STUDIA INFORMATICA 2006
Volume 27 Number 1 (66)

Tomasz MYSŁEK, Tomasz GRUDZIŃSKI, Jacek ROSS
Politechnika Śląska, Instytut Informatyki

ZAAWANSOWANE TECHNIKI ANIMACJI TRÓJWYMIAROWYCH MODELI SZKIELETOWYCH W ŚRODOWISKU FRS

Streszczenie. W publikacji wyjaśniono ideę animowania modeli trójwymiarowych za pomocą danych szkieletowych oraz opisano sposoby automatycznej generacji animacji. W ramach przedstawiania metod automatycznej generacji animacji zostały opisane problemy kinematyki prostej i odwrotnej oraz opisano algorytm przybliżonego rozwiązywania problemu kinematyki odwrotnej.

Słowa kluczowe: model, szkielet, animacja, kinematyka prosta, kinematyka odwrotna

THE ADVANCED TECHNIQUES OF THREE-DIMENSIONAL SKELETON MODELS ANIMATION IN FRS ENGINE

Summary. The paper discusses the idea of the animation of a 3-dimensional model by using its skeleton data and also presents some methods of automatic animation generation. During description of methods of automatic animation generation, topics such as forward kinematics and inversed kinematics are mentioned and also the algorithm of approximate solution of inversed kinematics problem is presented.

Keywords: model, skeleton, animation, forward kinematics, inversed kinematics

1. Środowisko FRS

FRS (ang. Flexible Reality Simulation) jest programistycznym silnikiem wirtualnej rzeczywistości, pozwalającym na łatwe pisanie aplikacji opartych właśnie na wirtualnej rzeczywistości. Silnik jest obecnie intensywnie rozwijany i nie wszystkie założenia są już spełnione,

jednak osiągnięto już pewną funkcjonalność, pozwalającą na zaprezentowanie prac szerszej publiczności i wyciągnięcie pewnych wniosków.

Twórcy FRS postawili sobie za główny cel stworzenie silnika uniwersalnego i łatwego w użyciu, który poza standardowymi elementami podobnych projektów zawierałby komponenty bardziej wyspecjalizowane, możliwe do użycia, lecz nieobowiązkowe dla piszącego aplikację bazującą na silniku. W ten sposób powstaje silnik, który poza możliwością wyświetlania trójwymiarowej grafiki, interakcją między obiektami i sprawną obsługą dźwięku umożliwi także symulację pogody, programowe generowanie modeli roślinności, symulację mimiki ludzkiej twarzy, użycie zaawansowanych algorytmów sztucznej inteligencji oraz sterowanie całością poprzez oferujący dużą funkcjonalność język skryptowy.

Silnik FRS omawiany jest szerzej w [1, 3]. Jednym z wielu komponentów wchodzących w skład silnika jest komponent obsługujący animowanie modeli trójwymiarowych, bazujące na informacjach o szkielecie modeli. W dalszej części publikacji skupiono się na zaprezentowaniu wniosków i tez wynikłych z pracy nad tym komponentem.

2. Wprowadzenie do problemu animacji szkieletowej

Modele trójwymiarowe umieszczone na scenie poruszane są nie tylko jako całość w obrębie samej sceny, ale także może zostać wykonany ruch pewnych elementów modelu względem niego samego. Oznacza to, że niezależnie od położenia modelu (jako pewnej całości) na scenie mogą być zmieniane położenia pewnych jego części, np. stojący w miejscu model istoty humanoidalnej może jednocześnie machać swoją ręką.

Realizacja przemieszczenia całości modelu jest operacją niemal elementarną, natomiast przemieszczanie tylko pewnych fragmentów modelu względem innych części jest znacznie bardziej złożone i temu zagadnieniu poświęcony jest niniejszy artykuł. Samo przemieszczenie fragmentów modelu może być albo utworzone na etapie projektowania modelu, jeszcze przed jego wykorzystaniem, albo już w trakcie używania modelu, bez udziału projektanta modelu. Ten ostatni aspekt tworzenia ruchu fragmentów modelu zwany jest dalej programową metodą generowania animacji, albo dynamicznym generowaniem animacji.

Źródłem doświadczeń do napisania tego artykułu była implementacja biblioteki udostępniającej tego typu operacje na modelu. Podobne problemy poruszono również w [5, 6, 7].

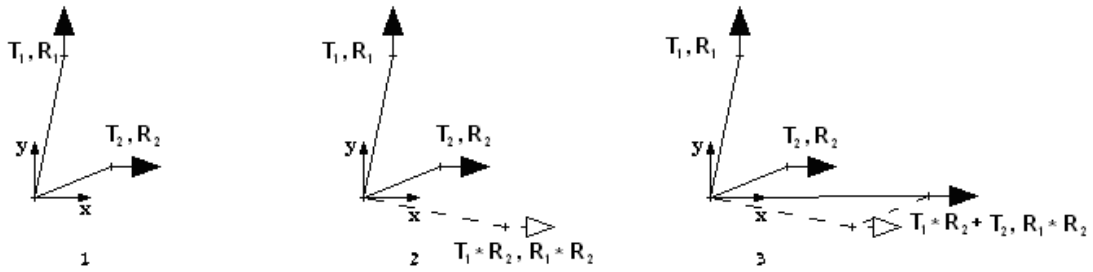
3. Idea trójwymiarowych modeli szkieletowych

Przez pojęcie modelu trójwymiarowego należy rozumieć każdy obiekt, który może zostać umieszczony na scenie trójwymiarowej. Dokładny opis modelu wymaga zdefiniowania następujących elementów:

- wierzchołek – punkt w przestrzeni trójwymiarowej; poza trzema współrzędnymi przestrzennymi (x, y, z) związane z nim są dodatkowe informacje, takie jak współrzędne wektora normalnego do powierzchni, w skład której wchodzi dany wierzchołek,
- siatka – zbiór wierzchołków pogrupowanych w trójkąty. Jest podstawowym elementem opisującym kształt i wygląd modelu. Na jej podstawie określane są interakcje modelu z otoczeniem. Z siatką związane są dodatkowe informacje określające wygląd modelu oraz sposób wyświetlenia siatki na scenie, jak np. materiał pokrywający siatkę. Sam model może się składać z więcej niż jednej siatki.

W artykule często używane będą pojęcia związane z opisem oraz przeliczaniem transformacji wierzchołków. Do danych składających się na transformację należą:

- translacja – położenie elementu lub jego przesunięcie względem środka układu odniesienia, reprezentowane przez wektor,
- rotacja – orientacja elementu lub jego obrót, reprezentowana przez kwaternion.



Rys. 1. Kroki obliczenia nowej wartości transformacji T_1, R_1 przekształcanej transformacją T_2, R_2. 1 – początkowe transformacje, 2 – T_1, R_1 obrócona o rotację R_2, 3 – T_1, R_1 obrócona o rotację R_2 i przesunięta o translację T_2

Fig. 1. Sequence of steps of calculating new value of transformation T_1, R_1 transformed by T_2, R_2. 1 – initial transformations, 2 – T_1, R_1 rotated by the rotation of R_2, 3 – T_1, R_1 rotated by the rotation of R_2 and translated by the translation of T_2

Zwykle zmiana transformacji będzie wynikać z wartości innej transformacji. Obliczenie nowej wartości transformacji polega na (rys. 1):

- obliczeniu nowej rotacji – poprzez pomnożenie rotacji pierwotnej przez rotację transformacji przekształcającej, zgodnie ze wzorem: $R_1 = R_1 * R_2$, gdzie R_1, R_2 to rotacje transformacji pierwotnej i przekształcającej reprezentowane przez kwaterniony,
- zmianie translacji – poprzez obrócenie translacji pierwotnej rotacją transformacji przekształcającej, zgodnie ze wzorem $T_1 = T_1 * R_2$, a następnie przesunięciu otrzymanej translacji o translację przekształcającą, zgodnie ze wzorem $T_1 = T_1 + T_2$, gdzie T_1, T_2 to translacje transformacji pierwotnej i przekształcającej reprezentowane przez wektory.

3.1. Animowanie modelu

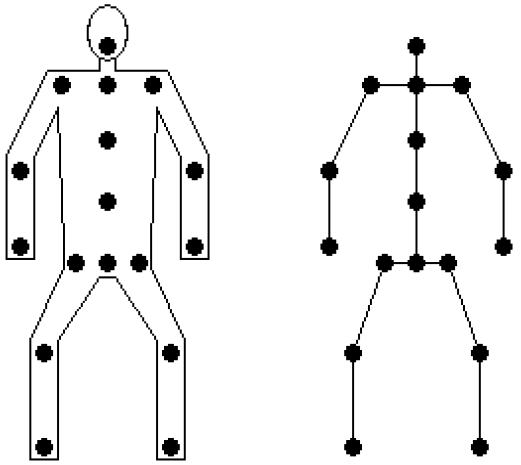
Przez pojęcie animowania należy rozumieć zestaw czynności prowadzących do tego, by w wyniku kolejnych odświeżeń wyglądu modelu na scenie model zmieniał transformacje swoich elementów, a co za tym idzie zmieniał swój kształt.

Wykonując ruch tylko części modelu, należy wziąć pod uwagę wiedzę o tym, które wierzchołki należy transformować, oraz zależności między wierzchołkami. Istotny jest przede wszystkim ten ostatni z wymienionych czynników, ponieważ model powinien zachowywać pewne pierwotne proporcje swojego kształtu, ustalone w czasie jego projektowania.

Z powodu złożoności opis ruchu jako modyfikacji siatki modelu może być stosowany głównie w przypadku prostych animacji, nie wymagających dbania o kształt modelu, lub w przypadku modeli niezwykle prostych. Skoro ruch siatką modelu jako całością jest zbyt skomplikowany, należy w pewien sposób ją uprościć, tak by animowanie fragmentów modelu nie było tak skomplikowane. W tym celu wprowadzono pojęcie szkieletu modelu.

3.1.1. Idea szkieletu

Szkielet jest pewnym zestawem wierzchołków nie wchodzących w skład modelu i nie mających sensu wizualnego. Wierzchołki te są uporządkowane hierarchicznie, tworząc strukturę drzewiastą, a ich ułożenie w przestrzeni sceny powinno w przybliżeniu oddawać kształt modelu (rys. 2). W odróżnieniu do rzeczywistości, gdzie kości szkieletu można uprościć odcinkami, w modelu szkieletowym za kości uznaje się wierzchołki, wchodzące w skład szkieletu. Można to przedstawić jako analogię do modelu biologicznego, gdzie kość modelu wyznacza koniec kości biologicznej, natomiast początek kości biologicznej wyznaczany jest przez kość-rodzica. Korzeń drzewa jest elementem bazowym wszystkich kości, może nim zostać każda kość, jednak ze względów optymalizacyjnych wskazane jest, by długość ścieżki od korzenia do poszczególnych liści nie zmieniała się.

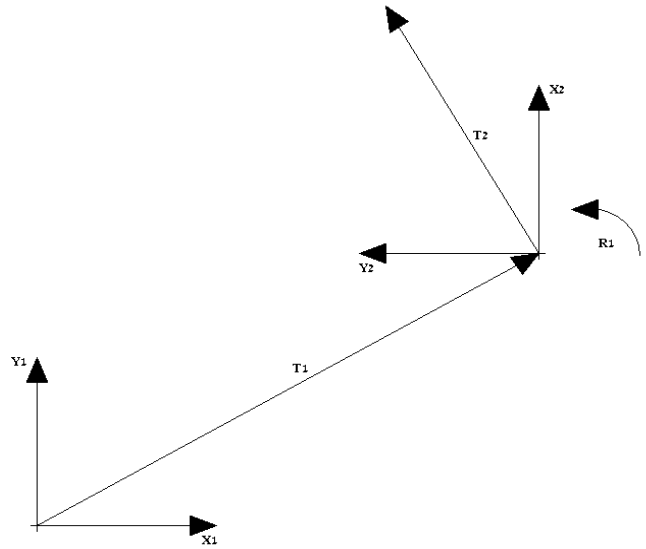


Rys. 2. Dwuwymiarowy model i jego szkielet (kropkami zaznaczono kości)
Fig. 2. Two-dimensional model and its skeleton (bones marked by the dots)

Celem zastosowania porządku hierarchicznego kości jest, aby zmiana transformacji kości powodowała tę samą zmianę u wszystkich kości-potomków danej kości.

Jedną z informacji, jaka jest związana z kością, jest jej transformacja. Transformację kości można wyrazić w układzie współrzędnych związanym z całym modelem (tak samo jak jest opisana transformacja wierzchołków). Wadą tego sposobu jest konieczność przeliczenia wszystkich transformacji kości-dzieci po zmianie transformacji rodzica.

Inne rozwiązanie przewiduje, że tylko parametry węzła-korzenia (tzn. położenie i orientacja) określone są w globalnym układzie współrzędnych, natomiast parametry wszystkich węzłów potomnych określone są w lokalnych układach współrzędnych ich rodziców (rys. 3).



Rys. 3. Translacja kości-dziecka (T_2) względem transformacji kości-rodzica (T_1, R_1)
Fig. 3. Child-bone translation (T_2) relative to the parent-bone transformation (T_1, R_1)

W ten sposób można pominąć aktualizowanie transformacji kości-dzieci podczas zmiany transformacji kości-przodka. W przypadku kości korzeniowej, czyli kości nie posiadającej rodzica, jej transformacja jest zdefiniowana w przestrzeni całego modelu. Transformacja kości korzeniowej wpłynie zatem na bezwzględną transformację wszystkich kości szkieletu, natomiast transformacja dowolnej innej kości wpływać będzie tylko na daną kość i wszystkie jej kości potomne.

3.1.2. Wpływ transformacji kości na siatkę modelu

Poza danymi związanymi z hierarchią oraz z transformacją należy przechować dane związane z wpływem kości na wierzchołki siatki. Przez pojęcie wpływu kości na wierzchołek siatki należy rozumieć transformację, opisującą zmianę bieżącej transformacji kości w stosunku do transformacji początkowej kości. Przedmiotem zainteresowania jest zmiana, a nie bieżąca transformacja kości, ponieważ modyfikacja transformacji wierzchołka polega na obliczeniu jego bieżącej transformacji przy wykorzystaniu transformacji początkowej wierzchołka.

Należy zaznaczyć, że takie obliczenia mają sens tylko wtedy, gdy są wykonywane na danych, których wartości są opisane względem tego samego układu współrzędnych. Z tego powodu transformację kości należy uprzednio przeliczyć do układu współrzędnych wyznaczonego przez model.

Ponieważ na dany wierzchołek może wpływać więcej niż jedna kość, więc poza samą wartością wpływu należy uwzględnić wagi, z jakimi kości wpływają na wierzchołek. W takim przypadku, gdy na wierzchołek wpływa więcej niż jedna kość, zmianą transformacji wierzchołka jest średnia ważona ze wszystkich zmian transformacji odziedziczonych od kości.

4. Animowanie modeli szkieletowych

Animacja szkieletowa polega na zmienianiu transformacji każdej kości w wybranym zestawie kości na taką, jaką kość powinna przyjąć w zadanej chwili. Ponieważ transformacja każdej kości jest opisana względem jej rodzica, więc zmiana transformacji danej kości nie pociąga za sobą żadnych dodatkowych obliczeń, a co za tym idzie, można opisać ruch każdej kości z osobna niezależnie od pozostałych kości szkieletu.

4.1. Istota animacji szkieletowej

Aby opisać ruch pojedynczej kości, wykorzystano w implementacji metodę klatek kluczowych (ang. *key frame method*). Polega ona na przechowywaniu informacji o transformacji

tylko w niektórych momentach określanych klatkami kluczowymi. W sytuacji gdy zachodzi potrzeba dostarczenia transformacji kości dla klatki znajdującej się między klatkami kluczowymi, następuje interpolacja transformacji między tymi klatkami. Tak przechowywane klatki kluczowe tworzą tzw. ścieżkę kości bądź trajektorię ruchu kości (ang. *bone track*), czyli opis ruchu danej kości w animacji.

Podsumowując, animacja szkieletowa składa się ze zbioru ścieżek kości (ścieżki występują tylko dla kości biorących udział w animacji), a te ostatnie składają się ze zbioru klatek kluczowych przechowujących transformacje kości.

4.2. Czas życia animacji

Aby animacja mogła wpływać na transformacje kości, a co za tym idzie na wierzchołki modelu, należy ją włączyć, tj. podać do przetwarzania listę jej ścieżek kości. Samo przetwarzanie polega na ustawianiu nowej transformacji kości na zadaną chwilę. W przypadku gdy podana chwila przekracza czas trwania całej animacji, następuje zakończenie przetwarzania animacji.

Dodatkowym elementem związanym z uruchamianiem oraz wyłączaniem animacji jest stopniowe wykonywanie wymienionych procesów. Włączając animację, można podać czas, przez który powinno trwać włączanie animacji, i aż do osiągnięcia tego czasu wszelkie zmiany wprowadzane do szkieletu przez tę animację są zmniejszane proporcjonalnie do ilości czasu pozostałego do pełnego włączenia. Analogiczna sytuacja może mieć miejsce przy wyłączaniu animacji (tj. stopniowe wyłączenie).

4.3. Składanie animacji

Oczywisty jest fakt, że w tym samym czasie na kość może wpływać wiele animacji. W celu obliczenia transformacji kości, w przypadku gdy działa na nią więcej niż jedna animacja, należy uśrednić uzyskane interpolowane wartości transformacji, pochodzące z różnych animacji. Wprowadzenie dodatkowo wag do animacji pozwoli na kontrolę procesu animowania w szerszym zakresie.

4.4. Klasyfikacja animacji ze względu na sposób działania

Biorąc pod uwagę sposób działania animacji, mogą zostać wyróżnione dwa przypadki działania animacji:

- Animacje, które mogą działać właściwie przez cały czas. Wygląd modelu przy takiej animacji nie będzie odbiegał od normy (np. animacja stania, chodu, biegu). Tego typu animacje to tzw. animacje cykliczne (ang. *cycle animation*). Klatki kluczowe w ścieżkach

kości takich animacji tworzą zwykle cykle, dzięki czemu animacja może być ciągle włączona bez szkody dla wyglądu modelu. W praktyce, animacja po zakończeniu swojego czasu działania nie jest wyłączana i włączana na nowo, lecz następuje przejście na początek listy klatek kluczowych.

- Animacje, które powinny być uruchamiane tylko w szczególnych przypadkach i zaraz wyłączone (np. animacja wyciągnięcia ręki do przywitania). Te animacje nazywane są animacjami akcyjnymi lub w skrócie akcjami (ang. *action animation*). Włączane są okazjonalnie i po zakończeniu swoich czasów trwania są wyłączane. Należy zwrócić uwagę, że ze względów wizualizacyjnych animacje akcyjne powinny mieć znacznie większą wagę od cyklicznych, wynika to z faktu, że skoro akcja zostaje włączona, to po to, aby miała widoczny efekt, a nie została zagubiona między włączonymi uprzednio animacjami cyklicznymi.

4.5. Klasyfikacja animacji ze względu na źródło pochodzenia

Dane szkieletowe, podobnie jak dane modelu, w znacznej większości pochodzą z tego samego źródła, tzn. są tworzone przez projektantów modeli przy użyciu specjalistycznych narzędzi (np. programu *3D studio MAX* firmy *Discreet*).

Do tego typu danych szkieletowych należą struktura szkieletu, początkowe wartości transformacji kości oraz animacje. Animacje tworzone podczas projektowania modelu są zwykle dopracowane wizualnie. Jednak na etapie projektowania nie można przewidzieć wszystkich możliwych transformacji kości, jakie mogą być osiągnięte w trakcie wykonania.

Rozwiązaniem tego problemu jest generowanie dynamiczne animacji w czasie wykonania, tj. wtedy, gdy znana jest oczekiwana transformacja kości. Animacje generowane dynamicznie (lub w skrócie animacje dynamiczne) pozwalają na precyzyjne kontrolowanie zachowania szkieletu, a co za tym idzie, także modelu.

5. Dynamiczne generowanie animacji

W ramach tworzenia dynamicznego animacji można wyróżnić następujące etapy generacji i użycia:

- utworzenie podstawowej ścieżki animacji,
- przetworzenie wygenerowanej ścieżki animacji,
- złożenie przetworzonych ścieżek do jednej animacji,
- włączenie animacji jako cyklu lub jako akcji.

Wszystkie, z wyjątkiem pierwszego, wymienione etapy tworzenia animacji są opcjonalne. Pozwalają one programiście na zwiększenie kontroli nad uzyskiwanym wizualnym efektem, natomiast nie są niezbędne do utworzenia danych pozwalających na animowanie modelu.

5.1. Utworzenie podstawowej ścieżki animacji

Podstawowym etapem generowania animacji jest utworzenie podstawowej ścieżki animacji, ponieważ właśnie wtedy następuje generacja transformacji kości poruszanych w generowanej animacji. Transformacje kości w klatkach kluczowych mogą zostać utworzone poprzez:

- skopiowanie transformacji ścieżki z innej istniejącej już animacji,
- rozwiązanie problemu kinematyki prostej,
- rozwiązanie problemu kinematyki odwrotnej.

5.1.1. Kopiowanie transformacji ścieżki z już istniejącej animacji

Ten sposób tworzenia klatek kluczowych jest wykorzystywany, gdy wymagana jest modyfikacja istniejącej animacji. Wtedy pierwszym etapem jest właśnie skopiowanie klatek kluczowych celem dalszej obróbki.

5.1.2. Rozwiązanie problemu kinematyki prostej

Problem kinematyki prostej (ang. *forward kinematics*) polega na obliczeniu zmiany orientacji kości (obrócenia kości) o zadany kwaternion (zadany kąt wokół zadanej osi) oraz obliczeniu transformacji tej kości i jej dzieci w czasie oraz po wykonaniu tego obrotu.

Wydawać by się mogło, że skoro – zgodnie z punktem 3 – wartości przechowywanych transformacji kości są wyznaczone w układzie odniesienia wyznaczonym przez rodzica tej kości, to problem sprowadza się do obliczenia nowej transformacji kości względem jej rodzica po obróceniu tej kości zadanym kwaternionem. Tak też jest w istocie, jeśli zadany kwaternion jest także podany w układzie odniesienia związanym z rodzicem danej kości. Taka sytuacja raczej nie występuje (z wyjątkiem przypadku obracania kości korzeniowej szkieletu), ponieważ obrót zwykle będzie przyjmował wartości z przestrzeni całego modelu, należy zatem uprzednio przeliczyć zadany kwaternion do układu odniesienia wyznaczanego przez rodzica danej kości.

Należy być świadomym, na czym polega obracanie względnej transformacji kości. Obrócenie kości względem jej rodzica sprowadza się do obrócenia jej wektora translacji oraz do obrócenia jej kwaternionu rotacji. Pierwszy krok powoduje widoczny obrót kości, drugi z kolei jest wymagany, by kości-dzieci także wykonały obrót.

5.1.3. Rozwiązanie problemu kinematyki odwrotnej

Bardziej skomplikowanym problemem do rozwiązania jest problem kinematyki odwrotnej (ang. *inversed kinematics*). Problem polega na takim doborze orientacji kości kolejnych przodków, aby dana kość osiągnęła żądaną pozycję w przestrzeni modelu.

Przed opisem sposobu rozwiązania tego problemu należy zdefiniować kilka pojęć:

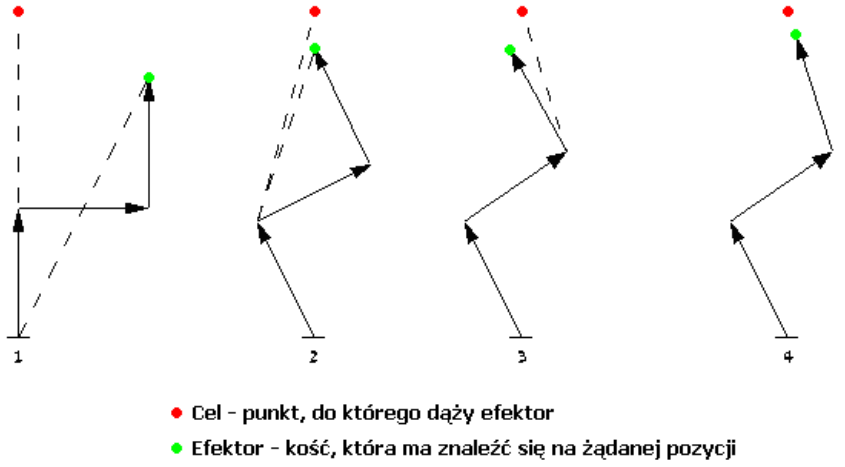
- efektor – kość, która ma znaleźć się na zadanej pozycji,
- łańcuch kości – ciąg kości rodziców rozpoczynający się od efektora, który będzie brał udział w przesunięciu efektora do zadanej pozycji,
- cel – punkt, do którego ma zostać przesunięty efektor.

Podczas rozwiązywania problemu kinematyki odwrotnej należy uwzględnić fakt, że znalezienie rozwiązania nie zawsze jest możliwe do zrealizowania, a nawet jeśli jest, to znalezione rozwiązanie nie musi być tym najlepszym. Można wyróżnić kilka metod rozwiązania tego problemu. Metody dokładne, opisane w [4], sprowadzają się do obliczeń całkowych, przez co wymagają wysokich mocy obliczeniowych. Moc ta nie jest dostępna podczas tworzenia multimedialnych prezentacji (jest zarezerwowana na inne elementy prezentacji), więc to rozwiązanie nie może zostać wykorzystane. Z drugiej strony, warte zastosowania są metody przybliżone, które dają wynik akceptowalny i – co równie ważne – dają go szybko.

Zastosowana przez autora metoda przybliżona (wzorowana na algorytmie cyklicznego dochodzenia do współrzędnych, ang. CCD – Cyclic Coordinate Descent [9]) działa na podobnych zasadach jak iteracyjne metody rozwiązywania układów równań liniowych (np. metoda Seidla); po wyznaczeniu kości wchodzących w skład łańcucha wykonywane są następujące operacje dla każdej z kości łańcucha (rys. 4):

- wyznaczenie wektora od pozycji bieżącej kości łańcucha do efektora,
- wyznaczenie wektora od pozycji bieżącej kości łańcucha do celu,
- wykonanie obrotu bieżącą kością, tak by wyznaczone wektory się zrównały.

Wymienione operacje wykonywane są dla każdej kości w łańcuchu, rozpoczynając od kości najstarszej. Całość powinna być powtarzana dopóty, dopóki efektor nie znajdzie się w pobliżu celu lub w wyniku kroków iteracyjnych algorytmu nie zostaną już wyznaczone żadne zmiany w rotacjach kości.



Rys. 4. Kolejne operacje na łańcuchu kości; 4 – wynik iteracji
Fig. 4. Sequence of operations on the bone chain; 4 – the result of the iteration

Podobnie jak przy rozwiązywaniu problemu kinematyki prostej, także przy rozwiązywaniu tego problemu należy zwrócić uwagę na układy współrzędnych, w których przeliczane są poszczególne transformacje.

5.1.4. Usprawnienia algorytmu iteracyjnego

W oryginalnym algorytmie CCD każda iteracja zaczyna się od najmłodszej kości łańcucha. Dzięki rozpoczęciu iterowania od najstarszej kości łańcucha od razu na początku działania algorytmu efektor może zostać przeniesiony w pobliże celu. Wada tego rozwiązania jest widoczna, gdy łańcuch na początku jest wyprostowany, a cel znajduje się bliżej, niż suma długości kości w łańcuchu – wtedy po pierwszym obrocie cały łańcuch znajdzie się na prostej łączącej cel, efektor oraz podstawę łańcucha, a kolejne ruchy będą wykonywane poprzez gwałtowne obroty o 180^{O}, co nie wygląda realistycznie.

Tego efektu można uniknąć poprzez ograniczenie kąta obrotu kości w pojedynczej iteracji. Dobre efekty otrzymano przy ograniczeniu do $10\text{-}20^{O}$.

Ponieważ przed przystąpieniem do obliczenia należy znać długość łańcucha kości, który posłuży do przesunięcia efektora, warto wstępnie oszacować szukaną wielkość. Proponowanym szacunkiem jest minimalna liczba kości, których suma długości jest większa od odległości celu do podstawy łańcucha.

Kolejnym elementem wartym zautomatyzowania jest maksymalna liczba iteracji (przejść przez cały łańcuch kości) potrzebna do przesunięcia efektora. Szacunek wynika z kąta, o jaki należy obrócić najstarszą kość w łańcuchu, podzielonego przez ograniczenie kąta obrotu na jedną iterację. Dodatkowo, na podstawie doświadczeń uważa się, że dla podwyższenia szansy zakończenia algorytmu poprawnym wynikiem warto tak uzyskany wynik pomnożyć przez 2.

5.1.5. Uwagi do efektów uzyskiwanych za pomocą prostej i odwrotnej kinematyki

Projektant modelu podczas pracy nad nim może zawsze przekonać się, czy utworzone przez niego animacje nie zniekształcają siatek modelu w nienaturalny sposób, przez co wyglądają nierealistycznie. Podczas generowania dynamicznego animacji nie można stwierdzić, jak będzie wyglądała utworzona animacja, dlatego w celu wymuszenia zachowania umiaru w obracaniu kośćmi należy do danych szkieletowych dodać informacje o granicach możliwych zgięć kości – analogicznie do ograniczeń, jakie na biologiczny szkielet nakładają stawy.

Ponadto, należy także oszacować czas trwania animacji, uzależniając go od kąta obrotu, jaki pokonują kości w czasie animacji. Z pracy związanej z weryfikacją informacji opisanych w [13] wynika, że prędkość kątowa 3 rad/s pozwala na uzyskanie zadowalających rezultatów.

5.2. Przetworzenie wygenerowanej ścieżki animacji

Po utworzeniu podstawowej ścieżki animacji kolejnym krokiem tworzenia animacji jest opcjonalne dokonanie pewnych zmian w wygenerowanej ścieżce. Głównym elementem tego etapu jest odgórne ustalenie czasu trwania ścieżki, poprzez przeskalowanie znanego już czasu trwania ścieżki, lub przez ustawienie nowego czasu trwania.

Innym wartym uwagi elementem jest możliwość złożenia ścieżek ze sobą, zarówno jako dołączenie jednej ścieżki na koniec drugiej, jak i jako wzajemny przeplot klatek kluczowych z obu ścieżek. Ostatnim, choć nie najmniej ważnym, elementem, na który warto zwrócić uwagę na tym etapie, jest utworzenie cyklu. Cykl można utworzyć na dwa sposoby: albo dodając na koniec ścieżki kilka klatek kluczowych doprowadzających transformacje z klatki końcowej do klatki początkowej, albo przez lustrzane odbicie klatek kluczowych, tzn. skopiowanie klatek kluczowych ścieżki i umieszczenie ich na końcu w odwrotnej kolejności.

5.3. Złożenie przetworzonych ścieżek do jednej animacji

Uzyskane ścieżki kluczowe dla różnych kości należy po przetworzeniu zgrupować w animację.

Na tym etapie można dołożyć kolejne efekty już do całej animacji, m.in. wspomniane w poprzednim punkcie generowanie cyklu, tym razem jednak w odniesieniu do wszystkich ścieżek zgrupowanych w animacji, oraz manipulacje czasem trwania animacji. Wydaje się jednak, że bardziej wartościową operacją wykonywaną w tym kroku może być ustalenie przesunięć czasowych między poszczególnymi ścieżkami kości.

5.4. Uruchomienie animacji jako cyklu lub jako akcji

Wygenerowana animacja może już zostać uruchomiona, w tym miejscu animacja ma wszelkie cechy animacji dostarczonej do modelu przez projektanta. Nic nie stoi na przeszkodzie, aby uruchomić animację wygenerowaną dynamicznie jako cykl lub jako akcję. Należy jednak pamiętać, że dynamiczna generacja animacji nie zawsze pozwala na uzyskanie zadowalających efektów, a z poziomu kodu nie ma możliwości oceny spodziewanego efektu wizualnego. Z tego też powodu preferowanym obszarem zastosowań animacji generowanych dynamicznie są różnego rodzaju akcje wykonywane przez model. Animacje cykliczne wymagają bowiem na ogół bardziej złożonych ruchów, odnoszą się najczęściej do całego szkieletu (np. chód, bieg), co sprawia, że ich dynamiczne wygenerowanie staje się bardzo trudne.

6. Uwagi końcowe

Podsumowując, można stwierdzić, że wykorzystanie danych szkieletowych podczas animowania modelu trójwymiarowego znacznie ułatwia wykonanie tego zadania.

Autorzy chcieliby również zwrócić uwagę, że w przypadku wykorzystywania animacji generowanych dynamicznie najlepiej jest ich używać okazjonalnie, starając się łączyć ze sobą zarówno animacje generowane, jak i animacje dostarczone na etapie projektowania modelu. Warto także zadbać, aby podczas uruchamiania akcji wygenerowanej dynamicznie, na modelu była wykonywana jakaś animacja cykliczna, co pozwoli przynajmniej na częściowe ukrycie niedoróbek w utworzonej animacji.

LITERATURA

1. Grudziński T., Mysłek T., Ross J.: Wykrywanie kolizji obiektów trójwymiarowych w środowisku FRS, (w przygotowaniu).
2. Mysłek T.: Techniki trójwymiarowych animacji szkieletowych, dynamika modeli trójwymiarowych, Praca magisterska, Politechnika Śląska, Instytut Informatyki, 2005.
3. Grudziński T.: Rendering animowanych obiektów trójwymiarowych realizowany w czasie rzeczywistym, Praca magisterska, Politechnika Śląska, Instytut Informatyki, 2003.
4. Craig J. J.: Wprowadzenie do robotyki. Mechanika i sterowanie. WNT, Warszawa 1995.
5. Lander J.: Skin Them Bones: Game Programming for the Web Generation, Game Developer Magazine, May 1998, dostępne pod adresem http://www.gamasutra.com
6. Lander J.: Making Kine More Flexible, Game Developer Magazine, November 1998, s. 15÷22, dostępne pod adresem: http://www.darwin3d.com/gamedev/articles/col1198.pdf

7. Weber J.: Run-Time Skin Deformation, Game Developers Conference Proceedings (GDC 2000), s. 703÷721, dostępne pod adresem http://www.imonk.com/baboon/bones
8. Bobick N.: Rotating Objects Using Quaternions, Game Developer Magazine, February 1998, s. 34÷42, dostępne pod adresem: http://www.gamasutra.com
9. Weber J.: Ograniczona kinematyka odwrotna (IK), Perełki programowania gier. Vademecum profesjonalisty. Tom 3, Helion 2003

Recenzent: Dr hab. inż. Maria Pietruszka, Prof. Pol. Łódzkiej

Wpłynęło do Redakcji 2 stycznia 2006 r.

Abstract

The paper presents the idea of the animation of an 3-dimensional model by using its skeleton data.

It discusses the simplification of an animation of a model, that can be achieved by adding a skeleton (fig. 2). Moreover, it explains the hierarchical structure of bones and how bones' transformations influence model data.

Most general math operations that are used in making calculations on transformations in 3-dimensional space are mentioned either, along with the way, how bones' transformations are calculated to receive data suitable for modifying model vertices (fig. 3).

The idea of animating model by using its skeleton data and particularly the ways of starting and finishing animations of different animation types are discussed in section 4.

Moreover, beside describing the primary area of use of the skeleton animations, the possibilities of dynamic animation generation, i.e. by solving the problem of forward or inversed kinematics, are introduced in section 5. Section 5.1.3. shows the idea of solving the inversed kinematics problem.

Adresy

Tomasz GRUDZIŃSKI: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Polska, tomasz.grudzinski@polsl.pl.

Jacek ROSS: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Polska, jacek@artifexmundi.com.

Tomasz MYSŁEK: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Polska, kwll@poczta.onet.pl.