Sección de referencias del proyecto

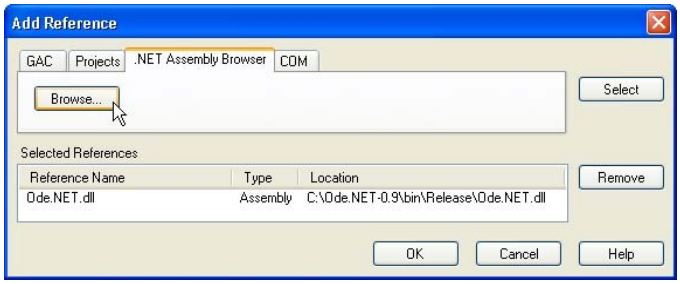

Figura A. 3: Ventana para agregar referencias de proyecto

En cada archivo de código se debe agregar la referencia a ODE.NET agregando la directiva:

using Ode.net;

De esta manera, es posible utilizar el motor de física ODE en código escrito en C# para obtener simulaciones realistas. Sin embargo ODE.NET no cuenta con capacidades de renderizado por lo que se necesita un sistema de renderizado independiente (Tao.OpenGL, DirectX) para mostrar de manera gráfica el resultado de la simulación generado por ODE.

A.3.3 Ejemplo de ODE.NET

El ejemplo siguiente es la sección dedicada a la física de un programa que muestra diferentes prismas rectangulares en caída libre. Está escrito en el lenguaje de programación C# utilizando trozos de código de ODE.NET. El funcionamiento del código esta descrito en las secciones de comentario.

```
//Nombres de espacios a quienes se hace referencia en el código
using System;
using System.Collections.Generic;
using Ode.NET;

namespace Fisica
{
   #if dDOUBLE
            using dReal = System.Double;
```

```
#else
            using dReal = System.Single;
#endif

public class MundoFisica
{
        //Tipo de dato que se devolverá al renderizador para ser dibujado.
    public struct Geomdibujable
    {
        public Vector lados;
        public float [] matrizrt;

        public Geomdibujable(d.Vector3 lados, d.Vector3 pos, d.Matrix3 rot)
        {
            this.lados = new Vector(lados.X, lados.Y, lados.Z);
            matrizrt = new float[16];

            matrizrt[0] = rot.M00;
            matrizrt[1] = rot.M01;
            matrizrt[2] = rot.M02;
            matrizrt[3] = 0;
            matrizrt[4] = rot.M10;
            matrizrt[5] = rot.M11;
            matrizrt[6] = rot.M12;
            matrizrt[7] = 0;
            matrizrt[8] = rot.M20;
            matrizrt[9] = rot.M21;
            matrizrt[10] = rot.M22;
            matrizrt[11] = 0;
            matrizrt[12] = pos.X;
            matrizrt[13] = pos.Y;
            matrizrt[14] = pos.Z;
            matrizrt[15] = 1;
        }
    }

    #region constantes
    const int NUM = 100; // Maximo numero de objetos en escena
    const float DENSITY = 5.0f;
    const int MAX_CONTACTS = 8; //Maximo de contactos en simulacion
    #endregion

    #region Variables Miembro
    static IntPtr world; // Mundo en que se encuentran nuestros cuerpos
    static IntPtr space;// Espacio en que se detectan las colisiones
    static IntPtr contactgroup; // Grupo de contactos generado por el detector de colisiones
    public static List<IntPtr> obj = new List<IntPtr>();        // objetos en escena
    static d.ContactGeom[] contacts = new d.ContactGeom[MAX_CONTACTS];
    static d.Contact contact;
    static d.NearCallback nearCallback = near; // apuntador a funcion near usada por el detector de colisiones
    public List<Geomdibujable> geomsadibujar = new List<Geomdibujable>(); //lista de geoms a renderizar
    #endregion

    public MundoFisica()
    {
        // Inicializacion de la escena
            world = d.WorldCreate();//creación del mundo
            space = d.HashSpaceCreate(IntPtr.Zero); //creación del espacio
            contactgroup = d.JointGroupCreate(0);//creación del grupo de contactos
            d.WorldSetGravity(world, 0.0f, -9.81f, 0.0f);//especificación de la gravedad del mndo
            d.WorldSetCFM(world, 1e-5f);
            d.WorldSetAutoDisableFlag(world, true);
            d.WorldSetContactMaxCorrectingVel(world, 0.1f);
            d.WorldSetContactSurfaceLayer(world, 0.001f);
            d.CreatePlane(space, 0, 1, 0, 0);//creación del plano

        // Configuracion de los parametros de respuesta al contacto
            contact.surface.mode = d.ContactFlags.Bounce | d.ContactFlags.SoftCFM;//caracteristicas del plano
            contact.surface.mu = d.Infinity;//Índice de fricción del plano
            contact.surface.mu2 = 0.0f;
            contact.surface.bounce = 0.01f;//Índice de elasticidad del plano
            contact.surface.bounce_vel = 0.1f;
            contact.surface.soft_cfm = 0.01f; //Índice de suavidad del plano
```

```
    }

    ~MundoFisica()
    {
        // Limpieza
        d.JointGroupDestroy(contactgroup);
        d.SpaceDestroy(space);
        d.WorldDestroy(world);
        d.CloseODE();
    }

    //Crea un cubo de tamaño aleatorio
    public void CreaCuboAlAzar()
    {
        IntPtr geom;
        d.Mass mass;
        //se crean 3 valores aleatorios de tamaño x, y, z
        d.Vector3 sides = new d.Vector3(d.RandReal() * 0.5f + 0.1f, d.RandReal() * 0.5f + 0.1f, d.RandReal() * 0.5f +
0.1f);
//          d.Vector3 sides = new d.Vector3(1,1,1);

        d.MassSetBox(out mass, DENSITY, sides.X, sides.Y, sides.Z);
        geom = d.CreateBox(space, sides.X, sides.Y, sides.Z);
        addObject(geom, mass);
    }

    // Agrega un nuevo objeto a la escena - agrega body al geom y
    // asigna la posicion inicial y la orientacion
    static void addObject(IntPtr geom, d.Mass mass)
    {
        // Crea un body a este objeto
        IntPtr body = d.BodyCreate(world);
        d.GeomSetBody(geom, body);
        d.BodySetMass(body, ref mass);
        if (obj.Count < NUM)
            obj.Add(geom);
        else
        {
            d.BodyDestroy(body);
            d.GeomDestroy(geom);
        }

        // Asigna la posicion y rotacion a este nuevo objeto (aqui aleatoria)
        float[] m = Matrix.Transform(45, 0, 0, 0, 0, 0);
        d.Matrix3 R = new d.Matrix3(m[0], m[4], m[8], m[1], m[5], m[9], m[2], m[6], m[10]);
        d.BodySetPosition(body, d.RandReal() * 2 - 1, d.RandReal() + 2,d.RandReal() * 2 - 1);
        d.BodySetRotation(body, ref R);
    }


    // Near callback - Crea las uniones de contacto
    static void near(IntPtr space, IntPtr g1, IntPtr g2)
    {
        IntPtr b1 = d.GeomGetBody(g1);
        IntPtr b2 = d.GeomGetBody(g2);
        if (b1 != IntPtr.Zero && b2 != IntPtr.Zero && d.AreConnectedExcluding(b1, b2, d.JointType.Contact))
            return;

        int count = d.Collide(g1, g2, MAX_CONTACTS, contacts, d.ContactGeom.SizeOf);
        for (int i = 0; i < count; ++i)
        {
            contact.geom = contacts[i];
            IntPtr joint = d.JointCreateContact(world, contactgroup, ref contact);
            d.JointAttach(joint, b1, b2);
        }
    }

    // Dibuja un objeto en la escena al agregarlo a la lista del renderizador
    void drawGeom(IntPtr geom,int indice)
    {
            //Extrae el cuerpo de la geometría proporcionada
        IntPtr body = d.GeomGetBody(geom);
```

```
            //Extrae la posición del cuerpo
        d.Vector3 pos;
        d.BodyCopyPosition(body, out pos);

            //Extrae la rotación del cuerpo
        d.Matrix3 R;
        d.BodyCopyRotation(body, out R);

            //Obtiene el tipo de geometría a dibujar
        d.GeomClassID type = d.GeomGetClass(geom);
        switch (type)
        {
            case d.GeomClassID.BoxClass:
                d.Vector3 sides;
                d.GeomBoxGetLengths(geom, out sides);
                if (geomsadibujar.Count <= indice)
                    geomsadibujar.Add(new Geomdibujable(sides, pos, R));
                else
                    geomsadibujar[indice]=new Geomdibujable(sides, pos, R);
                break;
        }
    }


    // Llamado una vez por recuadro, actualiza las posiciones de la escena
    public void step(bool pause)
    {
        d.SpaceCollide(space, IntPtr.Zero, nearCallback);
        if (!pause)
            d.WorldQuickStep(world, 0.02f);
        d.JointGroupEmpty(contactgroup);

        for (int i = 0; i < obj.Count; i++)
            drawGeom(obj[i],i);
    }
  }
}
```

La clase MundoFisica devuelve una lista de objetos de la estructura Geomdibujable, que son los objetos que tienen que dibujarse en cada paso del reloj temporizador. El renderizador puede ser programado en cualquier motor grafico. A continuación se muestra una captura de pantalla del programa diseñado utilizando Tao.OpenGL como motor gráfico.

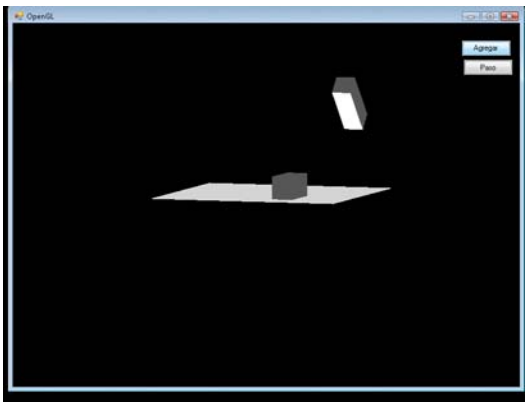

Figura A. 4: Captura 1 del ejemplo

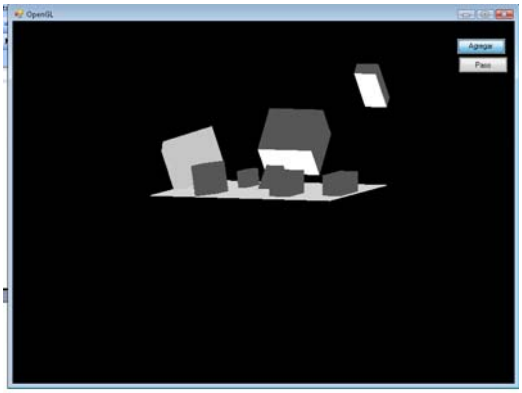

Figura A. 5: Captura 2 del ejemplo

En el ejemplo anterior se puede observar que el programa simula un conjunto de prismas rectangulares de distintos tamaños que se desplazan en caída libre. Al colisionar entre ellos se aprecia un realismo considerable provisto por el motor de física ODE.