UNIVERSIDAD DE LAS CIENCIAS INFORMATICAS

Facultad 5

Control Dinámico de personajes

Trabajo de diploma para optar por el
Título de Ingeniero en Ciencias Informáticas

Autor: Freddy Campbell Mendoza

Tutores: Ing. Liudmila Pupo Peña
MSc. Yanoski Rogelio Camacho Román

Ciudad de la Habana, 2010

Dedicatoria

A mi madre que es lo más sagrado que tengo en la vida,

A mi padre, mi ejemplo a seguir en todo momento,

A mi hermano, sin duda el mejor del mundo.

Agradecimientos

A mi papá por todo ser mi soporte en mi vida estudiantil,

A Lidia por hacer el mayor de sus esfuerzos,

A mi hermano por ayudarme en los momentos cruciales de mi vida,

A mi abuela por su apoyo, amor y cariño,

A Arianna, Taelen y Néstor por ser los mejores amigos del Mundo,

Al piquete de la STK, en especial, Laura, Alfredo y "El Doctor",

A todos los que de una forma u otra han hecho posible este sueño realidad.

Resumen

Las aplicaciones de Realidad Virtual en los últimos años han tenido gran auge en un amplio número de sectores de la sociedad. Estas aplicaciones casi siempre tienen como objetivo acercar lo más posible a la realidad las simulaciones realizadas. Para ello los desarrolladores, principalmente de videojuegos, utilizan el manejo de datos de movimientos o animaciones prediseñadas logrando poco realismo cuando los personajes interactúan dentro de un entorno dinámico bajo diferentes eventos físicos. La simulación de personajes utilizando leyes físicas es óptima para que actúen con la mayor realidad posible, pero es una tarea difícil que abarca un gran número de ecuaciones y procedimientos. El objetivo de este trabajo es mezclar estas dos tendencias, de manera tal, que aprovechemos las ventajas de cada una para lograr la animación lo más real posible.

En la investigación realizada se abordan los conceptos acerca de las animaciones, específicamente las prediseñadas y físicas, necesarias para simular cinemática y dinámica de los personajes respectivamente. Se analizan las principales técnicas para lograr dichas animaciones y alcanzar interacción entre ellas; y se presenta una solución al problema planteado mediante la implementación de sistema de control dinámico de personajes.

La solución resultante de este proyecto, es desarrollada usando el proceso unificado de software e implementado en C++ estándar, está acoplada a la herramienta “SceneToolKit” y permitirá un mayor realismo en los productos finales que se elaboren, al lograr incorporar propiedades físicas como complemento de las animaciones prediseñadas que determinen el comportamiento de personajes humanoides.

Palabras Claves

Realidad Virtual, simulación física, dinámica de cuerpos rígidos, cinemática, animación prediseñada, Controladores PD.

Contenido

INTRODUCCIÓN

La gráfica por computadora es una de las ramas de las ciencias de la computación que goza de mayor popularidad en la actualidad. Su acelerado desarrollo tiene un marcado impacto en el avance de las ciencias, tal es el caso de los sustanciales logros que se han alcanzado en los Sistemas de Realidad Virtual (SRV).

Las crecientes potencialidades de las computadoras hoy día posibilitan la representación de grandes y complejos entornos virtuales. La presencia de objetos animados es de vital importancia en estas aplicaciones dotándolas de gran dinamismo y realismo, ejemplo de ello son los animales, plantas, automóviles, aviones, etc.

En los videojuegos, simuladores, entrenadores, es necesaria la utilización de personajes[1], y frecuentemente se utilizan animaciones predefinidas para simular comportamientos propios del ser humano, siempre tratando de que sea lo más real posible. Se pueden simular así acciones que el propio hombre no puede realizar en la vida real. Se pueden desarrollar aplicaciones que tienen gran importancia económica e incluso muchas de ellas contribuyen a la preservación de la vida humana.

Figura 1 Videojuegos, simuladores y entrenadores que trabajan con animación de personajes

En contraste, los personajes en ocasiones se comportan irreales cuando interactúan con entornos dinámicos, debido a que sus movimientos son predefinidos mediante animaciones y no logran una simulación cercana a la realidad. [1]

[1] Temática de la animación que trata la simulación de los distintos movimientos de humanoides bípedos.

En el Polo Productivo de Realidad Virtual de la Universidad de las Ciencias Informáticas se desarrolla una Herramienta de Desarrollo para Sistemas de Realidad Virtual (HDSRV) que lleva por nombre SceneToolKit, que más que una herramienta básica, es un conjunto de herramientas con diferentes funcionalidades para la creación de aplicaciones de realidad virtual como son los Entrenadores y los Videojuegos, y que puede dar soporte a los proyectos dentro de la UCI encargados de desarrollar productos de este tipo. Contiene clases para el manejo de elementos de la escena, como es el caso particular de las animaciones (prediseñadas) mediante un módulo de animación de personajes. Asimismo, cuenta con un módulo físico cuya implementación básica está soportada utilizando la Biblioteca Física Bullet que permite controlar objetos rígidos articulados mediante propiedades físicas en un Entorno Virtual.

Esta Herramienta carece de funcionalidades que integren las animaciones prediseñadas de personajes con la simulación física para lograr un mayor nivel de realismo en las escenas del “mundo virtual”, donde los personajes tengan un comportamiento natural y lógico.

En consecuencia existe la **necesidad** de combinar estas dos formas de animación: la de movimientos predefinidos (Cinemática), y la animación mediante simulación física en un Entorno (Dinámica) para su integración e interacción.

Por tanto el **problema** a resolver por este trabajo es ¿Cómo combinar las animaciones predefinidas y simulación dinámica de personajes para la biblioteca SceneToolKit?

El **objeto de estudio** de este trabajo son las animaciones de personajes. Por tal motivo se plantea como **objetivo** incorporar la animación de personajes en la SceneToolKit mediante la combinación de animaciones predefinidas y animaciones físicas.

Como **campo de acción** se establecen los sistemas que combinan las animaciones predefinidas y aquellas que se realizan mediante la aplicación de leyes físicas a los personajes.

Para esto se definen un grupo de tareas que se enuncian a continuación:

- Estudio y evaluación de los motores físicos más importantes para seleccionar el que permita la simulación de propiedades físicas de personajes según el problema planteado.
- Descripción de las principales técnicas, tendencias y métodos que combinan animaciones prediseñadas y dinámica de personajes.
- Evaluación de la arquitectura y funcionalidades de la SceneToolKit (STK) para adicionarle las nuevas funcionalidades.
- Planteamiento de las soluciones técnicas que permitan cumplir el objetivo propuesto.
- Desarrollo de la solución propuesta realizando la ingeniería de software correspondiente.

Con la realización de estas tareas se espera poder combinar las animaciones prediseñadas y la simulación dinámica a través de la aplicación de leyes físicas; logrando mayor realismo en los comportamientos de los personajes en la escena mediante el uso de un control dinámico que complemente las animaciones prediseñadas para tales propósitos.

El desarrollo de este trabajo estará organizado de la siguiente manera:
En un primer capítulo, “Fundamentación Teórica”, se hace un análisis bibliográfico donde se investigan las características de las animaciones, los motores físicos más importantes y sus características, los algoritmos, tendencias, métodos y técnicas actuales necesarias para la creación de sistemas híbridos de animación de personajes con el objetivo de llevarlas a los entornos virtuales. En el capítulo 2 “Soluciones Técnicas”, se exponen las características técnicas que presentará dicho sistema como solución a los problemas planteados. En el capítulo 3, “Descripción de la solución propuesta”, se hace la captura de requisitos y se crean los modelos de casos de uso del sistema. En el capítulo 4, “Diseño e Implementación del sistema”, se presentan los diagramas de clases de diseño. Finalmente, se ofrece un glosario de términos para ayudar a la comprensión del lenguaje técnico utilizado a lo largo del trabajo.

Capítulo 1: Fundamentación Teórica

Introducción

La utilización de animaciones en las aplicaciones gráficas son muy útiles y bien vistas por los usuarios. Las figuras animadas ya no son totalmente dependientes de la habilidad del animador por su realismo, y los nuevos métodos proporcionan formas de animación de personajes con muy poca intervención de este [2]. Los desarrolladores de videojuegos y producciones fílmicas, siempre trabajan en pos de hacer cada vez más reales las simulaciones de los movimientos de personajes. Pero cuando se utilizan animaciones predefinidas para simular el comportamiento de personajes en un entorno dinámico, no siempre se logra el objetivo que se persigue y puede lucir irreal[1], por ejemplo en un juego de *baseball* cuando el *pitcher* realiza un lanzamiento que golpea al bateador, como la animación es prediseñada, no se tiene en cuenta la dirección, la velocidad de la pelota en contraste con el lugar del cuerpo donde pegó la misma y la reacción del personaje, en este caso el bateador.

Esto es un gran problema cuando se simulan comportamientos con animaciones predefinidas solamente, debido a que en un entorno dinámico pueden ser varias las respuestas que podría(n) dar el(los) personaje(s) y que no pueden ser premeditadas, es decir, no se puede predecir una futura acción que resulte como consecuencia de alguna otra del (los) personaje (s).

En el presente capítulo se hará un estudio de los principales conceptos que tienen la simulación física en aplicaciones gráficas así como de los principales motores físicos buscando el más óptimo para nuestro objetivo. Asimismo se describen definiciones acerca de las animaciones, esencialmente la de personajes y se analizan tendencias actuales de cómo se combinan e interactúan las animaciones prediseñadas y la aplicación de propiedades físicas a los personajes.

1.1 Animación Gráfica

La animación en general es la simulación de un movimiento, creada por la muestra de una serie de imágenes o fotogramas. Un ejemplo sencillo de esto son las caricaturas que pertenecen a la animación tradicional (Figura 2). Con el paso del tiempo, la animación de imágenes ha evolucionado de forma considerable, hace algunos años se debía dibujar cada fotograma y se unían para formar una imagen animada. Ahora, el uso de la computadora permite crear escenas mucho más reales (Figura 3).

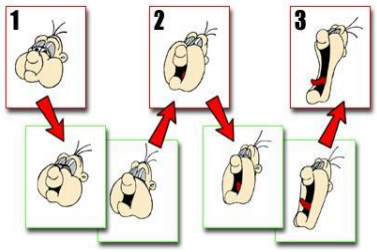

Figura 2 Animación Clásica

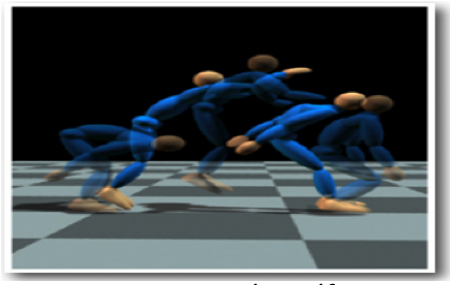
Figura 3 Animación Gráfica

Un concepto básico del cual se debe partir para comprender las nuevas tendencias de la gráfica computacional es la animación. Esta palabra puede interpretarse literalmente como *dar vida* (del griego “anemos” y del latín “animus”). En el campo de la informática gráfica se define como la simulación de movimientos a través de secuencias de imágenes o fotogramas. [3]

Con ella se consigue dar vida a las escenas, por tanto la animación hace referencia a cualquier cambio que se produzca en la escena que se está dibujando y que tenga consecuencias visuales. [4] Una definición más específica y concreta es:

“La animación es la generación, almacenamiento y presentación de imágenes que en sucesión rápida producen sensación de movimiento”. [5]

1.1.1 Animación 3D por ordenadores

Existen varios tipos de animación según su origen, forma de realización y producto final que se desea. Una de ellas es la animación por ordenadores que se podría definir como la creación de una secuencia de imágenes por medio de un proceso automático a partir de una representación de los objetos que forman parte de la escena y de su

movimiento. Los objetos pueden estar representados en una escena bidimensional lo cual tiene un resultado similar a la animación clásica[2], o pueden representarse en escenas tridimensionales en la cual se pueden aplicar métodos realistas de representación.

La animación por ordenadores se basa en una unidad llamada *FRAME* (Marco o fotograma) que no es más que cada imagen estática que forma la secuencia animada.[6] Lo realmente novedoso de la animación 3D por ordenadores es que permite la creación de escenas tridimensionales y su visualización desde diferentes puntos de observación. En este tipo de animación son posibles adicionar elementos y métodos que hacen las escenas mucho más reales como son: enfoque de las cámaras, luces, sombras, simulación física, etc. [4]

La animación 3D por ordenadores generalmente requiere de computadoras potentes por el alto consumo de memoria gráfica y cantidad de cálculos necesaria para representar las escenas por lo que los métodos utilizados deben ser lo suficientemente eficientes para garantizar la velocidad del procesamiento y lograr animaciones fluidas.

1.1.2 Animación de objetos articulados

Se puede apreciar una animación como resultado de cambios de diferentes tipos de un fotograma a otro, los cambios en los objetos pueden ser producto de que su geometría tome otra forma, su textura cambie, o su posición. Ejemplos: animación por deformación, animación de textura y *morphing*.

Figura 4 Objetos Articulados

[2] Consiste en la generación de una secuencia de imágenes por métodos pictóricos, formada a través de píxeles asignada manualmente o semiautomáticamente por mecanismos sencillos guiados manualmente

La animación de objetos articulados es de las más usadas en las modernas aplicaciones de realidad virtual donde se apliquen leyes físicas como la dinámica y cinemática. Actúa sobre cuerpos que están conformados por elementos conectados jerárquicamente como esqueletos o brazos mecánicos. [4]

1.1.3 Animación por esqueleto

Consiste en deformar la malla de un cuerpo a través de una estructura jerárquica de huesos, a los que se le asocia un grupo de vértices. Es una técnica sumamente utilizada actualmente, dada su rapidez para procesar y obtener excelentes resultados, y la posibilidad de incluir una gran cantidad de detalles en la animación de un personaje, desde las arrugas de la piel hasta la definición de los músculos de su cuerpo. [7]

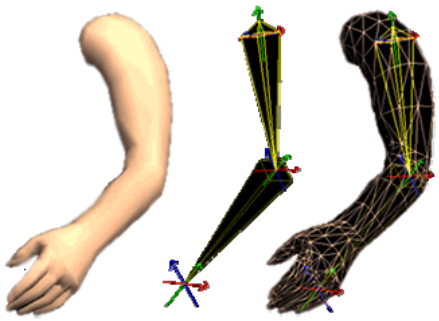

Figura 5 Malla de un brazo asociado a una estructura de

Con el sistema de animación por esqueleto, para representar a un personaje y su animación, se necesitan junto a los datos de una malla 3D, una jerarquía de “huesos”. Estos últimos son una serie de transformaciones jerárquicas que, como los huesos en el cuerpo humano, permiten la deformación de este. Usualmente, los datos “huesos” son representados básicamente como matrices de transformación. Estos eliminan la necesidad de almacenar la posición de los vértices por cada fotograma de la animación [8]. Otras ventajas de la animación por esqueleto está en la suavidad del cambio de una animación a otra, además, como las animaciones son independientes a las mallas, en una escena una misma animación puede aplicársele a diferentes mallas.[7]

1.1.4 Animaciones predefinidas

Las animaciones predefinidas son aquellas que se cargan desde un fichero, o sea, son las que están construidas con antelación por medio de otros *softwares*. Dichas animaciones son muy prácticas debido a su costo relativamente bajo en cuanto a cálculos. Además permiten evaluar con antelación el realismo de los movimientos. [9]

Animaciones desde el diseño

Existen varios *softwares* que tienen como función la de crear o editar animaciones y que soportan las animaciones por huesos. Estos softwares permiten a los diseñadores crear animaciones a su gusto, combinar otras y almacenarlas de forma que hace mucho más rápida la realización de futuros trabajos donde se puedan reutilizar.

Entre los *softwares* que permiten el trabajo con animaciones utilizando la técnica de huesos se pueden encontrar algunos como: 3D Studio Max [10], LightWave [11], Maya [12], Kaydara Motion Builder [13], Blender [14], entre otros.

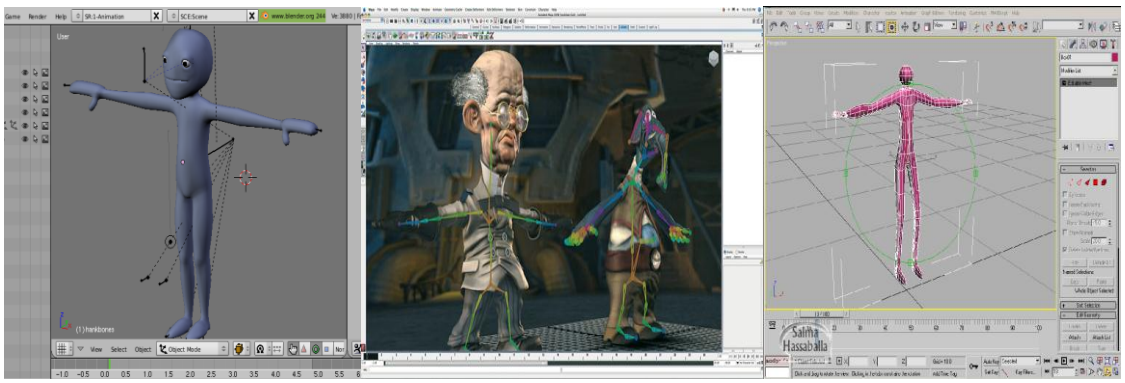

Figura 6 Algunos software que permiten diseñar animaciones

Animaciones desde la captura

Si se busca más realismo en los movimientos y mejores resultados, se puede optar por la Captura de Movimiento (MOCA). Estas técnicas consisten en atrapar los movimientos de actores mediante herramientas que pueden ser diversas dependiendo de la técnica que se utilice para obtener la animación final que sea aplicable a un modelo 3D. [15]

Tipos de sistemas de captura

Sistemas ópticos de captura de movimiento: en ellos se utilizan cámaras para rastrear el movimiento de marcadores acoplados a las articulaciones del cuerpo. El movimiento captado por las diferentes cámaras es combinado por el método de triangulación para calcular sus posiciones fotograma a fotograma en el espacio 3D. Los sistemas sencillos o duales de cámaras son adecuados para la captura facial, un sistema de 3 a 16 o más cámaras son los utilizados para realizar la animación del cuerpo completo.[15]

Sistemas magnéticos de captura de movimiento: Consisten en el uso de un transmisor centralmente localizado y un grupo de receptores adheridos a partes del cuerpo del actor. Estos receptores son capaces de medir su relación espacial con el transmisor central. Cada recibidor es a su vez conectado a una interfaz que puede ser sincronizada. [15]

Sistemas Electro-Mecánicos de captura de movimiento: El actor se coloca una armadura humana de piezas metálicas (similar a un esqueleto). Esta armadura es enganchada hacia la parte trasera de las articulaciones del actor, la armadura posee sensores que van exactamente en las articulaciones del actor para de esta forma sentir las rotaciones que realizan las mismas. Otros tipos de sistemas mecánicos utilizan guantes o modelos articulados. [15]

Figura 7 Animación mediante la captura de movimientos

1.2 Animación Física

Para animar cuerpos con apariencia real, ¿qué mejor que utilizar los modelos físicos que describen su movimiento? Por tanto la animación física se basa en la aplicación de modelos físicos de la realidad para definir la evolución de los elementos animados de la escena.[4] Los objetos se mueven de acuerdo a leyes físicas tales como gravedad, elasticidad, fluidez y asimismo deben reaccionar de acuerdo a estas leyes en cualquier interacción con otro u otros objetos.

Para lograr altos niveles de realismo en las animaciones debemos acercar estas a la naturaleza y a sus leyes. Mediante estas leyes se agregan parámetros reales a los movimientos particularmente del campo de la dinámica y la cinemática.

1.2.1 Cinemática directa

Es la posibilidad de mover algunas de las "piezas" de un personaje o montaje 3D actuando sobre un punto y produciendo un movimiento sobre su eje o centro de rotación (por ejemplo, mover el brazo fijada la rotación sobre el hombro). El programa de animación 3D genera con fórmulas geométricas simples todos los movimientos necesarios de las partes ligadas a su vez a ella. En este caso, en la jerarquía de movimientos o giros definida, se parte de un eje más importante fijo (por ejemplo, el hombro) para mover elementos más sencillos (por ejemplo, el brazo). [16]

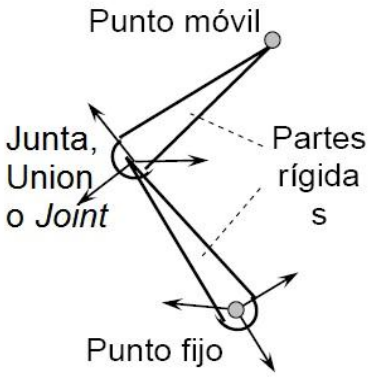

Figura 8 Representación del *joint*

Para la animación de personajes se puede pensar en la cinemática directa como el proceso de especificar explícitamente todos los movimientos de los *joints* (Figura 8) a fin de determinar la posición del último elemento de una cadena jerárquica (conocido como efector final) [2].

1.2.2 Cinemática inversa

Es la posibilidad de que, moviendo elementos más sencillos en la jerarquía, el programa interpola el resto de articulaciones o puntos de giro, que pueden ser configurados por el animador, para conseguir que se muevan acorde a eso. Este tipo de movimiento es mucho más interesante pero a la vez más complejo, ya que en general no hay un sólo modo de rotar los elementos entre sí para conseguir seguir el movimiento final que pretende el usuario [16]. Determina la posición y orientación de todas los *joints* en la herencia dada el estado del efector final. La generación de movimiento realista usando cinemática inversa es una problemática, puesto que el movimiento de las piezas en una figura articulada es identificado usando atributos específicos de movimientos en lugar de usar fuerzas y aceleraciones [2].

1.2.3 La Dinámica

Mientras la cinemática trabaja independiente de fuerzas subyacentes, la dinámica simula las leyes newtonianas con el fin de obtener una verdadera animación física. Se puede obtener gran realismo, pero resulta difícil especificar la animación. Hay que tomar en consideración masas, aceleraciones, grados de libertad, restricciones al movimiento, movimientos prioritarios y otras propiedades físicas.

La dinámica directa controla los objetos mediante manipulación de fuerzas y torques a fin de obtener trayectorias y velocidades. La dinámica inversa, por otra parte, determina las fuerzas y torques que son necesarios para producir movimiento en un sistema. Los sistemas basados en dinámica son frecuentemente conocidos como “físicamente basados” para poder diferenciarlos de los enfoques puramente geométricos como es la cinemática [2], es decir, que cuando nos referimos a la simulación dinámica nos estamos refiriendo a la simulación física y viceversa.

1.2 Animación de Personajes

Después de conocer algunos conceptos importantes de la animación en general, al referirnos a los personajes (específicamente los humanoides), pueden existir las siguientes interrogantes: ¿Qué es la animación de personajes? ¿Cómo es modelado un personaje? La animación de personajes puede ser definida como la expresión de emoción o comportamiento de una vida u objeto inanimado a través del uso de movimiento. La animación de movimientos realistas para personajes humanoides ha sido un difícil problema en el campo de la animación por computadoras. El hecho de que el ser humano reconozca fácilmente las características de un movimiento hace la tarea bastante difícil [2]. Para representar el movimiento de una criatura viviente, como es un humano, es conveniente almacenar las relaciones entre cada una de las partes movibles de la criatura. Por ejemplo la cabeza está unida al cuello, el pie está unido al tobillo, etc. Cada parte movible es conocida como un hueso, las uniones entre cada hueso son *joints* y la colección entera de huesos es conocida como un esqueleto [17].

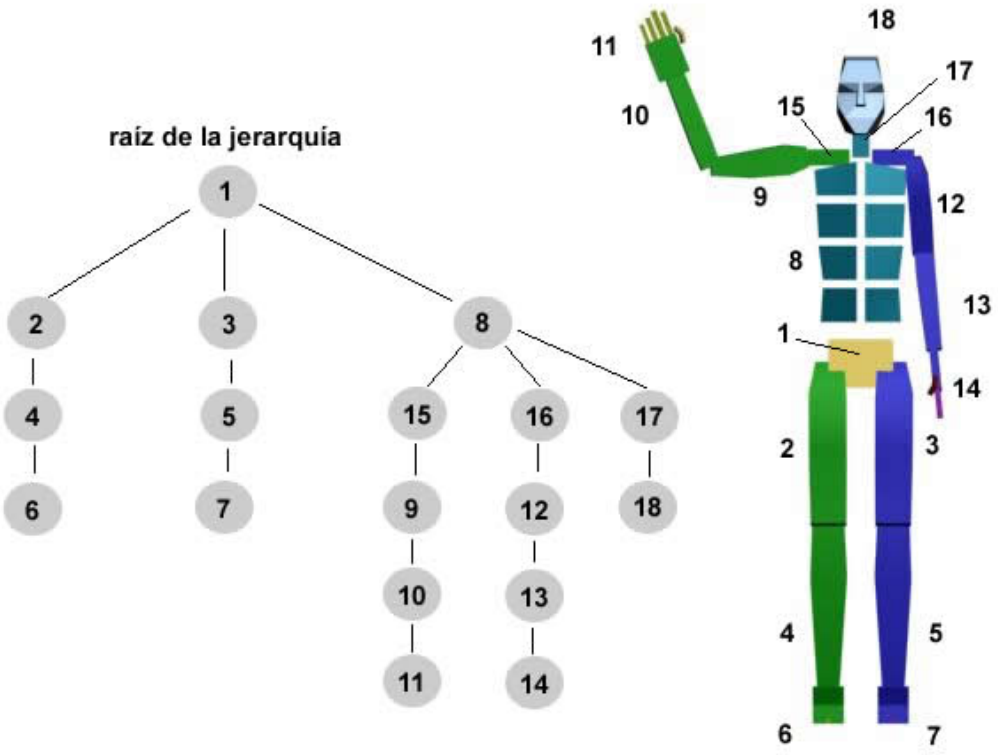

Figura 9 Estructura de huesos de un modelo de esqueleto humano

Los desarrolladores de aplicaciones de realidad virtual han utilizado distintos métodos para la animación de personajes; tratando siempre de lograr el mayor realismo posible de su comportamiento en el entorno. Se han enfocado fundamentalmente en dos tendencias: la animación a través de animaciones predefinidas o captura de movimiento o a través de la simulación física utilizando las leyes newtonianas. Así desde el punto de vista físico la primera contempla la cinemática y la segunda la dinámica.[18] Varios investigadores y desarrolladores han propuesto o implementado técnicas que mezclan estos dos tipos de animaciones.

1.3.1 Técnicas Cinemáticas

El manejo de datos de movimientos para reproducir movimientos humanos es más preciso que otras técnicas como son la cinemática inversa. Muchos desarrolladores han trabajado en esta tendencia a través de técnicas que sintetizan movimientos mediante procesamiento y edición.

Muchas técnicas utilizan la información explícita e implícita de los fotogramas claves (*keyframes*) para producir la mejor animación posible. Los grafos de movimientos utilizan una base de datos de movimientos cinemáticos para capturar la transición entre estos y producirlas entre los fotogramas claves. Las técnicas basadas en trayectoria y restricciones emplean física como parte de un problema de optimización de restricciones. Estas son excelentes herramientas para la creación de movimientos, sin embargo, no pueden ser utilizados para los personajes interactivos en su forma actual [18].

1.3.2 Técnicas Dinámicas

Muchos investigadores han apostado por la simulación de comportamientos de personajes humanos bajo leyes físicas. Usar la simulación física para manejar el movimiento de un humano virtual puede dar un nivel de interacción física con el entorno que las animaciones prediseñadas no pueden igualar, a menos que el movimiento haya sido grabado para una situación específica. [1] Hay varias técnicas dinámicas utilizadas para animar físicamente a los personajes. Hace un tiempo que las industrias de videojuegos comenzaron a tomar como alternativa el uso de la física en

las animaciones de personajes. Como resultado de eso surgió el término de comportamiento *Ragdoll*[3].

Ragdolls

En la simulación 3D el concepto de cuerpos rígidos es ampliamente usado, no solo debido a su simplificación de conceptos físicos, sino también porque simplifica el manejo de colisiones y el *rendering* (dibujado de la escena). [19]

Figura 10 Representación de un personaje *Ragdoll* cayendo por una escalera

Ragdoll, es la habilidad de dejar a los personajes ser influenciados por el entorno circundante mientras aparentemente no pueden sostenerse por ellos mismos. [19]. El sistema interpreta las características del cuerpo como una serie de huesos conectados entre sí, poniéndole diferentes tipos de *joints* para simular las articulaciones. La simulación modela lo que ocurre con el cuerpo cuando se cae al suelo, como si fuese un muñeco de trapo (de ahí el nombre *Ragdoll*).

Es importante hacer énfasis en que el comportamiento *ragdoll* no implica el control mediante física del usuario sobre los personajes. Éste tipo de comportamiento solo concierne los efectos de simulación física para personajes o partes del cuerpo de estos que carezcan de vida. [19]. Este tipo de comportamiento carece que un sistema de control para coordinar los movimientos de las partes del cuerpo que generen un comportamiento cercano al del humano por naturaleza.

[3] El significado de *Ragdoll* proviene de las palabras inglesas *ragdoll* (muñeco de trapo). Como su nombre lo indica, éste es un procedimiento de animación y técnica de simulación para mostrar el movimiento de una persona como si estuviera muerto.

Controladores Dinámicos

Los controladores proveen un mecanismo de bajo nivel para manejar las partes de un personaje humano virtual mediante el cálculo de los torques internos de los distintos *joints* que las unen, lo cual simularía una aproximación a la actuación de un músculo. [1].

Los controladores dinámicos han sido desarrollados para diferentes comportamientos humanos, en sus inicios solo para maniobras atléticas como son correr, montar bicicleta, voltereta, entre otras. Poco después se desarrollaron esquemas de control usando ciclos limitados. Por otra parte se presentó un esquema de control basado en prioridad y que usó máquinas de soporte vectorial para automáticamente determinar el controlador dinámico apropiado para el estado del personaje. Otro de los métodos que usaron controladores dinámicos fue el de estructurar por capas diferentes niveles de controladores para alcanzar simular la natación basada en dinámica de un personaje. En otras investigaciones se lograron implementar otros comportamientos, como la respiración para lo cual se utilizaron simples controladores, para agarrar y soltar se usaron controladores pasivos, entre otras. [20] La mayoría de estos controladores dinámicos son basados en estrategias de controladores proporcionales-derivados.

1.3 Controladores Proporcionales-Derivados

Los controladores proporcionales-derivados (controladores PD) son comúnmente usados por los desarrolladores para guiar la simulación dinámica de una posición actual a una deseada del personaje, y esencialmente basan su estrategia en el cálculo y actualización de los torques que serán aplicados a cada uno de los *joints* del cuerpo en cada instante de tiempo.

Las estrategias híbridas combinan el uso de captura de movimiento u otro método de almacenar movimientos cinemáticos con simulación física, seleccionando para cada instante de tiempo o fotograma el tipo de animación más conveniente para generar el efecto deseado. Shapiro[18] aplicó controladores cinemáticos para movimientos ordinarios que alternaba con controladores dinámicos bajo alteraciones físicas; así como también determinó un método para especificar cuándo sería apropiado alternar los controladores. Michael Mandel[1] usó la misma estrategia bajo contacto físico e hizo énfasis en movimientos reactivos, como son la caída y la recuperación.

Zordan[21] alternó entre la captura de movimiento y la simulación física, mediante el seguimiento de un camino proyectado y usando un simple controlador basado en pose, y así combinar el movimiento con el comienzo de alguna secuencia de captura de movimientos almacenada que se corresponda. Estos métodos no consideran la física del entorno y limitan la interacción física.[22]

Wrotek[22], por otra parte, siguió la captura de movimiento usando espacio globales (*global spaces*) en vez de locales (*local spaces*) bajo un entorno con modificaciones físicas, es decir, manejar todas las modificaciones a los *joints*, como es la aplicación de torques, globalmente en vez de localmente como básicamente se hace. Además incluye una técnica que denomina *weak root springs*, con el objetivo de producir un balance persuasible.

Todos estos métodos utilizan Controladores PD, donde los coeficientes k_s y k_d (coeficiente *spring* y coeficiente *damped* respectivamente) son calculados indistintamente para lograr el efecto deseado.

Los Controladores PD calculan la magnitud del torque alrededor de un *joint* por cada instante de tiempo (o fotograma):

$$\tau = k_s\left(\hat{\theta} - \theta\right) + k_d\left(\hat{\omega} - \omega\right)$$

Donde $\hat{\theta}$ y $\hat{\omega}$ son la posición y la velocidad angular deseadas respectivamente, θ y ω la posición y velocidad angular actual, y k_s y k_d que son los parámetros (coeficientes) especifican la trayectoria del movimiento del *joint*. [23] Cada uno de los sistemas de control dinámico que han usado controladores PD han necesitado encontrar una manera de calcular estos parámetros. La más común ha sido cuidadosamente ajustar estos parámetros hasta ser obtenida la trayectoria del movimiento deseada.

La investigación realizada en [23] aporta como elementos fundamentales el cálculo de k_s y k_d analíticamente eliminando el cálculo "manual" o heurístico implementado en sistemas de control dinámico anteriores, el control del tiempo entre fotogramas independientemente de las interacciones que se produzcan del personaje en el entorno y el control de la tensión, lo cual complementa el control del tiempo y ambos las respuestas dinámicas de manera natural a las perturbaciones.

1.4.1 Interpolación física de Fotogramas

En este último enfoque de control del tiempo y la tensión un fotograma o "fotograma físico" es una estructura definida por la posición del *joint* deseada $\hat{\theta}$, la velocidad angular deseada $\hat{\omega}$ y el tiempo $\hat{t}$: $\gamma = \left(\hat{\theta}_k, \hat{\omega}_k, \hat{t}_k\right)$

El torque del i-ésimo *joint*, τ_i corresponde con el k-ésimo fotograma clave $\gamma_k = \left(\hat{\theta}_k, \hat{\omega}_k, \hat{t}_k\right)$ es calculado usando el controlador derivado-proporcional. Durante el ciclo de actualización del controlador, los parámetros k_s y k_d, son calculados para que pueda alcanzar la animación, en el tiempo requerido, el próximo fotograma clave.

La función ψ es definida para calcular estos parámetros:

$$(k_s, k_d) = \psi(D(q), \gamma_k) = \psi\left(D(q), \left(\hat{\theta}_k, \hat{\omega}_k, \hat{t}_k\right)\right)$$

Donde $D(q)$ es el tensor de inercia compuesto, y depende el estado actual del personaje q. El cálculo de ψ requiere determinar el momento de inercia compuesto, el momento escalar, y la compensación para el efecto del Torque de los *joints* padres (*parent torque*).[23]

1.4.2 Parámetros Derivados-Proporcionales

La función ψ provee los parámetros que permiten a un *joint* en particular alcanzar su posición en un tiempo $\hat{t}$, usando un controlador PD de (1). Esto se puede escribir como una ecuación diferencial de segundo grado en θ, con $\theta' = \omega$ y m como el momento de inercia en el eje del *joint:*

$$m\theta'' - k_s\left(\hat{\theta} - \theta\right) - k_d\left(\hat{\omega} - \omega\right) = 0$$

La solución analítica provee la base para determinar ψ. La diferencia entre θ y ω y el estado del sistema actual es reducido en el tiempo más corto cuando el sistema está "críticamente amortiguado". Esta condición requiere que los parámetros satisfagan $k_s^2 - 4mk_d = 0$, donde m es el momento de inercia en torno al torque aplicado en el eje.[24] Esta restricción permite una que pueda ser derivada una parametrización más

simple usando $\hat{t}$ para eliminar una fracción del error dada f. Un valor razonable para f sería 0.9, el cual reduce el error a un 10% en un tiempo $\hat{t}$.[24] El número n de constantes necesitadas para eliminar f es $n = -1/\ln(f)$.[25] Así se puede determinar la constante de tiempo, λ, necesitada para reducir adecuadamente el error en el tiempo $\hat{t}$: $\lambda = \hat{t}/n$. A partir del este análisis el torque se calcularía:

$$\tau = \frac{m}{\lambda^2}\left(\hat{\theta} - \theta\right) - 2\frac{m}{\lambda}\left(\hat{\omega} - \omega\right)$$

Para determinar m es necesario el tensor de inercia compuesto.

1.4.3 Tensor de Inercia Compuesto

El tensor de inercia compuesto para los *joint i* hasta *n* es:

$$D^{i\ldots n} = \sum_{j=i}^{n} D_j$$

Sin embargo esta expresión funciona solamente para tensores de inercia en el mismo sistema de coordenadas. Para darle solución a este problema se toma cada *joint* conectado se transforma el tensor de inercia del "hijo" en el sistema de coordenadas del "padre", y se reemplaza el tensor de inercia del "padre" por la suma de este con la del hijo. Este método se basa en la formulación recursiva Newton-Euler. Iniciando la recursividad en cada efector final y garantizando acumular más por cada joint, inercia compuesta para cada eslabón se calcula en un tiempo $O(n)$.[23]

Para realizar la transformación desde el "hijo" al "padre", se descompone el sistema de coordenadas en una matriz de rotación R, y la distancia linear desde las coordenadas de fotograma del "hijo" a las del "padre" p. El vector de posición del centro de masa en las coordenadas de fotograma es c. El tensor de inercia del "hijo" D es transformado en las coordenadas de fotograma de su padre $\dot{D}$ mediante:

$$\dot{D} = RDR^T - M\left(\left[p\right]\left[Rc\right] + \left[Rc\right]\left[p\right] + \left[p\right]\left[p\right]\right)$$

Donde M es la masa escalar del "hijo" y [-] es la notación para matrices anti-simétricas por $u \times v = \left[u\right]v$ o el equivalente a:

$$[u] = \begin{pmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{pmatrix}$$

Usando esta expresión, un tensor de inercia compuesto $D_i^{i\ldots n}$ del eslabón *i* puede ser calculado recursivamente, desde el más cercano hasta el más lejano. [23]

1.4.4 Momento de Inercia respecto al Eje del *joint*

Teniendo calculada la tensión de inercia compuesta del *i*-ésimo *joint* en coordenadas locales $D_i^{i\ldots n}$, se puede calcular el momento de inercia con respecto al *joint* del *i*-ésimo eje de torsión s_i usando el teorema del eje paralelo:

$$m_i = s_i \cdot D_i^{i\ldots n} s_i + M_{i\ldots n} \left\| \left(c_i^{i\ldots n} - d_i \right) \times s_i \right\|^2$$

Donde $c_i^{i\ldots n}$ es el vector de masa combinada y $M_{i\ldots n}$ la masa escalar combinada del i-ésimo eslabón y todos sus eslabones "hijos", d_i es el vector de las coordenadas de fotograma a el *joint*, y s_i es un vector unitario. Tanto $c_i^{i\ldots n}$ como $M_{i\ldots n}$ pueden ser calculados cuando se realiza el cálculo del tensor de inercia en el mismo método recursivo, reduciendo la complejidad temporal a $O(n)$.[23]

1.4.5 Compensación del Torque-Padre

El cálculo de ψ en cada actualización del controlador provee una estimación precisa del torque necesario para alcanzar el fotograma clave en un intervalo de tiempo apropiado. Para mejorar su precisión, cada *joint* estima cómo el torque combinado de todos los *joints* "padres" lo afectará. El *joint* entonces es capaz de compensar la aceleración inducida en su eje de rotación s_i. La aceleración total inducida aplicada al *joint i* en coordenadas mundiales es:

$$a_i = \sum_j^{i-1} \frac{\tau_j}{m_j} G_j^w s_j$$

Donde G_j^w es la matriz de transformación de las coordenadas locales del j-ésimo eslabón a coordenadas mundiales, m_j es el momento de inercia compuesto, y τ_j es el torque escalar calculado alrededor del *j*-ésimo *joint*.

El torque necesario para compensar la aceleración angular aplicada a_i es aquella proyectada sobre los ejes del *joint* en las coordenadas mundiales.

$$u_i = a_i \cdot \left(G_j^w s_i\right)$$

Con la adición de este término se obtiene la expresión final para el torque del i-ésimo *joint*:

$$\tau_i = \frac{m_i}{\lambda^2}\left(\hat{\theta}_i - \theta_i\right) - 2\frac{m_i}{\lambda}\left(\hat{\omega}_i - \omega_i\right) + a_i \cdot \left(G_j^w s_i\right)$$

Este torque final es aplicado al *joint* debiendo alcanzar valores que no sobrepasen el torque máximo permitido por el *joint* ($\tau_i < \tau_{\max}$). Si el torque sobrepasa el valor del torque máximo permitido ($\tau_{\max}$) entonces no sería suficiente con los valores de los parámetros calculados para que el personaje alcance la posición deseada.[23] Se pudiera manejar este problema mediante tratamiento de errores o mediante condiciones.

Primero, la pasada de la inercia compuesta, desciende recursivamente desde cada efector final hasta la raíz, visitando cada eslabón una vez, para acumular la inercia compuesta para el estado actual $D(q)$. Segundo, el torque es calculado una vez por cada *joint*, acumulando la aceleración angular a_i desde la raíz hasta el efector final.

1.4 Motores Físicos

En las aplicaciones de realidad virtual se persigue siempre simular los comportamientos de los objetos, personajes, autos, y demás, de la manera más natural y real posible. Una manera para imprimir realismo y presencia en un entorno virtual es imprimir física. Mediante física los objetos podrán rebotar, las esferas rodar por una colina, como mismo podrían hacerlo en la vida real [26]. Para ello se tiene en cuenta entre otros aspectos que los cuerpos cumplan con las leyes físicas como puede ser la gravedad, acción/reacción, leyes de newton, etc. Para lograr tales

propósitos se utiliza un motor físico que se podría definir como un programa de computadora que simula modelos físicos, usando variables como son la masa, velocidad, fricción, resistencia al viento, entre otras [27] o como una parte de un programa que calcula como los objetos podrían e interactuar con otros [26]. Puede también simular y predecir efectos bajo diferentes condiciones que podría aproximarse a lo que pasa en la vida real o en un mundo fantástico. Un motor físico es difícil de implementar debido a que envuelven varias áreas de investigación y a menudo requiere un alto nivel de conocimiento [28]. Se aplican fundamentalmente en las simulaciones científicas y en los videojuegos. Para las primeras se utilizan los motores físicos que se centran en la alta precisión mientras para los segundos se utilizan las que se centran en física de tiempo real.[27] En los últimos años han sido desarrollados motores físicos que bien son "código abierto" y/o libre para uso no comercial.

Desarrollar un motor físico desde cero para cumplir el objetivo de esta investigación no es objetivo puesto que existen muchos que ofrecen funcionalidades que pueden ayudar a satisfacer los requerimientos de cualquier tipo de aplicación gráfica, que son más maduros y tienen muchos años de experiencia y desarrollo. Por tal motivo, a continuación, se muestra un estudio de algunas de los motores físicos más populares que actualmente utilizan los desarrolladores gráficos.

Para el análisis de estos motores se utilizaron los siguientes parámetros [26]:

- Las características: Se necesita conocer cuáles son las principales características de los motores físicos: ¿cuáles son las primitivas que usan? ¿Soporta objetos deformables? ¿Soporta "*Ragdoll*" o cualquier otra funcionalidad especial? Así como otras funcionalidades que nos pueda aportar.

- La documentación: La documentación es muy importante para poder entender cómo funciona el motor gráfico, así como para saber cómo utilizar sus funcionalidades para poder lograr los objetivos que se tracen. Es importante saber de qué manera brinda la documentación, si posee ejemplos implementados para su mejor comprensión, cuán grande es su comunidad de desarrollo entre otros aspectos.

- La usabilidad: Este requerimiento es muy importante para saber si se fácil de integrar con motores gráficos o programas: Qué paradigma utiliza en su forma de programación, cuán fácil es su uso. Las plataformas en las que se puede desarrollar. La Usabilidad estará muy ligada a la documentación y disponibilidad de ejemplos implementados que tengan dichos motores físicos.

Entre los motores físicos más populares se encuentran:

- Open Dynamics Engine
- NVIDIA PhysX
- Bullet

Figura 11 Principales bibliotecas o motores físicos utilizados para la simulación dinámica de cuerpos rígidos articulados

A continuación se muestran las características más importantes de cada uno de ellos.

ODE (Open Dynamics Engine)

Esta es un popular motor físico *"Open Source"* bajo Licencia GNU. Ha sido desarrollada desde el año 2001 cuando fue creada por Russel Smith. Actualmente su última versión es la 0.11, posee amplia bibliografía online, así como una comunidad activa que ofrece ayuda en cualquiera de los temas que se requieran. Con ODE vienen una serie de ejemplos implementados que ayudan a comprender mejor su modo de uso, las funcionalidades y potencialidades que tiene. Ha sido desarrollada en C pero posee una interfaz en C++. Es Multiplataforma, lo cual la hace muy flexible.

Características	Descripción
Primitivas(*Collision Primitives*)	Box, Sphere, Cylinder, Plane, Ray and Trimesh (malla triangular)
Tipos de *Joints*(*Joint Types*)	Ball-Socket, Hinge, Slider (prismatic), hinge-2, Fixed, Angular motor, Universal

Otros	Ofrece espacios de colisión (*collision spaces*) donde las colisiones pueden ser organizadas jerárquicamente. Objetos compuestos: objetos creados con mayor complejidad y compuestos por otros pueden ser tratados como un solo cuerpo.
Plataformas	Windows, Linux and MacOS.

Tabla 1 Descripción de algunas de las características de ODE(Tomada de [26])

Figura 12 Tipos de *joint* utilizados en ODE

ODE es usada para la simulación de estructuras de cuerpos rígidos articulados. Es diseñada para ser usada en simulación interactiva o de tiempo real. Asimismo tiene incluido un sistema de detección de colisiones, lo cual puedes ignorar y hacer tus propias colisiones. Dicho sistema hace una rápida identificación de objetos potencialmente colisionando a través del concepto de "espacios".[29] En ODE las ecuaciones de movimiento se resuelven mediante un modelo de multiplicadores de Lagrange. Se emplea un modelo de contacto y fricción de Coulomb el cual se resuelve por medio de un método de Dantzing LCP.

Este motor de alto rendimiento es útil para simulación de vehículos, objetos en entornos virtuales y criaturas virtuales. Es muy usada en videojuegos, herramientas 3D y de simulación, como son BloodRayne 2, Call of Juarez, Resident Evil 4, Player Project, Webots, Rápido y Curioso (Polo de Realidad Virtual de la UCI), entre otros. La *SceneToolKit* ha desarrollado un módulo para la simulación física de objetos rígidos y articulados, como pueden ser los carros, postes, terrenos, basados en este motor físico.

NVIDIA PhysX

PhysX ha revolucionado la simulación física en videojuegos y demás aplicaciones gráficas. La PPU[4] (Unidad de Procesamientos de Física) PhysX es un chip y un kit de desarrollo diseñados para llevar a cabo cálculos físicos muy complejos. Conocido anteriormente como la SDK de NovodeX, fue originalmente diseñada por AGEIA y tras la adquisición de AGEIA, es actualmente desarrollado por NVIDIA e integrado en sus chips gráficos más recientes.[30] NVIDIA PhysX es un potente motor que permite realizar en tiempo real los cálculos de física de los juegos de PC y consola más avanzados. El software de PhysX ha sido adoptado para la creación de más de 150 juegos, lo emplean más de 10.000 desarrolladores de todo el mundo y puede utilizarse para Playstation 3, Xbox 360, Wii y PC. PhysX está optimizado para acelerar por hardware el cálculo de la física a través de numerosos procesadores paralelos de alta capacidad.

Características	Descripción
Primitivas(*Collision Primitives*)	Box, Sphere, Capsule, Plane, Convex y non-convex mesh
Tipos de *Joints*(*Joint Types*)	Ball-Socket (Spherical), Hinge (Revolute), Slider (prismatic), Fixed, 6 DOF joints con motores, Springs, proyección y joint con límites. Los *joints* pueden ser también frágiles.
Otros	Posee funcionalidades para el manejo avanzado de *ragdoll*, con *joints*, creación y edición de materiales y modelo de fricción para complementar las colisiones con la superficie. Soporte para detección de colisiones continuas para objetos de rápidos movimientos. Creación y simulación de fluidos volumétricos que permiten la simulación de líquidos y gases usando un sistema de partículas y emisoras. Simulación de tela y ropa. Permite

[4] Una Unidad de Procesamiento de Física (PPU por sus siglas en inglés) es un procesador especialmente diseñado para llevar a cabo cálculos físicos en un entorno 3D de videojuego. Procesos tales como partículas o humo son ahora calculadas y desarrolladas, no como hasta ahora que sólo eran animaciones.

	también la simulación de objetos volumétricos deformables. Asimismo permite la simulación de campos volumétricos de fuerza que son objetos similares a los actores y que afectan a las telas o ropas, cuerpos blandos, fluidos y rígidos, que entren dentro de su área de influencia. Estos campos de fuerza permiten implementar ráfagas de vientos, zonas anti-gravitacionales, entre otras. Controlador de Personaje que es principalmente usado para control de jugador en primera y tercera persona que no hagan uso de física de cuerpos rígidos[31]. Tiene soporte también para simular sistema de partículas.
Plataformas	Playstation 3, XBox 360, PC, Wii, Linux

Tabla 2 Características principales de PhysX

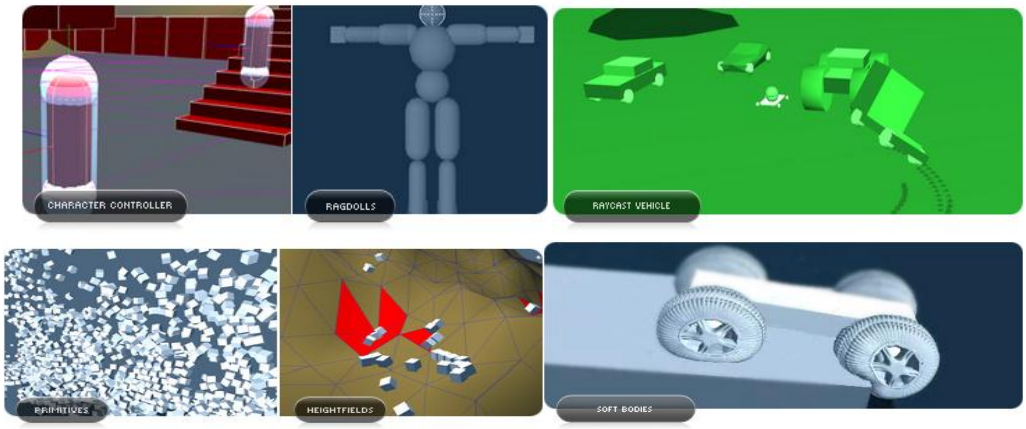

Figura 13 Potencialidades de PhysX NVIDIA

PhysX está sujeta a licencia propietaria, permitiendo utilizar PhysX SDK[5] para desarrollar aplicaciones gráficas, pero cumpliendo una serie de condiciones para su

[5] Un software development kit (SDK) o kit de desarrollo de software es generalmente un conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones para un sistema concreto, por ejemplo ciertos paquetes de software, frameworks, plataformas de hardware, ordenadores, videoconsolas, sistemas operativos, etc.

uso comercial. Para más información [32]. Debido a las numerosas funcionalidades y facilidades que brinda PhysX para acercar las simulaciones físicas a la realidad, las grandes empresas fabricantes de consolas de juegos así como las desarrolladoras de videojuegos han comprado las licencias de tarjetas aceleradoras PhysX y PhysX SDK respectivamente. Así se puede observar, como se menciona anteriormente, a PlayStation3 y XBox 360, que son de las consolas de juegos más usadas, que tienen una tarjeta de aceleración PhysX, como también Batman Arkham Asylum, Darkest of Days y Borderlands, que fueron hechos utilizando PhysX.

Bullet Physics Library

Bullet es un motor físico, *open source*, multi-hilo, de detección de colisiones 3D, y que maneja cuerpos rígidos y blandos, para juegos y efectos especiales. Es publicada bajo licencia *zlib*[6]. Su autor principal es Erwin Coumans, el cual trabajó previamente para el proyecto Havok [33]. La última versión estable es la 2.75 puesta a disposición el 10/9/2009.

Características	Descripción
Primitivas(*Collision Primitives*)	sphere, box, cylinder, cone, convex hull usando GJK[7], non-convex y triangle mesh (malla triangular)
Tipos de *Joints*(*Joint Types*)	Point to Point Constraint, Hinge Constraint, Slider Constraint, Cone Twist Constraint, Generic 6 Dof Constraint
Otros	Presenta detección de colisiones continua y discreta. Posee características de cuerpos blandos con soporte para tela, ropa y cuerpos deformables, plugins para *Houdini*, *Maya*, integrado en *Cinema 4D* y *Blender* e importa contenido físico de

[6] Ha sido aprobada por la Free Software Foundation (FSF) como una licencia de software libre, y por la Iniciativa Open Source (OSI) como una licencia de fuente abierta. Es compatible con la GNU General Public License.

[7] El algoritmo de distancia Gilbert–Johnson–Keerthi es un método de determinación de la distancia mínima entre dos conjuntos convexos. A diferencia de otros algoritmos de distancia, no requiere que la geometría sea almacena en un formato específico, sino que se basa exclusivamente en una función de apoyo para iterativamente generar los simplexes más cercanos a la respuesta correcta usando la suma Minkowski de dos figuras convexas.

	COLLADA 1.4. Rápido y estable solucionador de restricciones dinámicas para cuerpos rígidos. Posee también dinámica de vehículos, controlador de personajes y restricción de giro de cono para *ragdoll*
Plataformas	PLAYSTATION 3, XBox 360, Wii, PC, Linux, Mac OSX and iPhone

Tabla 3 Características principales de Bullet

Bullet autogenera archivos de proyectos MSVC[8] así como cuenta con un sistema de construcción (*build system*) JAM[9]. Cuenta con un manual, del cual su versión más avanzada es de la SDK 2.74. Presenta un diseño modular y personalizable que te permite utilizar todas o tan solo las partes del motor que necesites, lo que cual la hace muy flexible para los desarrolladores (Figura 14).

Dentro de sus aportes resalta una funcionalidad llamada *MotionState* (Estado de Movimiento), la cual hace el trabajo más fácil, ya que te brinda la posibilidad de ahorrarte el trabajo de tener que actualizar el estado mundial de los objetos en la escena como parte del renderizado del motor gráfico. Esto trae múltiples beneficios entre los que resalta que el cálculo del estado de los objetos nada más que se realiza para aquellos que van a sufrir cambios en su estado, es decir, para aquellos que no sufrirán cambios no se le calcula su estado. Asimismo Bullet maneja la interpolación de movimientos de los cuerpos a través de esta funcionalidad.

Debido a estas características y facilidad de uso ha sido utilizada en varios productos y proyectos tanto comerciales como no comerciales. Entre los primeros se puede destacar *Madagascar Kartz* producido por *Activision*, *Free Realms* por *Sony Online*

[8] Entorno de desarrollo integrado (IDE) para lenguajes de programación C, C++ y C++/CLI. Esta especialmente diseñado para el desarrollo y depuración de código escrito para las API's de Microsoft Windows, DirectX y la tecnología Microsoft .NET Framework.

[9] Es un sistema de construcción *open source* desarrollado por Christopher Seiwald of Perforce Software. Puede ser usada como sustitución del *make*. Su principal característica su habilidad para expresar la construcción de patrones en un lenguaje imperativo el cual soporta estructuras *namespaces*. Jam se puede ejecutar en Unix, OpenVMS, Windows NT, Mac OS y BeOS.

Entertainment, *Grand Theft Auto 4* por *Rockstar Games*, *Trials HD by RedLynx*; y productos *"open source"* como *Blender 3D* en un modelador 3D que usa Bullet para animaciones y potencialidades de su motor de juego interno llamado *Game Blender*, Ogre que integra Bullet a través de *OgreBullet*, *OpenSceneGraph* a través del *osgBullet plugin*, entre otros. [33]

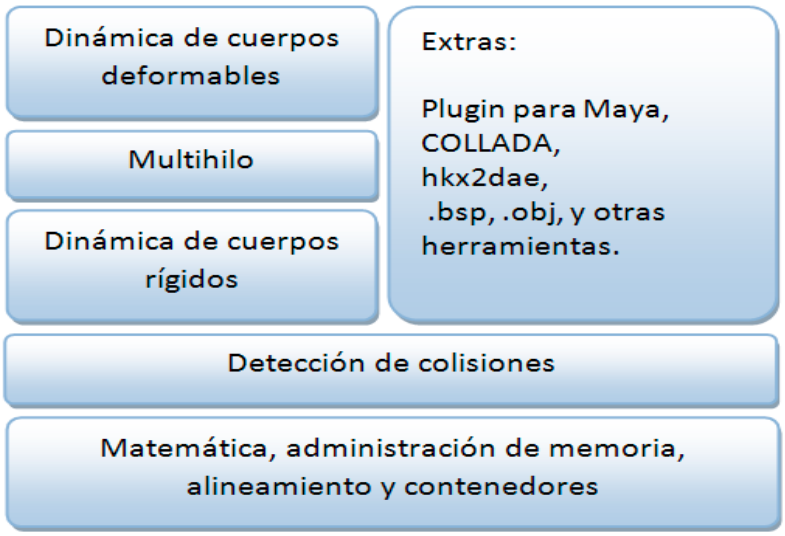

Figura 14 Estructura Modular de Bullet

Conclusiones

En este capítulo se ha hecho un estudio del objeto de la investigación, donde se han enunciado conceptos fuertemente relacionados con las animaciones en tiempo real, algunas de las técnicas utilizadas para lograr las mismas con altos niveles de realismo, ya sea con animaciones predefinidas como mediante la aplicación de leyes físicas.

En el estudio de las técnicas se centra la investigación en las animaciones de personajes utilizando técnicas cinemáticas a partir de animaciones predefinidas y dinámicas partir de la aplicación de leyes físicas, los conceptos que engloban cada una de ellas; así como también, y más fundamental, el control dinámico sobre las animaciones prediseñadas, utilizando los parámetros que aportan estas para lograr el efecto deseado. Finalizando el capítulo se ha hecho una explicación de los principales motores físicos utilizados en aplicaciones de realidad virtual, analizando como simulan la dinámica de cuerpos rígidos.

Capítulo 2: Soluciones Técnicas

Introducción

En el desarrollo de aplicaciones gráficas es de gran importancia lograr el mayor realismo con el mejor rendimiento posible. Para ello ha sido de gran utilidad la combinación de las animaciones prediseñadas y la simulación física.

En el presente Capítulo se proponen soluciones técnicas para lograr satisfacer objetivamente este trabajo. Asimismo se propondrá una solución para combinar las animaciones prediseñadas y la simulación física a fin de proporcionar un control dinámico sobre los personajes.

2.1 Combinación de animaciones prediseñadas y simulación física

A partir de los conceptos y técnicas estudiadas en la fundamentación teórica desarrollada en el presente trabajo se proponen para solución a su objetivo el desarrollo de un sistema que controle dinámicamente un personaje en tiempo real. Se proponen realizar este sistema mediante la modelación e implementación de funcionalidades que se enuncian en este capítulo y cuyo ciclo se muestra en la figura 15.

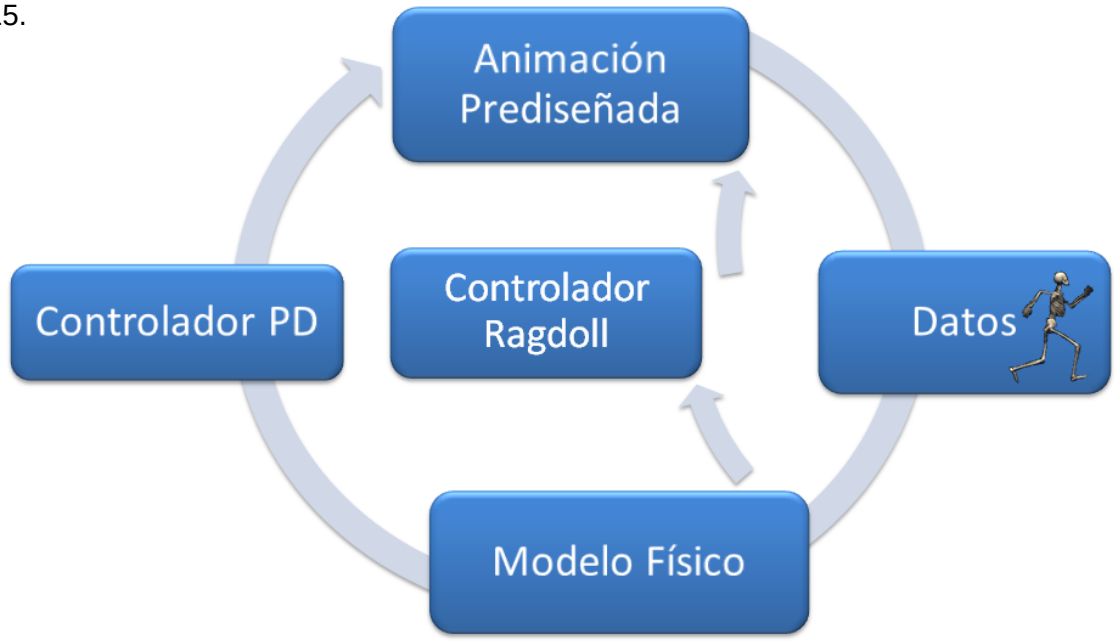

Figura 15 Propuesta del Sistema

2.2 Manejo de Animaciones prediseñadas

Después de un análisis de los conceptos abordados en el epígrafe 1.1, se propone la utilización de las animaciones prediseñadas haciendo uso de la animación por esqueleto. Para lograrlo se hace uso del módulo de animación que presenta la SceneToolKit, la cual nos brinda un grupo de funcionalidades para el manejo de las animaciones dentro de la escena.

Este módulo de la SceneToolKit nos permite la animación a través de trayectorias, la transición de animaciones, la interpolación de animaciones, entre otras características. Pero lo más útil para el objetivo de este trabajo es la información que por cada fotograma nos puede brindar este módulo. En este se maneja la información de los huesos del personaje por cada fotograma, como pueden ser la posición y orientación relativa y global de cada hueso, la jerarquía del esqueleto, entre otras. Esta información que proporciona el módulo de animación es la que utiliza el módulo físico para calcular el próximo estado en el que estará el personaje y que después envía al módulo de animación para su visualización.

Por tanto para la solución que se propone se hará uso extensivo de la información que proporcione el esqueleto y su animación para ser utilizada por el módulo físico en la simulación dinámica del personaje.

2.3 Animación Física del personaje

Con la información obtenida del módulo de animación de la STK, se procede a modelar físicamente el personaje. Para ello se hará uso esencial de un Motor físico para la simulación de cuerpos rígidos y de alguna(s) técnica(s) dinámica(s) para la construcción de un modelo físico que calcule los valores y datos suficientes para lograr la animación.

2.3.1 Motor Físico

El motor seleccionado es *Bullet Physic Library* debido al tratamiento con los cuerpos rígidos y su alto nivel de óptimo rendimiento frente a otros motores reconocidos de código abierto como lo es ODE. El motor de *Bullet* produjo los mejores resultados en general, dentro de los motores de código abierto, en [34] donde se desarrolló una evaluación de los principales motores físicos de simulación en tiempo real, llegando incluso a tener mejores resultados que otros motores privativos o comercializables.

Además de ello en la herramienta SceneToolKit se ha desarrollado un módulo físico basado en *Bullet* para el tratamiento físico de los objetos dentro de la escena. La utilización de este motor tiene como objetivo el manejo físico de los huesos del esqueleto como cuerpos rígidos articulados en la escena.

2.3.2 Comportamiento *Ragdoll*

El comportamiento por *Ragdoll* es una de las técnicas dinámicas que son utilizadas ampliamente en las aplicaciones gráficas. Para su desarrollo en la solución propuesta se obtendrá los valores de los datos que proporcione el módulo de animación (posición y rotación de cada hueso, la jerarquía de los huesos) con el cual se construirá un modelo físico. Este modelo físico configura físicamente al personaje dándole propiedades físicas a cada uno de los huesos, como es la masa, la gravedad, la velocidad angular, entre otras. Después de haber simulado físicamente al personaje, dicho modelo le facilita los datos necesarios al módulo de animación para que el resultado se pueda visualizar gráficamente. El desarrollo de esta técnica permitirá al personaje la simulación de una caída al suelo como si hubiese muerto, es decir, cae al suelo como un “muñeco de trapo”.

2.3.3 Controladores PD

Con el objetivo de que el personaje interactúe físicamente dentro del entorno dinámico es muy común el uso de los controladores PD. Para la solución propuesta se utilizará el uso particular de uso de Controlador PD descrito en el epígrafe 1.4, donde se calculan los coeficientes k_s y k_d de manera analítica a fin de poder calcular el torque necesario por cada joint para que el personaje pueda llegar hasta la posición deseada en instante de tiempo dado.

El proceso final de cálculo del torque consiste de dos pasadas a través de la jerarquía de huesos. La primera pasada de la inercia compuesta que recursivamente descendiendo por la jerarquía de huesos desde el efector final hasta la raíz, visitando cada hueso para acumular la inercia para el estado actual, $D(q)$. La segunda pasada es para el cálculo del torque una vez por cada hueso acumulando la aceleración angular a_i siguiendo la jerarquía desde la raíz hasta el efector final. El algoritmo quedaría definido de la siguiente forma:

ControlTorque $(q, \gamma) \rightarrow \tau$

{

Para todos huesos *l*, descendiendo **hacer**

Si padre(l) existe **entonces**

$D_l \leftarrow D_l$ + *transformInertia*($D_{padre(l)}, l$)

Fin

j = *outboardJointOfLink*(l)

$a_j \leftarrow 0$

fin

Para Todos *joints* i, ascendiendo **hacer**

$l \leftarrow$ *outboardLinkOfJoint* (i)

$(k_s, k_d) \leftarrow \psi(D_l, \gamma_i)$

$m_i \leftarrow$ momentoInercia(D_l, i)

$u_i \leftarrow$ TorquePadre(i, a_i)

$\tau_i \leftarrow$ PD(q_l, m_i, k_{si}, k_{di}, γ_i, u_i)

$\tau_i \leftarrow$ min($\tau_i, \tau_{\max}$)

Para todos joints c en Hijos(l) **hacer**

$a_i \leftarrow a_i$ + *transform*($\tau_i / m_i, c$)

fin

fin

return τ

}

Conclusiones

En este capítulo se ha explicado la solución propuesta para cumplir con los objetivos trazados en el comienzo de la investigación. Esta solución será capaz de manipular adecuadamente animaciones mediante la combinación de animaciones prediseñadas y control dinámico de estas con el fin de obtener animaciones de alto realismo para que los personajes puedan comportarse lo más natural posible en tiempo real. Esta solución permite que sea más flexible la construcción de aplicaciones como juegos y simuladores profesionales donde los personajes sean figuras representadas constantemente en las escenas y tengan diversos comportamientos.

Con el capítulo presentado quedan trazadas las guías que servirán para realizar las siguientes fases del trabajo, teniendo las bases técnicas por las cuales se regirá el método a realizar, es decir, el control dinámico.

Capítulo 3: Descripción de la Solución Propuesta

Introducción

Con el objetivo de obtener una descripción de la solución propuesta en su forma conceptual en el presente capítulo se abordarán los conceptos necesarios para el desarrollo de la solución. Asimismo se describen las restricciones a considerar para modelar dicha solución así como las potencialidades que tendrá el control dinámico de personajes descritas forma de caso de usos.

3.1 Reglas del Negocio

Con el propósito de definir las condiciones que debe satisfacer el correcto funcionamiento del Control Dinámico se establecen como reglas del negocio:

- Los personajes deben ser diseñados según el sistema de coordenadas de la mano derecha, debido al nuevo sistema de coordenadas de la STK (Sistema de Coordenadas de OpenGL).
- Los huesos deben de estar jerarquizados correctamente, tanto en el modelo de diseño como en el modelo físico.
- Las relaciones entre los huesos y los vértices de la malla deben de estar guardados en el fichero PWX que es el utilizado por la STK.
- La información del estado de cada hueso por cada fotograma deben estar guardados en el fichero ANX que es el utilizado por la STK.
- Cada modelo físico de personaje (“personaje físico”) tiene que estar asociado un modelo de diseño, por lo tanto si no existe un modelo diseñado de animación de personaje no existirá uno físico.
- Para la construcción del modelo físico tiene que estar creado el mundo físico mediante la utilización del motor físico Bullet Physic.

- Los nombres de los huesos, guardados en el fichero PWX, deben contener información de qué tipo de *joint* lo unirá a su padre.

3.2 Modelo de Dominio

El modelo de dominio mostrado a continuación representa un acercamiento a la solución propuesta donde se modelan los principales conceptos así como sus relaciones.

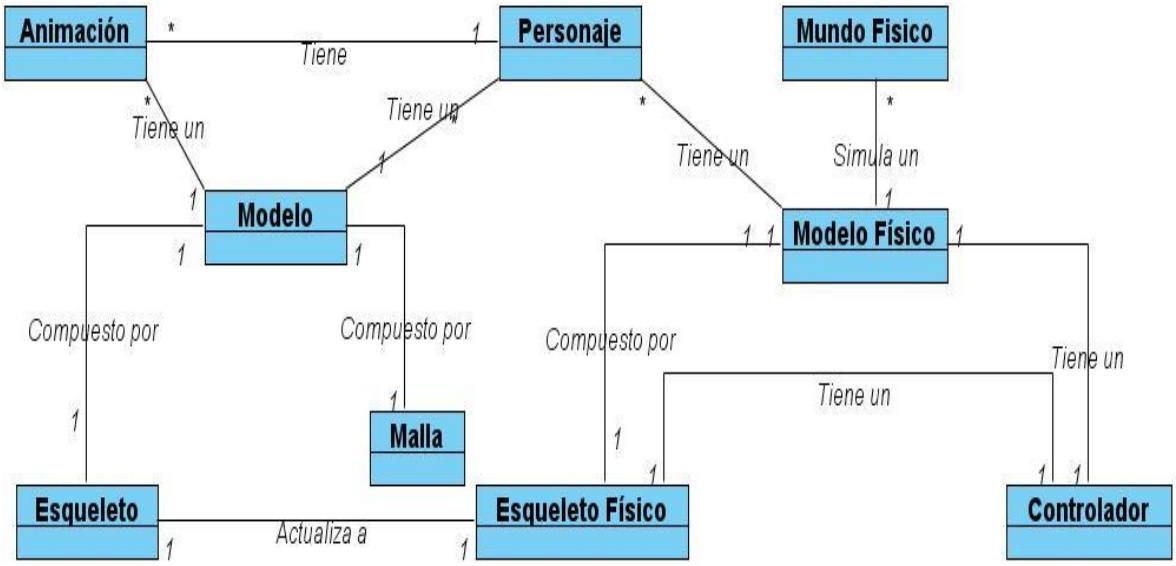

Figura 16 Modelo de dominio

3.3 Glosario de Términos

Es muy importante conocer el significado de los conceptos enunciados y que se relacionan en el modelo anterior para lo cual a continuación se especifican los mismos con el objetivo de complementar la comprensión de dicho modelo.

Personaje: actor de la escena de un mundo de realidad virtual, que soporta acciones (como un tipo de comportamiento), y que tienen entre sus atributos, cualidades físicas y emocionales que serán usadas a la hora de ejecutar las acciones, así como determinados roles.

Modelo: Prototipo para la animación que relaciona la malla con el esqueleto.

Malla: Forma de representar un modelo a partir de polígonos. Colección de vértices, aristas y polígonos conectados de forma que cada arista es compartida como máximo por dos polígonos.

Esqueleto: Estructura jerárquica que agrupa los huesos de un modelo.

Hueso: Estructura con influencia sobre una determinada cantidad de vértices de la malla determinando en estos la posición a partir de la posición y rotación del hueso.

Animación: Secuencia de transformaciones de cada uno de los huesos del esqueleto en una cantidad de fotogramas determinados a una velocidad dada.

Modelo Físico: Prototipo para el manejo físico del esqueleto de una animación en sincronización con un Modelo.

Mundo Físico: Es un contenedor para todos aquellos elementos físicos que se quieran simular, es decir, que se quiera pertenezcan al Mundo Físico. Todo elemento que se simule tiene que ser añadido al Mundo y este a su vez tiene control sobre ellos. Todos los elementos en un Mundo Físico existen el mismo instante de tiempo.

Esqueleto Físico: Estructura jerárquica que agrupa los huesos de un modelo físico.

Hueso Físico: Estructura análoga al hueso y que contiene un cuerpo físico para manejar su estado geométrico.

***Joint* (Unión):** Es la relación existente entre dos cuerpos tales que cada uno de ellos pueden tener solo ciertas posiciones y orientaciones relativas al otro.

Cuerpo Físico: Estructura que contiene y describe las propiedades físicas de un objeto en el “mundo real”. Masa, centro de masa, velocidad y aceleración lineal y angular, Inercia, Tensor de Inercia, son algunas de estas propiedades que determinan el estado y las transformaciones que sufre un cuerpo dentro del Mundo Físico.

Forma de Colisión: Son los elementos fundamentales en el sistema de colisión. Puede representar a un cuerpo físico (caja, esfera, capsula, etc.) o puede representar un grupo de otras formas de colisión. Es el elemento físico que contienen los cuerpos físicos en el Mundo Físico que permite las colisiones entre ellos.

Controlador: Es la estructura que se encarga de ejecutar las acciones definidas para el elemento asignado a controlar. El Controlador es el encargado de actualizar el estado de dicho elemento.

Controlador PD: Es el controlador que intenta llevar a cabo las acciones determinadas por una animación ante las perturbaciones que pueda sufrir en un entorno dinámico, intentado realizar la animación lo más real posible.

Controlador *Ragdoll*: Es el controlador que simula la carencia de vida de un personaje en un entorno dinámico, como puede ser la muerte. El personaje cae como si fuera un muñeco de trapo.

3.4 Captura de Requisitos

El sistema que se quiere obtener tiene que cumplir una serie de condiciones o potencialidades para solucionar las deficiencias para el control dinámico de los personajes que presenta la herramienta a la cual se integrará. Estas se definen en los siguientes requisitos funcionales y no funcionales.

3.4.1 Requisitos Funcionales

Dentro de los requisitos funcionales que debe cumplir el sistema propuesto encontramos:

1. Crear hueso físico
2. Modificar hueso físico
3. Actualizar hueso físico
    3.1. Actualizar propiedades físicas
4. Destruir hueso físico
5. Crear esqueleto físico
    5.1. Definir propiedades físicas
6. Modificar esqueleto físico
7. Actualizar esqueleto físico

7.1. Actualizar propiedades físicas

8. Destruir esqueleto físico
9. Crear articulaciones físicas
    9.1. Definir propiedades físicas
10. Modificar articulaciones físicas
11. Actualizar articulaciones físicas
    11.1. Obtener Tensor de Inercia Acumulada
    11.2. Obtener momento de inercia
    11.3. Obtener aceleración angular acumulada
    11.4. Aplicar torque necesario para alcanzar el estado deseado
12. Destruir articulaciones físicas
13. Seleccionar técnica dinámica
14. Cambiar de técnica dinámica
15. Crear Controlador Ragdoll
    15.1. Asignar esqueleto físico
16. Ejecutar Controlador Ragdoll
17. Actualizar Controlador Ragdoll
18. Destruir controlador Ragdoll
19. Crear Controlador PD
    19.1. Definir parámetros
    19.2. Modificar parámetros
    19.3. Asignar esqueleto físico

20. Ejecutar Controlador PD

21. Actualizar Controlador PD

    21.1. Actualizar parámetros

    21.2. Obtener Torque para aplicar a las articulaciones

22. Destruir Controlador PD

3.4.2 Requisitos No Funcionales

Entre los requisitos no funcionales que debe tener el sistema a partir de la investigación realizada, la modelación e implementación se encuentran los siguientes:

Rendimiento:

- Como aplicación de tiempo real, debe tener alto grado de velocidad de procesamiento o cálculo, tiempo de respuesta y de recuperación, y disponibilidad.

Soporte:

- El sistema debe ser multiplataforma, de tal forma que se pueda utilizar tanto en Windows como en Linux.

Usabilidad:

- Los usuarios del sistema deberán ser desarrolladores con conocimientos básicos de programación gráfica. Dicho sistema debe ser desarrollado objetivamente con el propósito de que el usuario solo determine el “qué” quiere hacer y no tenga que penar en “cómo” hacerlo.
- La biblioteca está concebida para ser reutilizable por aplicaciones finales.
- El producto debe estar concebido para ser acoplado a la STK, de tal manera que se use en caso de que el desarrollador lo necesite y no como parte indispensable de ella.

Legales:

- Todas las bibliotecas que se utilizan son libres bajo licencias compatibles con la *GPL*.

Software:

- Sistema operativo Windows y GNU/Linux
- *Bullet SDK*
- Microsoft Visual Studio.Net 2005 y CodeBlock 8.02

Hardware:

- Compatibilidad con tarjetas gráficas de la familia NVIDIA.
- PC Pentium 4, 2.4 GHz, 512 MB de memoria RAM.

Diseño e implementación:

- Debe implementada en el Leguaje C++ estándar.
- Se utilizará la biblioteca *SceneToolKit* como Motor gráfico para el dibujado de los elementos en la escena. Este motor utiliza las dos principales *APIs* gráficas: *OpenGL* y *Direct3D*.
- Se regirá por la filosofía de Programación Orientada a Objetos.

3.5 Definición de los Casos de Uso del Sistema

A partir de las funcionalidades que deben ser implementadas en el sistema para lograr el control dinámico de personajes, definidas en el epígrafe 3.4.1, se hace el agrupamiento de las mismas y se llega como resultado a los casos de uso del sistema representados en la figura 17.

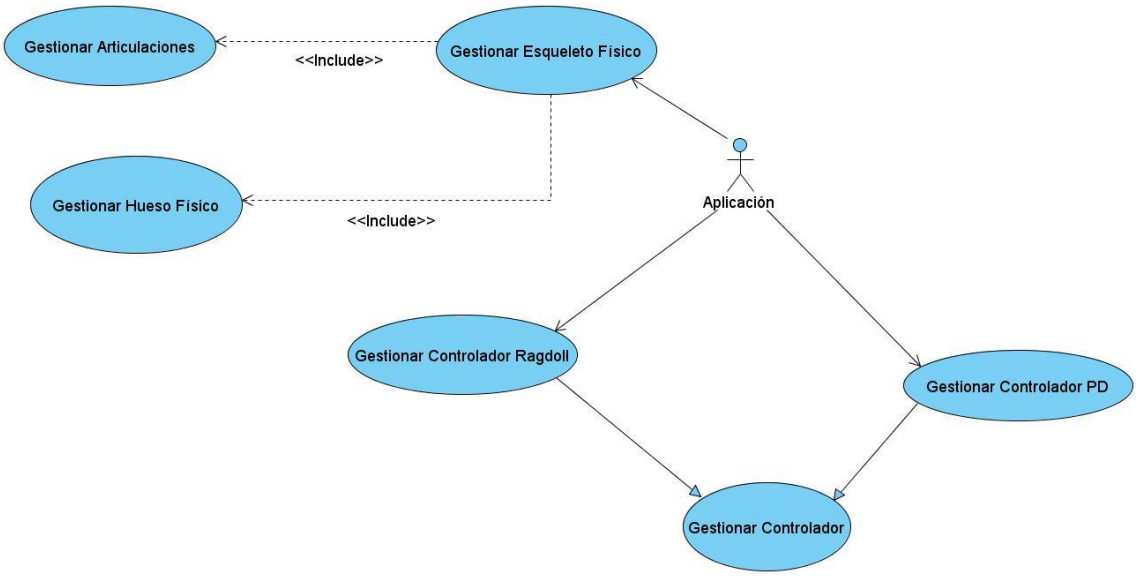

Figura 17. Diagrama de Casos de Uso del Sistema

3.5.1 Definición de actores del sistema

Actores	Justificación
Aplicación Final	Es quién se beneficia con las funcionalidades que brinda el control dinámico de personajes e interactúa con todo el sistema.

3.5.2 Descripción de los Casos de Uso

3.5.2.1 CUS Gestionar Esqueleto Físico

Nombre del caso de uso	Gestionar Esqueleto Físico
Actores	Aplicación final
Propósito	Permitir la creación, modificación, actualización y eliminación de un esqueleto Físico
Resumen: El caso de uso inicia cuando el actor decide crear, modificar, actualizar o eliminar un esqueleto.	

Referencias	RF-5, RF-6, RF-7, RF-8
Curso normal de los eventos	
Acción del actor	Respuesta del sistema
1- Si el actor desea crear un esqueleto físico, ir a la sección “crear”. a) Si el actor desea modificar un esqueleto físico ir a la sección modificar. b) Si el actor desea actualizar un esqueleto físico, ir a la sección “actualizar”. c) Si el actor desea eliminar un esqueleto físico, ir a la sección “eliminar”.	
Sección: “crear”	
Acción del actor	Respuesta del sistema
1- Llama al constructor de Esqueleto Físico.	
	1.1- Verificar que exista el esqueleto creado por diseño. 1.2- Por cada hueso del esqueleto por diseño se invoca al caso de uso “Gestionar Hueso Físico”. 1.3- Se invoca al caso de uso “Gestionar Articulaciones” 1.4- Se Crea el Esqueleto físico con propiedades de amortiguación y rigidez.

Sección: “modificar”	
Acción del actor	Respuesta del sistema
1- Cambia algún parámetro del esqueleto físico.	
	1.1- Si el parámetro modifica una articulación se invoca al caso de uso “Gestionar Articulaciones”. 1.2- Si el parámetro modifica un hueso físico se invoca al caso de uso “Gestionar Hueso Físico”. 1.3- Se modifica el parámetro especificado del esqueleto físico.
Sección: “actualizar”	
Acción del actor	Respuesta del sistema
1- Llama a la función de actualizar esqueleto físico.	
	1.1- Se actualiza las articulaciones. Se invoca al caso de uso “Gestionar Articulaciones”. 1.2- Se actualiza los huesos físicos. Se invoca al caso de uso “Gestionar Hueso Físico”. 1.3- Se procede a actualizar el esqueleto físico.
Sección: “eliminar”	
Acción del actor	Respuesta del sistema
1- Llama al destructor del Esqueleto Físico.	
	1.1- Se invoca al caso de uso “Gestionar Hueso Físico”.

	1.2- Se invoca al caso de uso "Gestionar Articulaciones". 1.3- Se procede a eliminar el Esqueleto Físico.
Flujo alternativo de los eventos	
	1.1- Se lanza un mensaje de error mostrando que el esqueleto por diseño no existe.

3.5.2.2 CUS Gestionar Articulaciones

Nombre del caso de uso	Gestionar Articulaciones
Actores	CUS Gestionar Esqueleto Físico
Propósito	Permitir la creación, modificación, actualización y eliminación de un esqueleto Físico.
Resumen: El caso de uso inicia cuando el actor decide crear, modificar, actualizar o eliminar una articulación.	
Referencias	RF-9, RF-10, RF-11, RF-12
Curso normal de los eventos	
Acción del actor	Respuesta del sistema
1- Si el actor desea crear una articulación, ir a la sección "crear". a) Si el actor desea modificar una articulación, ir a la sección modificar. b) Si el actor desea actualizar una articulación, ir a la sección "actualizar".	

c) Si el actor desea eliminar una articulación, ir a la sección "eliminar".	
Sección: "crear"	
Acción del actor	Respuesta del sistema
1- Llama a la función crear articulaciones.	
	1.1- Selecciona dos huesos de la lista de huesos físicos para crear una articulación. 1.2- Llama al constructor para crear una articulación. 1.3- Se crea la articulación.
Sección: "modificar"	
Acción del actor	Respuesta del sistema
1- Cambia parámetro de articulación	
	1.1- Se modifica el parámetro especificado.
Sección: "actualizar"	
Acción del actor	Respuesta del sistema
1- Llama a la función de actualizar articulación.	
	1.1- Se procede a calcular el Tensor de Inercia Acumulado. 1.2- Se procede a calcular el momento de Inercia acumulado.

Sección: “eliminar”	
Acción del actor	Respuesta del sistema
1- Llama al destructor de la Articulación.	
	1.4- Se procede a eliminar la articulación.

3.5.2.3 CUS Gestionar Hueso Físico

Nombre del caso de uso	CUS Gestionar Esqueleto Físico
Actores	CUS Gestionar Esqueleto Físico
Propósito	Permitir la creación, modificación, actualización y eliminación de un hueso físico.
Resumen: El caso de uso inicia cuando el actor decide crear, modificar, actualizar o eliminar un hueso físico.	
Referencias	RF-1, RF-2, RF-3, RF-4
Curso normal de los eventos	
Acción del actor	Respuesta del sistema
1- Si el actor desea crear un hueso físico, ir a la sección “crear”. a) Si el actor desea modificar un hueso físico, ir a la sección modificar. b) Si el actor desea actualizar un hueso físico, ir a la sección “actualizar”. c) Si el actor desea eliminar un hueso físico, ir a la sección	

"eliminar".	
Sección: "crear"	
Acción del actor	Respuesta del sistema
1- Selecciona un hueso del esqueleto por diseño. 2- Llama al constructor de Hueso físico.	
	2.1- Crea el hueso físico de acuerdo a los parámetros dados.
Sección: "modificar"	
Acción del actor	Respuesta del sistema
2- Cambia parámetro de hueso físico	
	1.2- Se modifica el parámetro especificado.
Sección: "actualizar"	
Acción del actor	Respuesta del sistema
2- Llama a la función de actualizar hueso físico.	
	1.3- Actualiza estado geométrico del hueso físico(Posición y Orientación).
Sección: "eliminar"	
Acción del actor	Respuesta del sistema
1- Llama al destructor del hueso físico.	

	1.1- Se procede a eliminar el hueso físico.

3.5.2.3 CUS Gestionar Controlador

Nombre del caso de uso	Gestionar Controlador
Actores	Aplicación final
Propósito	Permitir la creación, actualización, y eliminación de un controlador. Permite cambiar el tipo de controlador.
Resumen: El caso de uso inicia cuando el actor decide crear, actualizar, eliminar o cambiar un controlador.	
Referencias	RF-13, RF-14
Curso normal de los eventos	
Acción del actor	Respuesta del sistema
1- Si el actor desea crear un controlador, ir a la sección “crear”. a) Si el actor desea modificar un controlador, ir a la sección modificar. b) Si el actor desea actualizar un controlador, ir a la sección “actualizar”. c) Si el actor desea cambiar de controlador ir a la sección “cambiar controlador”. d) Si el actor desea eliminar un controlador, ir a la sección “eliminar”.	

Sección: “crear”	
Acción del actor	Respuesta del sistema
1- Selecciona el tipo de controlador por crear.	
	1.4- Si el controlador es *Ragdoll* se invoca el caso de uso Gestionar Controlador *Ragdoll*. 1.5- Si el controlador es PD se invoca el caso de uso Gestionar Controlador PD. 1.6- Se Crea el Controlador.
Sección: “modificar”	
Acción del actor	Respuesta del sistema
3- Cambia parámetro del Controlador	
	1.3- Si el controlador es *Ragdoll* se invoca el caso de uso “Gestionar Controlador *Ragdoll*”. 1.4- Si el controlador es PD se invoca el caso de uso “Gestionar Controlador PD”.
Sección: “actualizar”	
Acción del actor	Respuesta del sistema
3- Llama a la función de actualizar controlador.	
	1.4- Si el controlador es de tipo *Ragdoll* se invoca el CU “Gestionar Controlador Ragdoll” 1.5- Si el controlador es de tipo PD se invoca el CU “Gestionar Controlador PD”.

Sección: “Cambiar Controlador”	
Acción del Actor	Respuesta del sistema
1- Solicita cambiar el tipo de controlador	
	1.1- Obtiene los datos actuales del controlador activo. 1.2- Si el controlador es tipo *Ragdoll* se invoca el caso de uso “Gestionar Controlador PD”. 1.3- Si el controlador es de tipo PD” se invoca el caso de uso “Gestionar Controlador Ragdoll”.
Sección: “eliminar”	
Acción del actor	Respuesta del sistema
1- Llama al destructor del Controlador.	
	1.5- Si el controlador es de tipo R*agdoll* se invoca al caso de uso “Gestionar Controlador *Ragdoll*”. 1.6- Si el controlador es de tipo PD se invoca al caso de uso “Gestionar Controlador PD”. 1.7- Se procede a eliminar el controlador.

3.5.2.3 CUS Gestionar Controlador *Ragdoll*

Nombre del caso de uso	Gestionar Controlador *Ragdoll*
Actores	CUS Gestionar Controlador
Propósito	Permitir la creación, actualización y eliminación de un controlador *ragdoll*.
Resumen: El caso de uso inicia cuando el actor decide crear, actualizar o eliminar un	

controlador *ragdoll*.	
Referencias	RF-15, RF-16, RF-17, RF-18
Curso normal de los eventos	
Acción del actor	Respuesta del sistema
1- Si el actor desea crear un controlador *ragdoll*, ir a la sección “crear”. a) Si el actor desea modificar un controlador *ragdoll*, ir a la sección modificar. b) Si el actor desea actualizar un controlador *ragdoll*, ir a la sección “actualizar”. c) Si el actor desea eliminar un controlador, ir a la sección “eliminar”.	
Sección: “crear”	
Acción del actor	Respuesta del sistema
1- Selecciona un esqueleto físico. 2- Llama al constructor de controlador *Ragdoll*.	
	2.1- Crea un controlador *Ragdoll* con el esqueleto especificado.
Sección: “modificar”	
Acción del actor	Respuesta del sistema

1- Cambia parámetro del Controlador *Ragdoll*	
	1.1- Se procede a modificar el parámetro especificado.
Sección: “actualizar”	
Acción del actor	Respuesta del sistema
4- Llama a la función de actualizar del controlador *ragdoll*.	
	1.1- Se obtiene el estado geométrico de los huesos del esqueleto físico. 1.2- Se actualizan los huesos del esqueleto por diseño con los valores obtenidos.
Sección: “eliminar”	
Acción del actor	Respuesta del sistema
1- Llama al destructor del Controlador *Ragdoll*.	
	1.1- Se procede a eliminar el controlador *ragdoll.*

3.5.2.3 CUS Gestionar Controlador PD

Nombre del caso de uso	Gestionar Controlador PD
Actores	CUS Gestionar Controlador
Propósito	Permitir la creación, actualización y eliminación de un controlador PD.
Resumen: El caso de uso inicia cuando el actor decide crear, actualizar o eliminar un controlador PD.	
Referencias	RF-19, RF-20, RF-21, RF-22

Curso normal de los eventos	
Acción del actor	Respuesta del sistema
1- Si el actor desea crear un controlador PD, ir a la sección “crear”. a) Si el actor desea modificar un controlador PD, ir a la sección modificar. b) Si el actor desea actualizar un controlador PD, ir a la sección “actualizar”. c) Si el actor desea eliminar un controlador, ir a la sección “eliminar”.	
Sección: “crear”	
Acción del actor	Respuesta del sistema
1- Selecciona un esqueleto físico. 2- Llama al constructor de controlador PD.	
	2.1- Crea un controlador PD con el esqueleto especificado. 2.2- Define y modifica parámetros de rigidez y amortiguamiento de cada hueso del esqueleto.
Sección: “modificar”	
Acción del actor	Respuesta del sistema
1- Cambia parámetro del	

Controlador PD.	
	1.1- Se procede a modificar el parámetro especificado.
Sección: “actualizar”	
Acción del actor	Respuesta del sistema
1- Llama a la función de actualizar del controlador PD.	
	1.1- Se obtiene los datos de los fotogramas actual y siguiente. 1.2- Se obtiene el momento de inercia y el tensor de inercia acumulados en cada articulación del esqueleto físico. 1.3- Se procede a calcular y aplicar el torque acumulado en cada articulación del esqueleto físico. 1.4- Se procede a actualizar los huesos del esqueleto por diseño.
Sección: “eliminar”	
Acción del actor	Respuesta del sistema
1- Llama al destructor del Controlador PD.	
	1.5- Se procede a eliminar el controlador PD.

Capítulo 4: Diseño e Implementación del Sistema

Introducción

En este capítulo se muestran los diagramas de clases del diseño correspondiente sistema, haciendo énfasis en las clases arquitectónicamente significativas. Se enuncian además los estándares para la nomenclatura utilizados durante el desarrollo.

4.1 Estándares de codificación

Este módulo sigue los estándares del código de la biblioteca a la cual se acoplará (STK). Se respetaron los estándares de codificación de C++ y se programó en idioma inglés, debido a que las palabras son sencillas, no se acentúan y es un idioma universal en el mundo informático.

4.1.1 Nombre de los ficheros fuente

Se nombrarán los ficheros .h y .cpp de la siguiente manera:

STKNameOfUnits.cpp

Se usará **GT** para identificar el nombre de la herramienta.

4.1.2 Constantes

Las constantes se nombrarán con mayúsculas, utilizándose el "_" para separar las palabras: MY_CONST_ZERO = 0;

4.1.3 Tipos de datos

Los tipos se nombrarán siguiendo el siguiente patrón:

4.1.4 Enumerados

enum**E**MyEnum {**ME**_VALUE, **ME**_OTHER_VALUE};

Indicando con "**E**" que es de tipo enumerado. Nótese que las primeras letras de las constantes de enumerados son las iniciales del nombre del enumerado. Véase otro ejemplo:

enum**E**Node**T**ype {**NT**_GEOMETRYNODE,…};

4.1.5 Estructuras

struct**S**MyStruct {…};

Indicando con "**S**" que es una estructura. Las variables miembros de la estructura se nombrarán igual que en las clases, leer más adelante.

4.1.6 Clases

class **C**ClassName;

Indicando con "**C**" que es una clase

4.1.7 Listas e iteradores STL

vector<>**T**Name**List**;

TNameList::iterator**T**Name**ListIter**;

map<>**T**Name**Map**;

TNameMap::iterator**T**Name**MapIter**;

multimap<>**T**Name**MultiMap**;

TNameMultiMap::iterator**T**Name**MultiMapIter**;

4.1.8 Declaración de variables

Los nombres de las variables comenzarán con un identificado del tipo de dato al que correspondan, como se muestra a continuación. En el caso de que sean variables miembros de una clase, se le antepondrá el identificador "" (en minúscula), si son globales se les antepondrá la letra "g", y en caso de ser argumentos de algún método, se les antepondrá el prefijo "arg_".

4.1.9 Tipos simples

bool **b**VarName;

int **i**Name;

unsigned int **ui**Name;

float **f**Name;

char **c**Name;

char* **ac**Name; // arreglo de caracteres

char* **pc**Name; // puntero a un char

char** **aac**Name; // bidimensional

char** **apc**Name; // arreglo de punteros

bool **m_b**MemberVarName; //variable miembro

char **gc**GlobalVarName; //variable global

short **s**Name;

4.1.10 Instancias de tipos creados

EMyEnumerated **e**Name;

SMyStructure **k**Name;

CClassName **k**ObjectName;

CClassName* **pk**Name; //puntero a objeto

CClassName* **ak**Name; //arreglo de objetos

CClassName* **ak**Name; // variable miembro de clase

IMyInterface* **pI**Name; //puntero interfaces

4.1.11 Métodos

En el caso de los métodos, se les antepondrá el identificador del tipo de dato de devolución, y en caso de no tenerlo (void), no se les antepondrá nada. Solamente los constructores y destructores comenzarán con “”.

En el caso de los argumentos se les antepone el prefijo “arg_”

4.1.12 Constructor y destructor

CClassName (bool **arg**_bVarName, float& **arg**_fVarName);

~**C**ClassName ();

4.1.13 Funciones

b***ool bFunction1 (…);***

int* piFunction2 (…);

CClassName* pkFunction3 (…);

4.1.14 Procedimientos

void Location(…);

4.1.15 Métodos de acceso a miembros

Los métodos de acceso a los miembros de las clases no se nombrarán “Gets” y “Sets”, sino como los demás métodos, pero **con el nombre** de la variable a la que se accede y sin “m_”:

int**i**MyVar; //variable

4.1.16 Obtención del valor

int **i**MyVar();

{

return**i**MyVar;

}

4.1.17 Establecimiento del valor

void MyVar(char* arg_**i**MyVar)

{

iMyVar = arg_**i**MyVar;

}

4.1.18 Obtención y establecimiento del valor

```
int& iMyVar();
{
return iMyVar;
}
```

4.2 Diagramas de Clases del Diseño

Las clases necesarias para el funcionamiento del sistema a desarrollar tienen responsabilidades estrechamente relacionadas, pero debido a la extensión del diagrama de clases del diseño se agruparán las mismas para una mejor comprensión. El diagrama en su forma original se puede consultar en el Anexo . También como formará parte de otra aplicación debe mantener la estructura y organización de la misma.

Como el módulo formará parte de la SceneToolKit se seguirán las normas estructurales de la misma.

A continuación se muestran las clases del diseño agrupadas para una mejor comprensión.

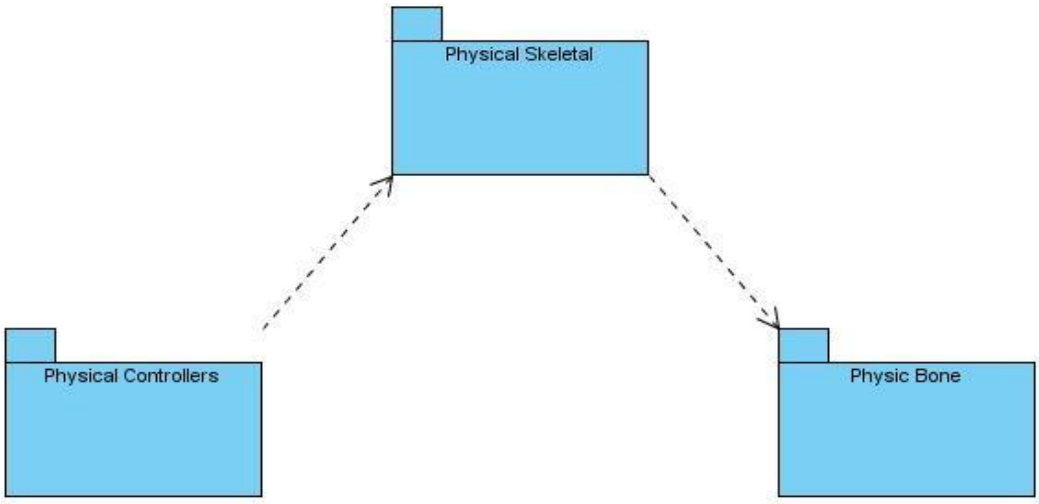

Figura 18. Agrupación de las clases del diseño

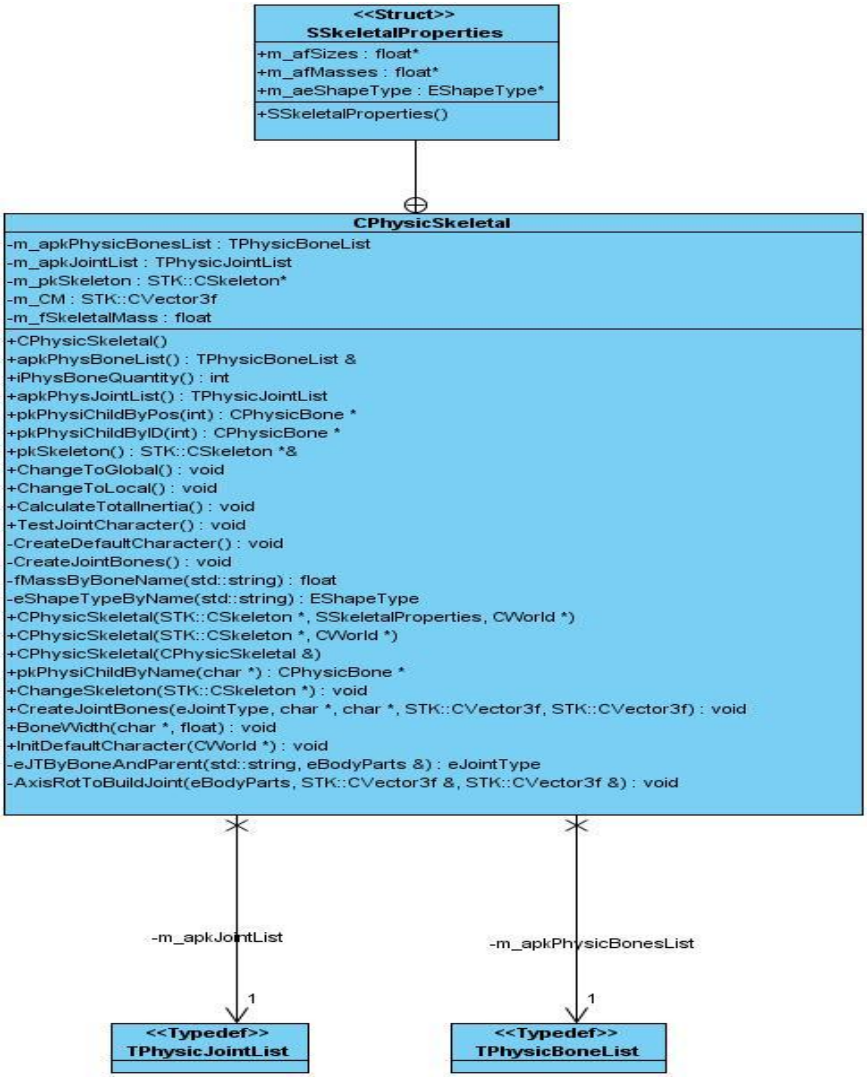

Figura 19 . Physic Skeletal

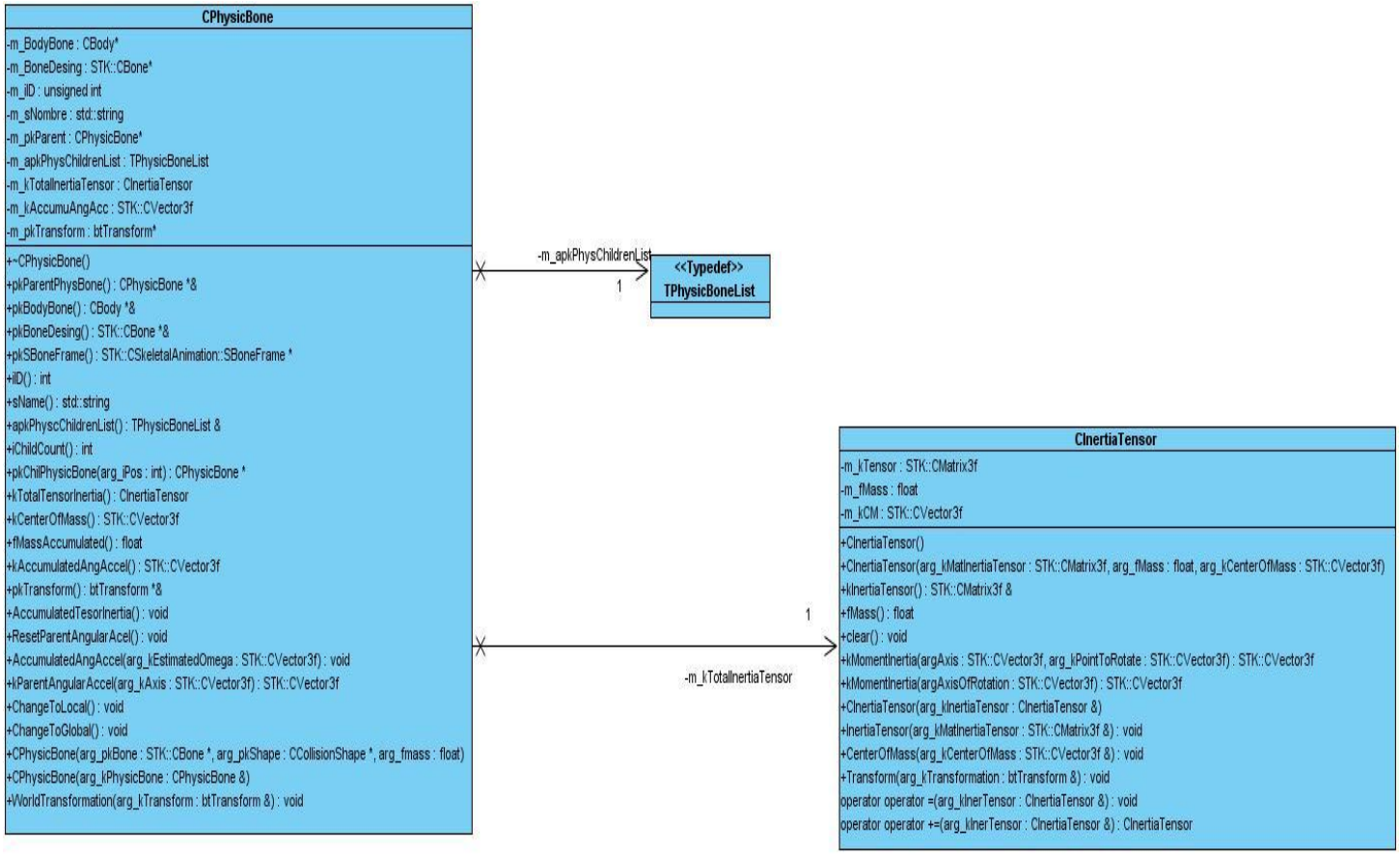

Figura 20. Physic Bone

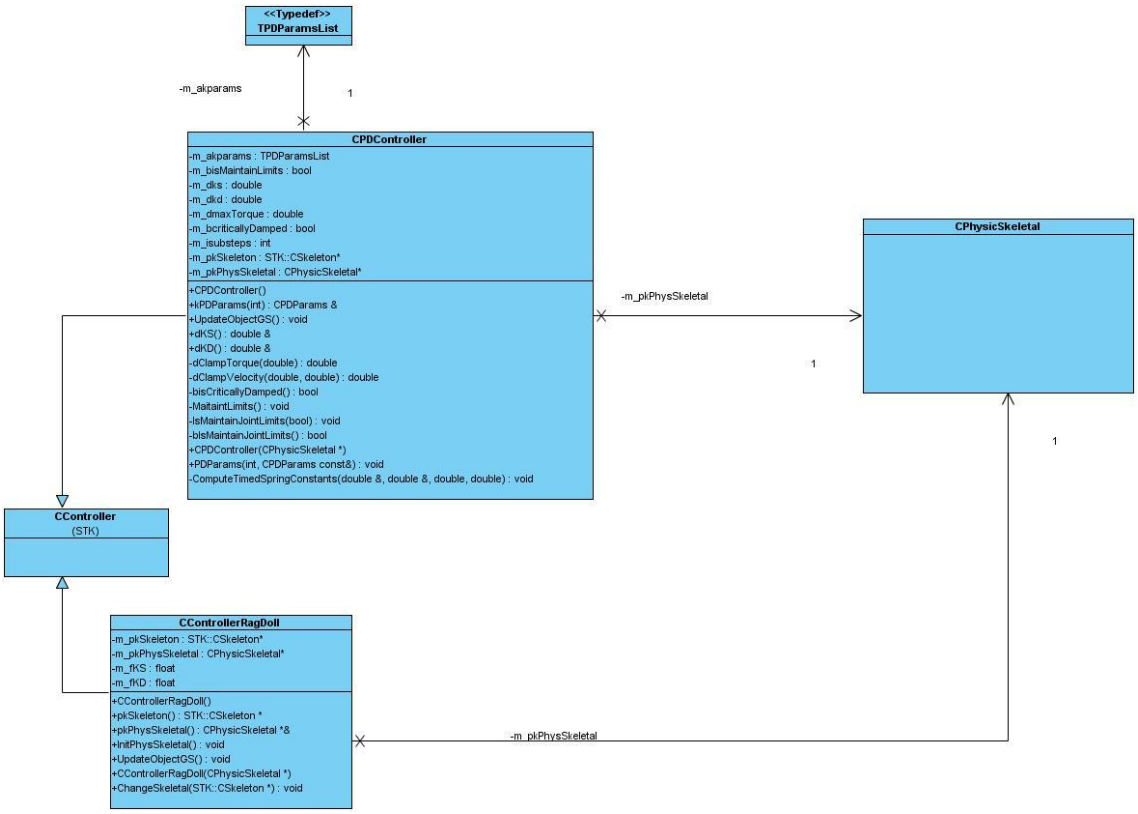

Figura 21. Physical Controllers

Conclusiones

A lo largo del este capítulo se mostraron las clases de diseño arquitectónicamente significativas. A partir de estas clases se obtuvo un módulo totalmente funcional, que cumple los requisitos funcionales expuestos, y que permite manipular de forma sencilla los objetos en los entornos virtuales, brindando la posibilidad al programador de variar distintos parámetros de dichos objetos en tiempo real así como aplicar técnicas personalizadas para lograr un alto nivel de realismo, nunca descartando la posibilidad de agregar funcionalidades nuevas y adaptar el módulo a necesidades particulares de cada aplicación.

Conclusiones

Una vez concluida la investigación previa al estudio de la simulación dinámica en la animación de personajes, se comprobó en la práctica que la STK tiene por déficit la incapacidad de combinar las animaciones desde el diseño con la simulación dinámica proporcionando una animación lo más cercana a la realidad. Con el propósito de ofrecer una solución a esta problemática se desarrolló un sistema capaz de controlar dinámicamente un personaje que le permita a este interactuar dinámicamente en un entorno y sus objetos.

El sistema resultante presenta características que lo hacen el complemento idóneo para que la herramienta (STK) sea funcional en cuanto al control dinámico de los personajes en el entorno. Algunas de estas características son:

- Es capaz de controlar personajes dinámicamente en animaciones realistas cumpliendo con el procesamiento en tiempo real.
- Brinda una interfaz de fácil uso por parte de los programadores que lo utilicen.
- Es flexible a futuras actualizaciones posibilitando incorporar mejoras a la STK.

La bibliografía es amplia pero carece de código desarrollado con la biblioteca física Bullet, por lo que todo el proceso tuvo que realizarse desde su mismo comienzo.

Por tanto, se puede asegurar que como resultado de todo el proceso realizado se obtuvo un sistema totalmente funcional y en correspondencia a los requisitos planteados, que permite el control dinámico de personajes con facilidades para el programador.

Referencias Bibliográficas

[1] Versatile and Interactive Virtual Humans:Hybrid use of Data-Driven and Dynamics-based Motion Synthesis. Agosto 2004. School of Computer Science, Carnegie Mellon University. 76 p.

[2] Thanh Giang, Robert Mooney, Christopher Peters, Carol O'Sullivan. Real-Time Character Animation Techniques. Año 2000. www.scss.tcd.ie/publications/tech-reports/reports.00/TCD-CS-2000-06.pdf

[3] MENDOZA, C. Animación. http://www.mendozajullia.com/papers/Animacion15marzo.pdf

[4] VALÉNCIA, D. I. U. D. Ampliación de Informática Gráfica. Tema 5: Animación 3D, 2007. http://informatica.uv.es/iiguia/AIG/web_teoria/tema5.pdf

[5] RUÍZ, D.; A. SANSANO, et al. Animación en Art of Illusion, 2004. http://www.dccia.ua.es/dccia/inf/asignaturas/RG/trabajos/trabajo-david-ruiz.pdf

[6] VELA, J. L. Introducción a la Informática Gráfica.Animación, 2005. http://wwwdi.ujaen.es/~juanjo/download/animacion.pdf

[7] MAESTRI, G. Character Animation 2. New Riders. 1999. 300 p.

[8] Karel Pérez Ramírez. Módulo de animación de personajes para Simuladores y Juegos. Universidad de las Ciencias Informáticas. Ciudad de la Habana, 2007. 104p.

[9] MULTON, F.; L. FRANCE, et al. Computer animation of human walking The Journal of Visualization and Computer Animation, 1999, 10(1): 39 – 54 http://www3.interscience.wiley.com/cgi-bin/abstract/60501346/ABSTRACT

[10] 3DMAX. Character studio user reference,

[11] ALBEE, T. LIGHTWAVE 3D (8) CHARACTER ANIMATION BOOK. 2004. 478 p.

[12] MARAFFI, C. MAYA CHARACTER DEVELOPMENT. New Riders. 2005. 320 p.

[13] ALIAS Alias MotionBuilder 6 User's Guide 2004.

[14] ANDAUER, C.; M. BASTIONI, et al. Blender Documentation - User Guide. WILEY

[15] Casas Y. Sistema de captura de movimiento. 2006

[16] JAUME, U. Curso de Multimedia. Capítulo 9: Animación, 2004. http://www4.uji.es/~belfern/IS34/

[17] Character Animation using Motion Capture

[18] A. Shapiro, F. Pighin, and P. Faloutsos. Hybrid control for interactive character animation. In *Pacific Graphics*, 2003.

[19] Johan Gästrin. Physically Based Character Simulation – Rag Doll Behavior in Computer Games. The School of Computer Science and Engineering. 2004. 44 p.

[20] Ari Shapiro, Derek Chu, Brian Allen, Petros Faloutsos. A dynamic Controller Toolkit. University of California, Los Angeles. 2007. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.3781&rep=rep1&type=pdf

[21] Zordan, V. B., Majkowska, A., Chiu, B., and Fast, M. (2005). Dynamic response for Motion capture animation. ACM Trans. Graph.

[22] P. WROTEK, O. C. JENKINS, M. MCGUIRE: Dynamo: Dynamic data-driven character control with adjustable balance. In Sandbox 06: Proceedings of the 2006 ACM SIGGRAPH Video Games Symposium (2006).

[23] Brian Allen, Derek Chu, Ari Shapiro, Petros Faloutsos. On the Beat! Timing and Tension for Dynamic Characters. University of California, Los Angeles. 2007.

[24] H. BARUH: Analytical Dynamics. McGraw-Hill, 1998.

[25] BOYCE W. E., DIPRIMA R. C.: Dynamic Analysis of Robot Manipulators: A Cartesian Tensor Approach, 1st ed. Springer, 1991.

[26] Axel Seugling, Martin Rölin. Evaluation of Physics Engines and Implementation of a Physics Module in a 3d-Authoring Tool. Marzo 2006. Umea University, Sweden. 90 p

[27] http://en.wikipedia.org/wiki/Physics_engine

[28] Erleben Kenny. Module based design for rigid body simulators. Technical report, University of Copenhagen. Denmark. 2002.
http://www.diku.dk/publikationer/tekniske.rapporter/2002/02-06.pdf

[29] Russell Smith. OPEN DYNAMICS ENGINE V0.5 USER GUIDE. Febrero 2006.

[30] http://es.wikipedia.org/wiki/PhysX

[31] http://developer.nvidia.com/object/physx_features.html

[32] NVIDIA® PHYSX™ SDK END USER LICENSE AGREEMENT

[33] http://en.wikipedia.org/wiki/Bullet_(software)

[34] Adrian Boeing, Thomas Bräunl. Evaluation on real-time physics simulation systems. University of Western. Australia. 2007.

Anexos

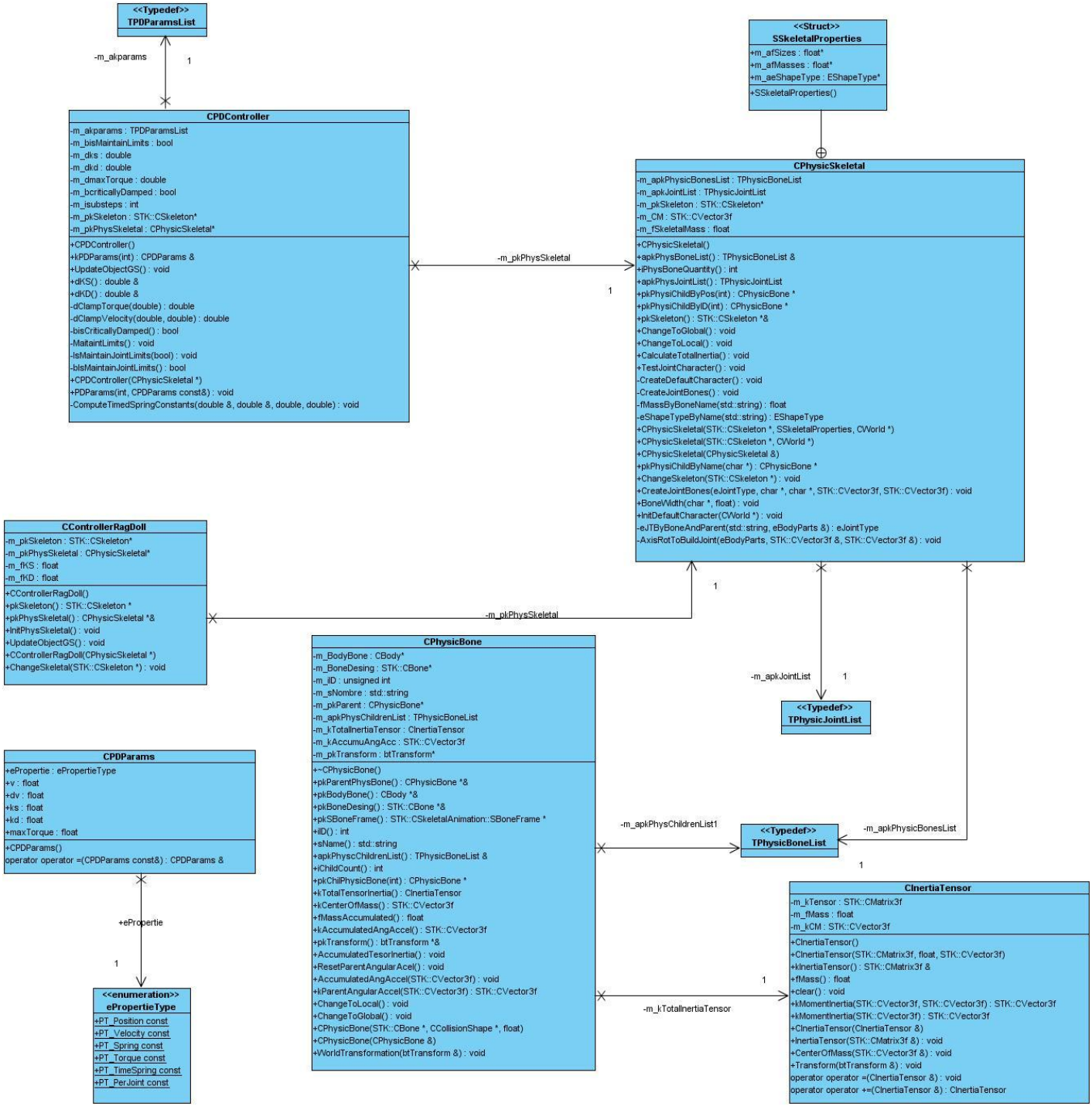

Figura 22. Diagrama de Clases del Diseño

Glosario de Términos

A

Animación: Es un proceso utilizado para dar la sensación de movimiento a imágenes o dibujos.

C

Controlador dinámico: Mecanismo de bajo nivel para el control físico sobre un cuerpo articulado.

Controladores PD: Controlador dinámico que utiliza constantes de rigidez y amortiguamiento para guiar la animación de un personaje.

D

Dinámica: Rama de la Física que estudia las causas del movimiento.

E

Esqueleto: Conjunto de piezas duras y resistentes, por lo regular trabadas o articuladas entre sí, que da consistencia al cuerpo de los animales, sosteniendo o protegiendo sus partes blandas.

F

Fotograma: Cada uno de los cuadros que conforman el movimiento de una imagen.

J

Joint: Relación de restricción entre dos cuerpos que ofrece una libertad de traslación y rotación relativa el uno del otro.

M

Momento de Inercia: es una magnitud escalar que refleja la distribución de masas de un cuerpo o un sistema de partículas en rotación, respecto al eje de giro. El momento de inercia sólo depende de la geometría del cuerpo y de la posición del eje de giro; pero no depende de las fuerzas que intervienen en el movimiento.

Motor: máquina que convierte energía en movimiento.

Motor físico: código de programa que es usado para simular la Mecánica newtoniana en el ambiente.

O

OpenCL: (Open Computing Language, Lenguaje de Computación Abierto) consta de una API y de un lenguaje de programación. Juntos permiten crear aplicaciones con paralelismo a nivel de datos y de tareas que pueden ejecutarse tanto en GPUs como CPUs.

Open Source: es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas las cuales destacan en el llamado software libre.

P

Personaje: Temática de la animación que trata la simulación de los distintos movimientos de humanoides bípedo.

PPU: Una Unidad de Procesamiento de Física (PPU por sus siglas en inglés) es un procesador especialmente diseñado para llevar a cabo cálculos físicos en un entorno 3D de videojuego. Procesos tales como partículas o humo son ahora calculados y desarrollados, no como hasta ahora que sólo eran animaciones.

R

Realidad virtual: Se define como realidad virtual un simulacro de entornos sintéticos en tiempo real, es la representación de la realidad a través de medios electrónicos, se trata de una realidad ilusoria, que existe sólo dentro de los ordenadores o medios electrónicos. De esta forma se puede definir la realidad virtual como: “La simulación de medios ambientes y de los mecanismos sensoriales del hombre por computadora, de tal manera que se busca proporcionar al usuario la sensación de inmersión y la capacidad de interacción con medios ambientes artificiales”.

***Ragdoll*:** proviene de las palabras inglesas *ragdoll* (muñeco de trapo). Como su nombre lo indica, éste es un procedimiento de animación y técnica de simulación para mostrar el movimiento de una persona como si estuviera muerto.

S

SceneToolKit (STK): Herramienta para sistemas de realidad virtual.

Simulación: Se trata de la representación simplificada, mediante un modelo, de la realidad de un proceso.

Simulación Dinámica: Es el uso de un programa de cómputo para modelas en un intervalo de tiempo el comportamiento de un sistema. Los sistemas descritos típicamente mediante ecuaciones diferenciales o parciales diferenciales.

T

Tensor de Inercia: es un tensor simétrico de segundo orden que caracteriza la inercia rotacional de un sólido rígido. Expresado en una base orto normal viene dado por una matriz simétrica, dicho tensor se forma a partir de los momentos de inercia según tres ejes perpendiculares y tres productos de inercia.

Torque: magnitud vectorial que da una medida de la capacidad de una fuerza para provocar una rotación acelerada alrededor de un eje dado, esto es, una rotación de frecuencia variable.

V

Videojuego: Según el diccionario de la Real Academia Española, videojuego es un: "dispositivo electrónico que permite, mediante mandos apropiados, simular juegos en las pantallas de un televisor o de un ordenador".