PROYECTO FINAL

INGENIERÍA INFORMÁTICA - ITBA

SIMULACIÓN Y ANIMACIÓN BIOMECÁNICA DE UN HUMANOIDE

Autores:

Altamiranda Graterol, Enzo
ealtamir@itba.edu.ar

Fontanella De Santis, Teresa
tfontane@itba.edu.ar

Mehdi, Tomás
tmehdi@itba.edu.ar

Tutor:

Dr. Parisi, Daniel Ricardo

Instituto Tecnológico de Buenos Aires - ITBA
INGENIERÍA INFORMÁTICA

Marzo 2016

Índice

Resumen

Este proyecto tiene como objetivo crear una simulación y animación de un humano virtual, con las siguientes propiedades:

- Biomecánica: que tanto su estructura (peso, altura y posición de cada una de sus partes) como su interacción con el entorno, respondan a comportamientos físicos reales y exactos.
- Inteligencia Artificial: que aprenda a caminar por sí mismo, utilizando para ello métodos de *soft computing* como Algoritmos Genéticos.

1. Introducción

No está terminada aún

Siempre ha sido de interés la simulación biomecánica de seres vivos, especialmente en las ciencias naturales (zoología, medicina, etc.). Pero últimamente se ha incrementado el interés en otras áreas de aplicación, como los videojuegos, para agregarle . Una característica muy importante de este trabajo es que, el humanoide no es fruto de una animación, sino un objeto compuesto de segmentos físicos, que interaccionan.

2. Herramientas

2.1. Motor Físico

Se le llama motor físico o *physics engine* a un "software capaz de realizar simulaciones de ciertos sistemas físicos, como la dinámica del cuerpo rígido, el movimiento de un fluido y la elasticidad"[1].

Actualmente, existen muchos motores físicos: ya sea de código propietario (PhysX, Havok), como open-source (*Bullet Physics*, *Box2D*, Newton, OGRE). Considerando análisis relacionados [2][3], y la necesidad de que el espacio simulado fuese en 3D, se decidió que *Bullet Physics*[4] es el más idóneo. Está implementado en C++ y ha sido utilizado en varios juegos (*Grand Theft Auto IV* y V, etc); en los efectos especiales de películas (Hancock, Bolt, etc.); y proyectos científicos, como la herramienta open-source *Tensegrity Robotics Toolkit* de la NASA[1]; entre otros.

Si bien (como se verá más adelante) Bullet tiene problemas asociados con el coeficiente de restitución, posee una muy buena performance en la detección de colisiones, la dinámica y la resolución de constraints. Esto se debe, en parte, a diferentes algoritmos iterativos de orden lineal (donde el más importante es *Sequential Impulse*), de caching y también a la utilización de un modelo de fricción de Coulomb aproximado [5]. Además, el motor físico brinda la posibilidad de regular la precisión requerida en estos cálculos (sin olvidar que, con iguales recursos, a mayor precisión, mayor capacidad de cómputo requerida y, ergo, mayor tiempo). Dado que la construcción del humanoide implica definir características y restricciones de movimiento de cada una de sus partes, lo antes mencionado fue crucial para la elección de *Bullet Physics* en este proyecto.

[1] http://bulletphysics.org/Bullet/phpBB3/viewtopic.php?f=17&t=9978

2.1.1. Modelo de fricción utilizado y su verificación

Hay reglas físicas relacionadas con el entorno y que son muy importantes para la caminata: el modelo de fricción, con sus respectivos coeficientes de fricción y restitución.
En base a los modelos físico-matemáticos utilizados en los dos fenómenos en cuestión (y que se explicarán a continuación), se llevaron a cabo dos experimentos para verificar que estuvieran en concordancia con los datos arrojados por *Bullet*:

- El primero simula un cubo con una velocidad constante en el eje horizontal, que gradualmente se detiene por acción de la fricción, hasta llegar al reposo. Se buscó determinar si el modelo utilizado por *Bullet* para simular las fuerzas resultantes sobre un cuerpo por acción de la fricción.
  Para este experimento se utilizó el modelo matemático que representa la posición del cuerpo en el eje horizontal en función del tiempo, representado por la siguiente ecuación:

$$x(t) = x_i + v_i t + \frac{1}{2} a t^2 \tag{1}$$

  En este caso, el cuerpo empieza su movimiento en el origen, por lo tanto la posición inicial (x_i) es cero.
  Debido a la fricción entre el cuerpo y el suelo, se genera una fuerza de rozamiento F_{μ_d} en ec. (2) en la misma dirección que la velocidad del sólido y en sentido contrario.

$$-F_{\mu_d} = \mu_d F_N \tag{2}$$

  donde $F_N = mg$ es la fuerza normal que actúa sobre la caja por acción de la gravedad g, y μ_d es el coeficiente de fricción dinámico.

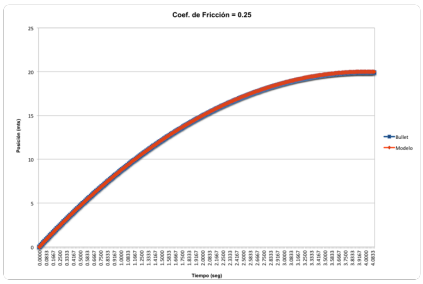
  Finalmente, se obtiene la aceleración:

$$a = \frac{F_{\mu_d}}{m} = \frac{-\mu_d F_N}{m} = \frac{-\mu_d mg}{m} = -\mu_d mg \tag{3}$$

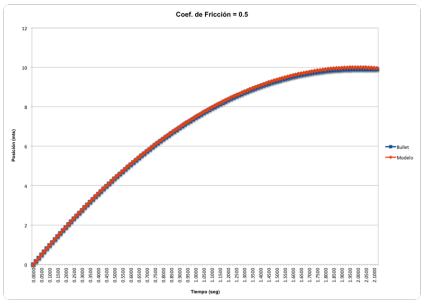
  Considerando las ecuaciones (1) y (3), se puede obtener el modelo matemático que predice el movimiento de la caja:

$$x(t) = x_i + v_i t - \frac{1}{2} \mu_d g t^2 \tag{4}$$

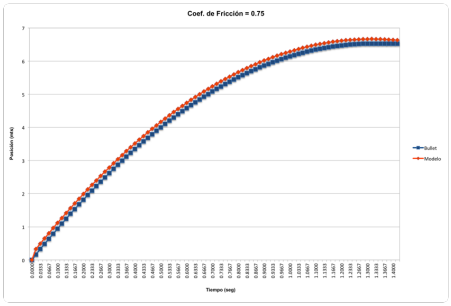
  Los resultados obtenidos -ver Fig. 1, 2 y 3- exponen que posiblemente *Bullet* utilice el modelo antes expuesto a la hora de simular. No obstante, vale aclarar que, cuanto mayor sea el paso de simulación -o *stepping*- empleado, mayor es la discrepancia entre la simulación y el modelo, posiblemente porque la precisión es menor y eso lleva a cometer un error mayor.



(a)

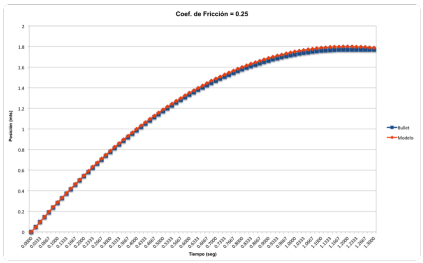


(b)

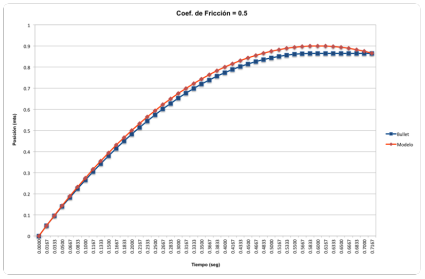


(c)

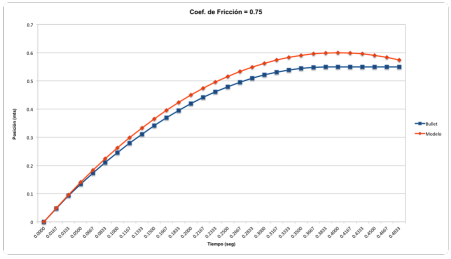
Figura 1: $v_i = 10\frac{m}{s}$: (a) $\mu_d = 0{,}25$, (b) $\mu_d = 0{,}50$, y (c) $\mu_d = 0{,}75$



(a)

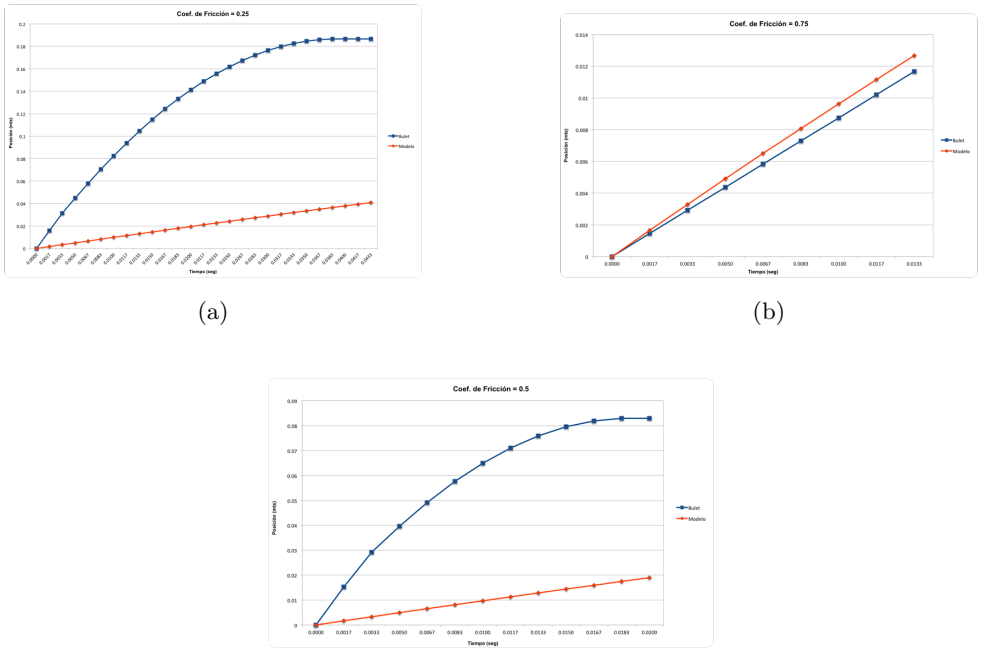


(b)



(c)

Figura 2: $v_i = 3\frac{m}{s}$: (a) $\mu_d = 0{,}25$, (b) $\mu_d = 0{,}50$, y (c) $\mu_d = 0{,}75$



(a) (b)

(c)

Figura 3: $v_i = 1\frac{m}{s}$: (a) $\mu_d = 0{,}25$, (b) $\mu_d = 0{,}50$, y (c) $\mu_d = 0{,}75$

– El segundo simula una esfera a una altura determinada sobre el suelo, que tiene una velocidad en el eje perpendicular al piso y que eventualmente colisiona contra el mismo. Se desea comprobar que la colisión entre el cuerpo y el suelo respete que la velocidad final de la esfera después del choque sea proporcional a su coeficiente de restitución e dado por la ecuación:

$$e = \frac{v_f}{v_i} \tag{5}$$

Las siguientes tablas muestran los experimentos realizados. En ellos, se mantienen fijos v_i (velocidad inicial) y e_{sim} (coeficiente de restitución esperado); se obtiene v_f (velocidad de la esfera al finalizar la simulación); y se calculan e_{medida} (coeficiente de restitución obtenido a partir de (5)) y ϵ_{rel} (error relativo entre los coeficientes):

v_i	-0.5		
v_f	0.000249	0.000219	0.001037
e_{medida}	0.000498	0.000438	0.002074
e_{sim}	0.2	0.5	0.8
ϵ_{rel}	0.997	0.999	0.997

Tabla 1: Coeficiente de fricción de 0.75

v_i	-3.5		
v_f	0.000057	0.000018	0.3
e_{medida}	0	0	0.0857
e_{sim}	0.2	0.5	0.8
ϵ_{rel}	1	1	0.893

Tabla 2: Coeficiente de fricción de 0.75

v_i	-4		
v_f	0.000473	0.000424	1.23
e_{medida}	0.00012	0.00011	0.3
e_{sim}	0.2	0.5	0.8
ϵ_{rel}	1	1	0.625

Tabla 3: Con coeficiente de fricción de 0.75

v_i	-5		
v_f	1	2.5	4
e_{medida}	0.2	0.5	0.8
e_{sim}	0.2	0.5	0.8
ϵ_{rel}	0	0	0

Tabla 4: Con coeficiente de fricción de 0.75

v_i	-10		
v_f	2	5	8
e_{medida}	0.2	0.5	0.8
e_{sim}	0.2	0.5	0.8
ϵ_{rel}	0	0	0

Tabla 5: Con coeficiente de fricción de 0.75

Los resultados exponen una limitación del motor físico: no representa correctamente las colisiones elásticas entre esferas y cuerpos rígidos, que ocurren a velocidades bajas. Esto queda en evidencia en las Tablas 1, 2, 3 y 4. En cada una de ellas el error fue de casi el 100 %.
La razón por la que ocurre este hecho se debe a que *Bullet* utiliza un algoritmo de colisión que frena la velocidad de un objeto que está a punto de colisionar. Haciendo esto puede evitar que los sólidos se traspasen y de esta forma se pueden realizar cálculos de fuerza más precisos. En el caso de los experimentos, las esferas poseen una rapidez muy baja, cuando están a punto de colisionar *Bullet* reduce aún más esta velocidad y eventualmente quedan con una velocidad tan baja que al chocar contra el suelo se aplica el efecto restitutivo a esta rapidez casi nula y se resuelve que la esfera debe quedar en reposo, cuando en realidad debería poseer

una velocidad baja, pero no despreciable.

2.1.2. Ventajas

- Código abierto: mayor conocimiento sobre las fórmulas y métodos implementados en el motor, a diferencia de lo acaecido en trabajos previos [6], en donde al usar frameworks físicos de código cerrado, no se tenía ni control ni conocimiento pleno de su funcionamiento.
- Soporte de la comunidad cientfífica.
- Licencia libre.

2.1.3. Desventajas

- Documentación poco clara y desordenada.
- Implementado en C++: al no formar este lenguaje parte de la expertise del equipo, fue más difícil de desarrollar.

2.2. Librería de Algoritmos Genéticos

Se utilizó la conocida librería de Algoritmos Genéticos para C++ GaLib, desarrollada por Matthew Wall del MIT [7]. Ofrece funcionalidades como: programación paralela, diversos métodos de selección (elite, ruleta), estrategias de reemplazo (de padres, aleatorio, del peor), entre otras. Se podría haber implementado una librería propia; pero hubo que desestimarla, considerando el tiempo requerido y su complejidad.

2.3. Código Fuente

Al estar *Bullet* implementado en C++, el código fuente también está desarrollado en ese lenguaje. En *Bullet*, se define un *World* -o mundo físico- en donde se puede insertar, entre otras cosas, cuerpos rígidos. En este caso en particular, el mundo consta de un plano -el suelo- y el humanoide encima -compuesto por cuerpos rígidos y otros elementos físicos-.
El programa incluye tanto la visualización gráfica de la simulación física de *Bullet* del humanoide, así como el algoritmo genético -y la definicón de los individuos, fitness, etc. y su integración con GaLib-.
El código fuente se adjunta a esta presentación y una descripción del mismo y modo de uso e instalación en el manual de usuario.

3. Modelo Utilizado

Hay diversos modelos. Algunos son más genéricos [8] [9] y complejos. Sin embargo, se procuró utilizar uno que fuera sencillo pero representativo a la vez.
Se modela al cuerpo humano, con el motor *Bullet Physics*, como un conjunto de segmentos unidos por articulaciones. A cada uno de ellos se les aplica una fuerza en el centro de masa de cada

segmento (denominada Actuador). Que la caminata se produzca o no, depende del tipo de actuador utilizado (la función utilizada para la fuerza), y de sus parámetros. El objetivo, entonces, se reduce a encontrar dichos parámetros. Para eso se usan los algoritmos genéticos, un método de Inteligencia Artificial. De este modo, se obtiene, de forma análoga a la selección natural, los individuos que mejor se adapten a la caminata. Tanto los actuadores como el algoritmo genético se explicarán más adelante.

3.1. Composición Física del Humanoide

No está terminada aún

Como ya se dijo, el humanoide fue modelado en *Bullet* como un conjunto de segmentos, unidos por articulaciones. Los segmentos son cuerpos rigídos

Se dividió al cuerpo humano en los siguientes segmentos: cabeza, tronco, miembro superior, pelvis y miembro inferior (muslo, pierna y pie). Considerar la mitad superior del cuerpo implicaba tener Para la simplificación del problema

A continuación se presenta la composición de cada segmento (de acuerdo a la biomecánica [10]):

Parte	Cantidad	Forma	Largo (en m)	Peso (en kg)	Uniones
Pelvis	1	esférico	0.08655	9.9718	Cadera
Muslo	2	esfero-cilindro	0.4015	10.3368	Cadera y Rodilla
Pierna	2	esfero-cilindro	0.4015	3.1609	Rodilla y Tobillo
Pie	2	esfero-cilindro	-	1.0001	Tobillo

Tabla 6: Segmentos del humanoide

3.2. Articulaciones

No está terminada aún

Para unir los distintos segmentos entre sí, se utilizaron articulaciones bisagra con 1 grados de libertad: en el eje Z -en donde ocurre la caminata- y en el eje Y -el perpendicular al piso-. Además, para cada caso en particular, se definieron cotas para los ángulos que pueden existir entre los segmentos. Esto es muy importante, no sólo porque se adecúa a datos biológicos, sino porque, de otro modo la caminata no podría lograrse: si los ángulos son demasiado altos, la caminata se produce girando las piernas por encima de la pelvis; si por el contrario, son demasiado bajos, las piernas van a estar muy rígidas, originando pocos pasos y muy cortos. Tanto para la
Por otra parte, a la pelvis se le restringe todo tipo de rotación.

4. Actuadores

Para representar la fuerza aplicada a cada uno de los segmentos, se utilizaron diferentes funciones (todas ellas periódicas). A fin de simplificar el modelo, el humanoide tiene el mismo tipo de actuador aplicado en todos los segmentos.

4.1. Genérico

Es el actuador más sencillo (tanto matemática como computacionalmente). No se consideraron las funciones trigonométricas básicas (es decir, solo sin o cos), por los resultados obtenidos en el trabajo del Cuadrúpedo, antes mencionado. La fase es la misma tanto en el seno como en el coseno, para evitar que se formen otro tipo de funciones no cclicas ($f(t) = 0$).

$$f(t) = A_1 \sin(\omega_1 t + \phi) + A_2 \cos(\omega_2 t + \phi) + C \tag{6}$$

4.2. Fourier

Este actuador utiliza una serie de Fourier de dos términos.

$$f(t) = A_1 \sin(\omega t + \phi) + B_1 \cos(\omega t + \phi) + A_2 \sin(2\omega t + \phi) + B_2 \cos(2\omega t + \phi) + C \tag{7}$$

4.3. Extra Fourier

Es una extensión del actuador anterior, pero con 9 términos. Por ser de mayor grado, brinda una mayor precisión. Sin embargo, es más dificil de manejar computacionalmente; y, además, que sea más preciso no garantiza que con él se pueda lograr una buena caminata.

$$\begin{aligned} f(t) =& A_1 \sin(\omega t + \phi) + B_1 \cos(\omega t + \phi) + A_2 \sin(2\omega t + \phi) + B_2 \cos(2\omega t + \phi) \\ &+ A_3 \sin(3\omega t + \phi) + B_3 \cos(3\omega t + \phi) + A_4 \sin(4\omega t + \phi) + B_4 \cos(4\omega t + \phi) \\ &+ A_5 \sin(5\omega t + \phi) + B_5 \cos(5\omega t + \phi) + A_6 \sin(6\omega t + \phi) + B_6 \cos(6\omega t + \phi) \\ &+ A_7 \sin(7\omega t + \phi) + B_7 \cos(7\omega t + \phi) + A_8 \sin(8\omega t + \phi) + B_8 \cos(8\omega t + \phi) \\ &+ A_9 \sin(9\omega t + \phi) + B_9 \cos(9\omega t + \phi) + C \end{aligned} \tag{8}$$

4.4. Doble coseno

Su función está dada por dividir a la funcin peridica en dos partes, cada una de ellas con funciones cosenoidales de frecuencias distintas.

$$\psi(t) = t + \phi - \left\lfloor \frac{t + \phi}{\pi/\omega_1 + \pi/\omega_2} \right\rfloor (\pi/\omega_1 + \pi/\omega_2) \qquad \psi : \mathbb{R} \to \left[0, \frac{2\pi}{\omega}\right] \tag{9}$$

$$\omega = \frac{2\omega_1\omega_2}{\omega_1 + \omega_2} \tag{10}$$

$$f(t) = \begin{cases} A\cos(\omega_1\psi(t)) + C & \text{si } \omega_1\psi(t) < \pi \\ A\cos(\omega_2(\psi(t) - (\pi/\omega_1) + (\pi/\omega_2))) + C & \text{en otro caso} \end{cases} \tag{11}$$

5. Condiciones iniciales y de contorno

Las funciones periódicas mencionadas en los actuadores no son suficientes para lograr la caminata. A los actuadores mencionados previamente, se le adosaron las funciones vistas a continuación.

5.1. Función Partida

El andar del humanoide es cíclico. Sin embargo, por la posición inicial del individuo, se requiere para el tiempo del primer paso, una función distinta a la del resto de la caminata. El tipo de función puede ser cualquiera de los actuadores vistos anteriormente (pero no necesariamente con los mismos valores de amplitud, frecuencia y fase asignados a las piernas). Empero, se utilizó la función vista en el actuador genérico.
Por otra parte, para simplificar el modelo, se decidió que el tiempo considerado para el primer paso sea fijo, y de 0.7 segundos. Dicho valor fue extraído de forma experimental.

5.2. Fase sincronizada

En una caminata, las piernas deben guardar simetría: mientras una va hacia adelante, la otra va hacia atrás (y viceversa). Esto, de acuerdo con los actuadores definidos en la sección anterior, implica que las funciones de movimiento de cada pierna estén desfasadas en medio ciclo ($\frac{\pi}{2}$):

$$f_{izquierda}(t) = f(t) \tag{12}$$

$$f_{derecha}(t) = f(t + \frac{\pi}{2}) \tag{13}$$

siendo $f(t)$ la función de movimiento (o actuador) en el momento t.

6. Algoritmo Genético

6.1. Individuo

Los algoritmos genéticos, generalmente, suelen ser lentos: tardan mucho tiempo, y también en converger a una solución óptima (necesitan muchas generaciones para llegar a eso, y pasando por varios mximos locales). El manejo de un individuo debe ser eficiente y simple. Y eso se logra con estructuras sencillas y básicas. Es por eso que el individuo, en este caso, está compuesto por un array de números, que identifican tanto los valores asociados a las fuerzas de los actuadores.

6.2. Fitness

El papel de la función de fitness en un algoritmo genético es evaluar qué tan bueno es un individuo. En este caso, está definida como un producto de tres módulos o propiedades: altura, velocidad y dirección:

$$\text{fitness} = \text{altura} * \text{velocidad} * \text{dirección} * \text{simetría} * \text{pies abajo} \tag{14}$$

Los tres tienen la misma importancia y, por eso, como se verá a continuación, están definidos de forma similar (con una función exponencial y pueden valer entre 0 y 1). Con todo esto, dado que el fitness está pensado como un producto, basta con que uno de los módulos sea muy chico para "anular" al individuo -es decir, otorgarle un valor que tiende a cero-. Sin embargo, los diferentes módulos no son completamente independientes entre sí: por ejemplo, si la altura es demasiado baja, posiblemente la velocidad y la dirección no sean adecuadas.

6.2.1. Altura

Es un factor relacionado con la altura del individuo en toda la simulación, y se expresa:

$$\text{altura} = \frac{\sum_{n=0}^{T} e^{-C(h_{t_n} - h_{t_0})^2}}{simulation\ steps} \tag{15}$$

donde t_0 es el tiempo inicial, t_T el tiempo final, simulation_steps la cantidad pasos de simulación y C una constante adimensional que vale 5 (dato experimental).
Se calcula a partir de la diferencia entre la altura en cada instante de la simulación, con su altura inicial -la altura está definida como la posición de la pelvis en el eje Z-. Cuanto mayor sea esa diferencia, más rápido el individuo cae, y por eso este módulo tiende a cero. Por el contrario, valdrá uno si la diferencia es ínfima -lo que significa que el humanoide mantiene su misma altura durante la caminata-.

6.2.2. Velocidad

Indica qué tan cercana es la velocidad del individuo con respecto a una velocidad objetivo -en este caso, es de 1.2 m/h-, y se expresa de la siguiente forma:

$$\text{velocidad} = \frac{\sum_{n=0}^{T} e^{-C(\|v_{t_n}\| - V_O)^2}}{simulation\ steps} \tag{16}$$

donde t_0 es el tiempo inicial, t_T el tiempo final, V_O la velocidad objetivo en el eje Z -el eje de la caminata-, simulation_steps la cantidad pasos de simulación y C es una constante adimensional que vale 5 (dato experimental).
Sigue una lógica y cálculo similares al factor de altura: a mayor discrepancia de la velocidad real del humanoide con V_O, menor -y más cercano a cero- es el valor arrojado por el módulo de velocidad.

6.2.3. Dirección

Señala qué tan similares son la dirección objetivo -un vector unitario, que en este caso se encuentra en el eje Z- y la dirección con la que camine el humanoide. Se calcula como sigue:

$$\text{dirección} = \frac{\sum_{n=0}^{T} e^{-C(\vec{v_{t_n}} \cdot \vec{V_O} - 1)^2}}{simulation\ steps} \tag{17}$$

donde t_0 es el tiempo inicial, t_T el tiempo final, $\vec{v_{t_n}}$ el versor de la dirección del humanoide en el momento t_n, $\vec{V_O}$ el versor de la dirección objetivo, simulation_steps la cantidad pasos de simulación y C es una constante adimensional que vale 5 (dato experimental).
El producto escalar entre los versores responde a la Similitud Coseno: $\cos\theta = \frac{A \cdot B}{\|A\|\|B\|}$, donde A y B son vectores que no se encuentran normalizados, y θ es el ángulo formado entre ellos. Así, si $\cos\theta = 1$, significa que los vectores están paralelos entre sí -que es el efecto buscado en el caso de la dirección-.
Al producto escalar se le resta 1, para que el módulo sea consistente con la función exponencial utilizada y que valga 1 cuando $\theta = 0$, y 0 cuando $\theta = \pi$. Cabe aclarar que se trata al ángulo en

forma simétrica, ya que, por ejemplo $\cos(-\pi/6) = \cos(\pi/6)$.

6.2.4. Simetría

Señala qué tan equidistantes se encuentran los pies de la cadera, a lo largo de la caminata. Aplicando solamente los módulos antes mencionados, provocaba resultados en donde una pierna quedaba más distante de la pelvis que la otra, lo que provocaba que el humanoide se terminara arrastrando (y posiblemente afectando a la velocidad).
Para mayor simplicidad, la simetría se calculó a partir de los pies (y no de las piernas). Se tomaron en cuenta sólo los ejes X y Z, porque son los relacionados a la velocidad y a la direcci'on, respectivamente.

$$\text{simetría} = \frac{\sum_{n=0}^{T} \frac{1}{2}[e^{-C(lfoot_Z + rfoot_Z|^2)} + e^{-C(lfoot_X + rfoot_X|^2)}]}{simulation\ steps} \tag{18}$$

donde $lfoot_X$ y $lfoot_Z$ es la distancia desde el pie izquierdo hasta la pelvis en los ejes X y Z, respectivamente; y en donde $rfoot_X$ y $rfoot_Z$ es lo mismo, pero para el pie derecho.

6.2.5. Pies abajo

Con los módulos sealados anteriormente, se resalta que el humanoide camine con una velocidad y dirección determinadas, que no se caiga y que mantenga simetría mientras ejecuta sus movimientos. Pero, todo esto daría, en el mejor de los casos, una caminata estilo "estrella".
Sin embargo, una característica fundamental en una caminata normal es que las piernas (ergo, los pies también) no sobrepasen la cadera. Si bien ésta es una propiedad negativa (expresa lo que no debe tener una caminata), y se puede correr el riesgo de restringir demasiado, su ausencia da resultados peores.

$$\text{pies abajo} = \frac{\sum_{n=0}^{T} \frac{1}{2}(\alpha[e^{-C(l\_diff\_foot^2)} + e^{-C(r\_diff\_foot^2)}]}{simulation\ steps} \tag{19}$$

donde $\alpha = max(min(leftfoot, rightfoot) - hip, 0), 1)$ (es decir, vale 0 si la altura del pie izquierdo o derecho supera a la de la cadera, y 1 en otro caso); y l_diff_foot y r_diff_foot es la diferencia entre la posición inicial de los pies y la altura en el momento t_n de los pies izquierdo y derecho, respectivamente.

6.3. Parámetros del Algoritmo

6.3.1. Métodos de selección

Varios son los métodos de selección que son usados con frecuencia: Elite (priorizan en cada generación los individuos con mejor fitness), y los estocásticos (los individuos que sobreviven en cada generación son elegidos probabilísticamente), como Rouleta, por Torneos, Boltzmann.
Actualmente, la función de fitness involucra características muy restrictivas (como la simetría y el hecho de prohibir que los pies estén por encima de la pelvis). Utilizar métodos de selección que prioricen lo estocástico no resulta conveniente (pues mayor será la probabilidad de generar

individuos no favorables). Por eso, se utilizó un método un poco más elitista, como el de Torneos. La probabilidad es de 0.7.

6.3.2. Métodos de cruza

xxxx

6.3.3. Mutación

xxxx

7. Resultados Obtenidos

ccccccvvvvv

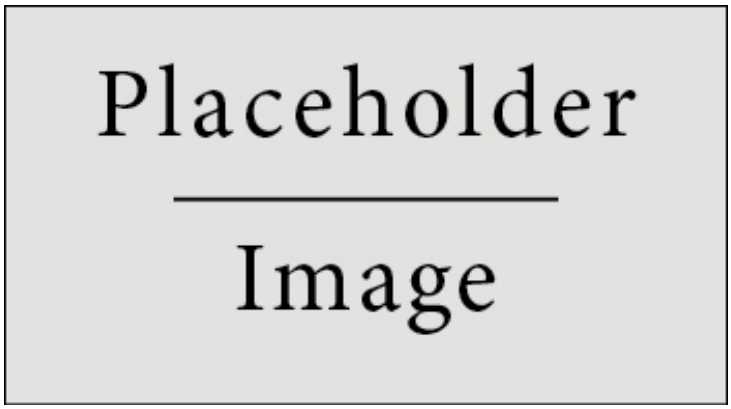


Figura 4: Figure caption.

8. Conclusiones

xxxxxxxx

Referencias

[1] Wikipedia: *https : //es.wikipedia.org/wiki/Physics_engine*

[2] Andreas Gerndt y otros, *An Evaluation of Open Source Physics Engines for Use in Virtual Reality Assembly Simulations, Fecha de publicación: 2012*

[3] Tom Erez y otros, *Simulation Tools for Model-Based Robotics: Comparison of Bullet, Havok, MuJoCo, ODE and PhysX*

[4] Sitio web de Bullet Physics: *http : //www.bulletphysics.org/*

[5] Erin Catto, *Iterative Dynamic with Temporal Coherence, Fecha de publicación: 2005*

[6] Kevin Kenny, Máximo Videla y Axel Wassington, *Simulación y Animación Biomecánica de un Cuadrúpedo, 2014 — ITBA*

[7] Sitio web de GaLib: *http : //lancet.mit.edu/ga/*

[8] Thomas Geijtenbeek, Michiel van de Panne y A. Frank van der Stappen, *Flexible Muscle-Based Locomotion for Bipedal Creatures, 2013*

[9] Marek Wojtyra, *Multibody Simulation Model of Human Walking, 2003 — Warsaw University of Technology*

[10] *http : //www.exrx.net/Kinesiology/Segments.html*