Piotr KOPNIAK

BUDOWA MODELI 3D W EDYTORZE BLENDER DLA SILNIKA GIER jMonkeyEngine

STRESZCZENIE *Celem artykułu jest przedstawienie mechanizmu importu modeli 3D postaci stworzonych w edytorze Blender przez silnik do tworzenia gier jMonkeyEngine. W silniku tym zaimplementowano funkcje importu modeli 3D. Razem z siatką modelu importowane są także tekstury, szkielety a nawet całe animacje. Silnik posiada także funkcje eksportu i importu modeli przy wykorzystaniu własnego formatu pliku XML (jME XML). Po zastosowaniu odpowiednich wtyczek do darmowego edytora grafiki 3D Blender możliwe jest wykorzystanie go na potrzeby tworzenia modeli postaci dla silnika jMonkeyEngine. Modele te mogą być także wykorzystane do analizy zagrożeń elektromagnetycznych dla ciała człowieka za pomocą takich programów jak FEKO, Opera3D czy Fulx3D.*

W artykule przedstawiono sposób wykonania modelu postaci 3D w edytorze Blender, dodania do postaci szkieletu oraz eksportu modelu do pliku jME XML oraz AutoCAD DXF, który może być wykorzystany w aplikacjach elektrotechnicznych. Artykuł zawiera także opis przygotowania programu w języku Java wykorzystującego silnik jMonkeyEngine do wizualizacji i animacji modeli postaci importowanych z plików XML.

Słowa kluczowe: *grafika 3D, silnik gry, język Java, MES.*

---

dr inż. Piotr KOPNIAK
e-mail: p.kopniak@pollub.pl

Zakład Ochrony Informacji,
Instytut Informatyki,
Politechnika Lubelska

PRACE INSTYTUTU ELEKTROTECHNIKI, zeszyt 249, 2011

1. WSTĘP

Grafika 3D jest obecnie domeną gier, przemysłu filmowego oraz różnego rodzaju symulacji badawczych [1]. Modele 3D wykorzystuje się także do analizy ruchu człowieka w zastosowaniach medycznych [2, 3] oraz elektrotechnicznych. Do zastosowań elektrotechnicznych zaliczyć można analizę negatywnego wpływu na ciało człowieka pola elektromagnetycznego generowanego przez różnego rodzaju nadajniki radiowe, domowe urządzenia elektryczne oraz urządzenia diagnostyki medycznej [4, 5].

W naszych badaniach grafika 3D wykorzystywana jest do wizualizacji rzeczywistego ruchu postaci ludzkich na podstawie danych uzyskiwanych z systemów przechwytywania ruchu (ang. Motion Capture). Tego typu dane są podstawą animacji postaci filmowych i bohaterów gier komputerowych gdzie realizm ruchów ma kluczowe znaczenie.

Do tworzenia modeli i animacji 3D mogą być wykorzystywane specjalizowane narzędzia jak 3d Studio, Blender, Maya czy Motion Builder. W przypadku programowania gier oraz badań związanych z analizą ruchu, najlepszym i często jedynym wyjściem jest samodzielne stworzenie aplikacji animującej trójwymiarowe modele. Takie podejście umożliwia wykorzystanie danych do animacji pochodzących z różnych źródeł zewnętrznych, np. z systemów Motion Capture.

Aplikację wizualizującą można stworzyć od podstaw wykorzystując bezpośrednio biblioteki przetwarzania grafiki 3D jak OpenGL lub DirectX, albo wykorzystać gotowy silnik gry, np. Cube 2 czy jMonkeyEngine. Wykorzystanie silnika gry ma tą przewagę, że uzyskuje się wiele gotowych funkcjonalności, których nie musimy implementować samodzielnie. Do takich funkcjonalności można zaliczyć możliwość importu modeli 3D stworzonych w różnych edytorach, możliwość łatwego zarządzania obiektami i ich zależnościami, wykrywanie kolizji obiektów, algorytmy fizyki czy interfejsy obsługi manipulatorów do sterowania ruchem obiektów, kamer a także oświetlenia [6].

W celu stworzenia aplikacji wizualizującej ruch w języku Java można wykorzystać w tym celu silnik gier jMonkeyEngine oparty na bibliotece OpenGL, zaś modele animowanych postaci można przygotować w darmowym edytorze grafiki 3D – Blender czego dotyczy niniejszy artykuł. Rozdział 4 zawiera dodatkowo przykład wykorzystania trójwymiarowego modelu ciała człowieka do analizy prądów indukowanych przez pole elektromagnetyczne za pomocą metody elementów skończonych (MES) lub metody elementów brzegowych (MEB).

2. BLENDER

Obecnie istnieje wiele komercyjnych i darmowych narzędzi do tworzenia modeli 3D oraz ich animacji. Jednym z najbardziej popularnych edytorów jest darmowe narzędzie Blender. Edytor ten umożliwia przygotowywanie modeli od podstaw na podstawie brył prostych takich jak walec, czy kula, oraz bardziej skomplikowane operacje jak definicje własnych siatek wierzchołków i rzeźbienie w bryłach. Stworzone siatki reprezentujące przestrzenne bryły mogą być następnie odpowiednio teksturowane oraz oświetlane. Blender posiada także zaawansowane funkcje animacji, detekcji kolizji oraz własny silnik gier korzystający z budowanych obiektów i animacji [7, 8].

Animacja postaci w Blenderze może zostać wykonana poprzez rejestrację przesunięć i obrotów poszczególnych brył składających się na animowany obiekt lub poprzez wykorzystanie ruchów szkieletu postaci. Wykorzystanie szkieletu jest bardzo wygodne, ponieważ wystarczy dodać do modelu szkielet, następnie związać go z bryłą postaci oraz wykonać animację szkieletu, a odpowiednia animacja bryły zostanie wykonana automatycznie.

W opisywanym przypadku przyjęto, że animacja obiektu będzie przeprowadzana poza edytorem grafiki 3D, czyli w aplikacji wizualizującej ruch napisanej w języku Java i wykorzystującej silnik gier jMonkeyEngine. W związku z tym w Blenderze przygotowano jedynie statyczny model 3D sylwetki człowieka wyposażony w szkielet, który będzie animowany przy wykorzystaniu silnika gier.

2.1. Budowa modelu

Pracę w edytorze Blender rozpoczynamy od utworzenia nowego projektu poprzez wybór opcji *New* z menu *File*. W wyniku tej operacji otrzymujemy nową scenę, na którą składają się trzy obiekty: sześcian, źródło światła oraz kamera. W celu dodania bryły do modelu wystarczy wybrać opcję z menu *Add / Mesh*, a następnie wybrać jedną z proponowanych brył. Do wyboru mamy: kwadratową powierzchnię (Plain), sześcian (Cube), koło (Citcle), dwa rodzaje sfer zbudowane z prostokątów i trójkątów (UVSphere i IcoSphere), walec (Cylinder), stożek (Cone) oraz płaską siatkę połączonych krawędziami wierzchołków (Grid).

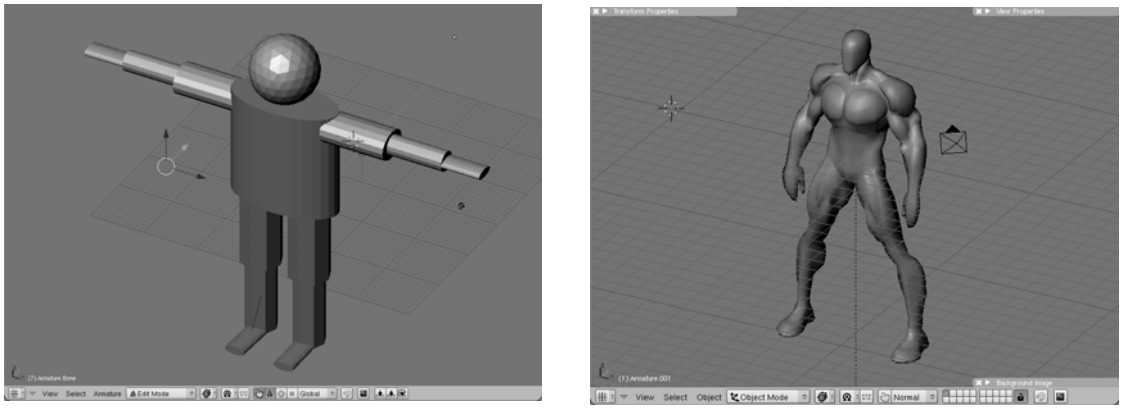
Postać do animacji można wykonać z brył prostych lub posługując się siatkami wierzchołków, które mogą być dowolnie deformowane. Postaci zbudowane z brył prostych i siatki wierzchołków przedstawia rysunek 1.

Postać człowieka przedstawiona na rysunku 1b wykonana została poprzez przekształcenie gotowego darmowego modelu Cyborg ninja [9]. Dla stworzonej

siatki włączone zostało wygładzanie powierzchni, źródło światła typu słonecznego oraz jeden materiał. Model nie został pokryty teksturą w celu lepszej wizualizacji ruchu.

Zarówno modele stworzone z brył prostych jak i modele zbudowane z siatek wierzchołków można łatwo edytować. Wystarczy zmienić tryb pracy edytora z trybu obiektowego (Object Mode), umożliwiającego operacje translacji, rotacji i skalowania całych brył, na tryb edycji (Edit Mode). Opcje zmiany trybu dostępne są w liście rozwijanej na dolnym pasku narzędziowym okna edycji sceny.

a) b)



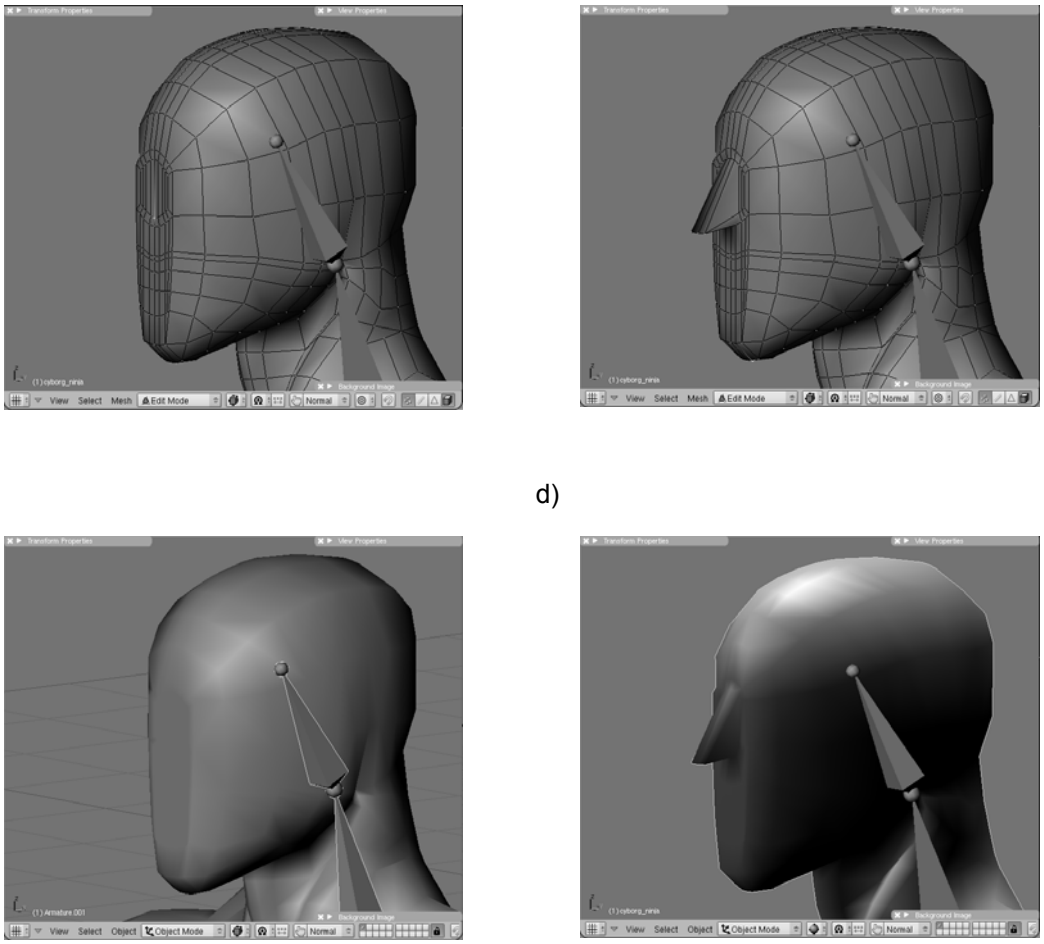
Rys. 1. Postacie stworzone w edytorze Blender,
a) postać wykonana z brył prostych, b) postać wykonana jako siatka wierzchołków

W trybie edycji bryły można modyfikować położenie przestrzenne poszczególnych wierzchołków, krawędzi lub całych wielokątów będących ściankami bryły. Przemieszczenie wymienionych elementów wykonuje się poprze przeciąganie ich w inne miejsce w przestrzeni za pomocą myszy. Można także powodować przekształcenia grupy sąsiadujących wielokątów jednocześnie, co daje efekt miejscowego rozciągania bryły. Operacja ta nosi nazwę rzeźbienia bryły i jest dostępna po zmianie trybu pracy edytora na tryb rzeźbienia (Sculpt Mode). Przykładową operację zmiany położenia wierzchołków przedstawiono na rysunku 2. W jej wyniku twarz człowieka uzupełniono o nos.

Wykonanie animacji postaci zbudowanej jako jedna siatka wierzchołków wymaga dodania do niej szkieletu. Szkielet buduje się z kości posiadających stawy na swoich zakończeniach. Kości dodaje się poprzez wybranie z menu opcji *Add*, a następnie podopcji *Armature*. Edytor tworzy pierwszą kość, którą możemy powielać wciskając klawisz „E” na klawiaturze. W ten sposób możemy stworzyć całe kończyny. Kończyny należy dołączyć do pozostałej części szkieletu, np. kości biodrowych poprzez określenie hierarchii rodzic-dziecko. Związki tego typu pomiędzy poszczególnymi segmentami szkieletu wykonujemy

poprzez zaznaczenie dwóch obiektów i wciśnięcie kombinacji klawiszy „Ctrl+p". Relacja rodzic-dziecko może zostać utworzona także pomiędzy segmentami nie połączonymi ze sobą za pomocą stawów.

a) b)

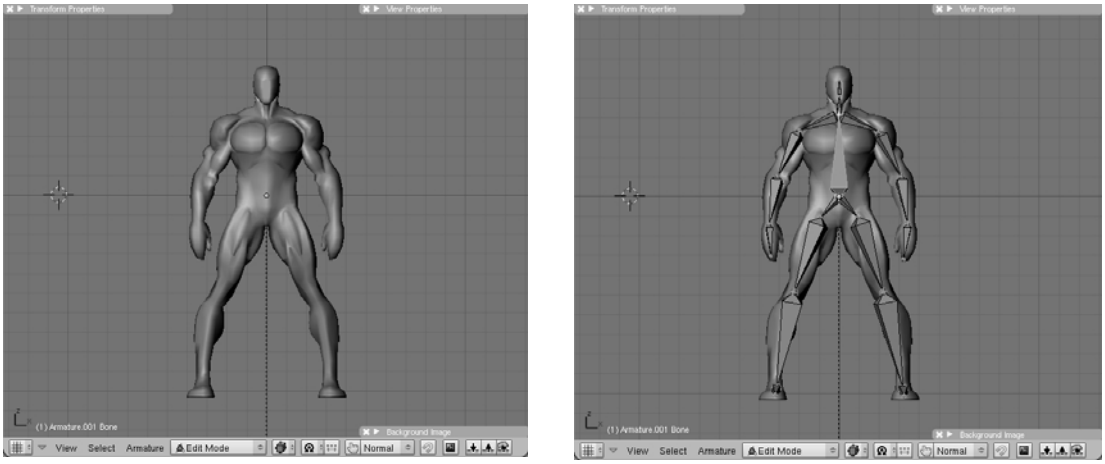


c) d)

Rys. 2. Edycja modelu poprzez przesuwanie wybranych wierzchołków bryły:
a) siatka wyjściowej bryły twarzy bez nosa, b) siatka po dodaniu nosa, c) oświetlona i wygładzona bryła twarzy bez nosa, d) oświetlona i wygładzona bryła twarzy z dodanym nosem

Żeby kości były widoczne poprzez otaczające je ciało należy włączyć prześwietlanie ciała, czyli opcję „X-Ray". Opcję tą włącza się oknie konfiguracyjnym „Armature" dostępnym w oknach narzędziowych edytora po włączeniu przycisku „Editing" z ikoną w kształcie kwadratowej siatki. Daje to efekt widoczny na rysunku 2, gdzie widać kość odpowiadającą za zmiany położenia głowy i jedną kość szyi. Na rysunku 3 widoczny jest już cały szkielet postaci. Działanie szkieletu można zweryfikować zmieniając tryb edytora na tryb póz poprzez wybór opcji „Pose Mode" z listy rozwijanej paska narzędziowego poniżej okna wizualizacji sceny.

a) b)



Rys. 3. Dodawanie szkieletu do postaci:
a) model bez szkieletu, b) model ze szkieletem

Utworzony szkielet można poruszać, chwytając i obracając w stawach kości za pomocą myszy. Kości podrzędne w stosunku do kości poruszanej także zmieniają swoje położenie, jednak nie powoduje to ruchu ciała, do którego szkielet jest dołączony. W celu zapewnienia ruchu ciała, tzn. deformacji bryły odpowiadających ruchom poszczególnych kończyn w ich stawach konieczne jest jeszcze utworzenie relacji rodzic-dziecko pomiędzy ciałem i szkieletem. Relację tą definiuje się w taki sam sposób jak relacje pomiędzy poszczególnymi kośćmi szkieletu.

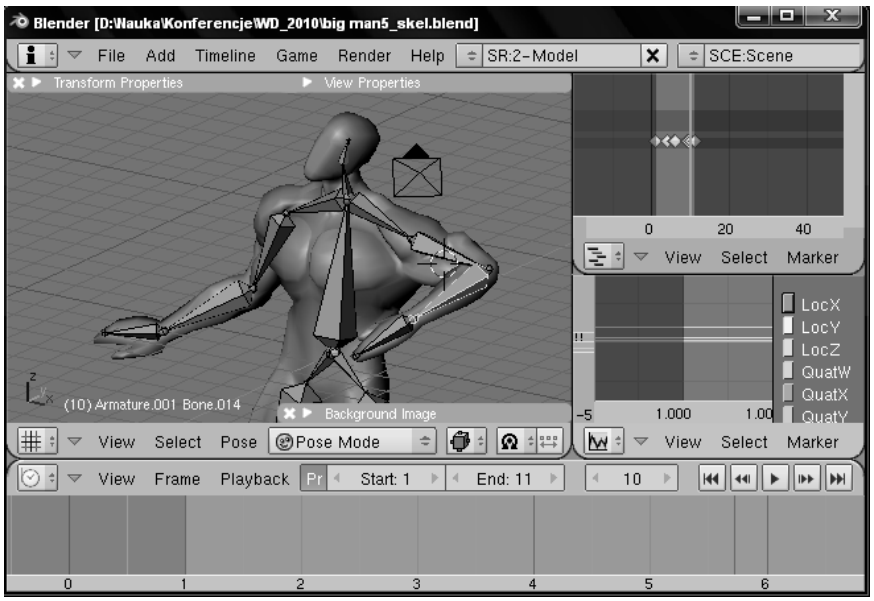
Przygotowany model może zostać poddany animacji w Blenderze. Zapis animacji rozpoczynamy od włączenia trybu póz (Pose Mode). Następnie należy w dowolnym z okien narzędziowych włączyć widok „Timeline”. Jest to panel służący do definicji poszczególnych klatek animacji. Panel „Timeline” daje nam możliwość określenia liczby klatek i ich rozkładu w czasie. Okno zawiera także panel odtwarzacza animacji z przyciskami do uruchamiana, przewijania i zatrzymywania animacji. Blender umożliwia także zarządzanie ruchem poszczególnych kości oraz płynnością tego ruchu. W tym celu wystarczy otworzyć dodatkowe panele edycji animacji, tzn. „NLA Editor” oraz „Ipo Curve Editor”. Okno Blendera z włączonymi trzema wymienionymi panelami edycji animacji przedstawia rysunek 4. Na dole ekranu widzimy panel „Timeline”, a z prawej strony dwa pozostałe panele.

Poszczególne klatki animacji dodajemy poprzez ustawienie znacznika czasu na odpowiednią klatkę w panelu czasu, rotację kości za pomocą myszy oraz zachowanie tej rotacji w postaci klatki animacji. Przypisanie rotacji do klatki wykonuje się z menu kontekstowego dostępnego po wciśnięciu klawisza „I” na

klawiaturze. W tym celu z menu należy wybrać opcję *LocRot*. Na rysunku 4 widoczne są rotacje kości przypisane do 10 klatki stworzonej animacji.

Pomimo tego, że podczas eksportu gotowego modelu do pliku możliwy jest także eksport jego animacji nie zostało to wykorzystane w opisywanym przypadku. W związku z tym, że model został stworzony na potrzeby animacji w aplikacji zewnętrznej korzystającej z silnika gier jMonkeyEngine wyeksportowano jedynie bryłę ciała z kośćcem.

Model został wyeksportowany do pliku jME XML. Jest to format zapisu modeli opracowana dla silnika jMonkeyEngine. Blender nie posiada domyślnie zaimplementowanej funkcji eksportu do tego typu pliku. W związku z tym konieczne było dołączenie odpowiedniej wtyczki - HottBJ Exportera w postaci skryptu napisanego w języku Phyton. Skrypt powoduje dodanie opcji *JMonkeyEngine (*-jme.xml)* do menu *File / Eksport* edytora. Dołączony moduł eksportu ma możliwość ustawienia różnych parametrów eksportu, takich jak np. decyzja o dołączaniu lub nie danych animacji i danych tekstur do wynikowego pliku XML.



Rys. 4. Okno tworzenia animacji w edytorze Blender

3. IMPORT MODELU DO APLIKACJI

Silnik jMonkeyEngine posiada interfejs importu modeli umożliwiający wczytanie modeli 3D łącznie z kośćcem, skórą i teksturami z plików w formatach: COLLADA, 3DS, Obj, MD2, MD3, Milkshape, X3D, ASE, Ogre XML, jME Binary oraz jME XML [10]. Klasy niezbędne do przekonwertowania modeli

pochodzących z różnych aplikacji zewnętrznych na wewnętrzny format jME XML, np. klasy `MaxToJme` i `Md3ToJme` znajdują się w pakiecie `com.jmex.model.XMLparser.Convereters`. Przykładowy fragment programu wczytujący obiekt przygotowany w 3d Studio Max do aplikacji wykorzystującej omawiany silnik gier przedstawia Listing 1.

Listing 1: Fragment programu wczytujący do aplikacji model przygotowany w 3d Studio Max.

```
//import niezbędnych klas silnika
import com.jme.scene.Node;
import com.jmex.model.XMLparser.Converters.MaxToJme;
import com.jme.util.export.binary.BinaryImporter;
…

MaxToJme maxtojme = new MaxToJme();    //stworzenie obiektu konwertera
Node node = null;                      //deklaracja węzła głównego

//uzyskanie adresu URL wczytywanego pliku
URL url = ModelGroup.class.getClassLoader().getResource("Model.3ds");

//utworzenie strumienia wyjściowego dla konwertera
ByteArrayOutputStream bOut = new ByteArrayOutputStream();

//wywołanie funkcji konwertującej pobierającej obiekt ze strumienia
//wejściowego i zwracającej skonwertowany obiekt do strumienia
//wyjściowego
maxtojme.convert(url.openStream(),bOut);

//konwersja obiektu ze strumienia do postaci siatki TriMesh
//i utworzenie węzła reprezentującego model w drzewie obiektów sceny
BinaryImporter binImp = new BinaryImporter();
node = (Node)binImp.load(new
ByteArrayOutputStream(bOut.toByteArray()));
```

W naszym przypadku wykorzystano eksport modelu stworzonego w edytorze Blender od razu do formatu jME XML. Import modelu 3D przygotowanego w rodzimym dla silnika jMonkeyEngine formacie jME XML przebiega dużo prościej. Klasa importu modelu w postaci pliku XML nazywa się `XMLImporter` i pochodzi z pakietu `com.jme.util.export.xml`. Utworzenie instancji klasy i wczytanie modelu wygląda następująco:

Listing 2: Fragment programu wczytujący do aplikacji model przygotowany w Blenderze.

```
//import niezbędnych klas silnika
import com.jme.util.export.xml.XMLImporter;
import com.jme.scene.Spatial;

//utworzenie instancji importera
XMLImporter xmlImporter = XMLImporter.getInstance();
//utworzenie obiektu węzła reprezentującego bryłę geometryczną
Spatial spatial = (Spatial)xmlImporter.load(new File("big_man.xml"));
//dodanie węzła modelu do korzenia drzewa obiektów sceny
rootNode.attachChild(spatial);
```

Powstały obiekt typu `Spatial` reprezentuje korzeń drzewa kości, czyli szkieletu, który może być rozwinięty na podstawie utworzonej w Blenderze hierarchii pomiędzy poszczególnymi kośćmi szkieletu. Dla potrzeb obsługi szkieletu przygotowano w silniku jMonkeyEngine odpowiednie klasy, m.in. klasę `Bone` reprezentującą kość oraz `BoneAnimation` opisującą animacje kości. Przykładowy fragment kodu programu odczytujący dane szkieletu zaimportowanego modelu przedstawia Listing 3:

Listing 3: Fragment programu odczytujący dane szkieletu z zaimportowanego modelu.

```
//utworzenie kolekcji List obiektów typu Spatial będących instancjami
//klasy Bone
List<Spatial> bones = rootNode.descendantMatches(Bone.class);

//dla każdej kości z listy można odczytać listę kości podrzędnych
for(Spatial s: bones){
        Bone b = (Bone)s;
        List<Spatial> subbones = b.getChildren();
}
```

Model postaci człowieka wczytany do aplikacji wizualizacyjnej przedstawia rysunek 5a. Rysunek 5b zawiera jedną klatkę animacji pokazującą zmianę położenia kończyn górnych dzięki funkcji lokalnej rotacji oferowanej przez silnik jMonkeyEngine.

a) b)

Rys. 5. Wizualizacja modelu poprzez silnik jMonekeyEngine:
a) model wyjściowy, b) model po rotacji kończyn

Rotacja wykonana została dla kości obu ramion. Przykładowa linia programu realizująca lokalną rotację kości na podstawie podanych kątów obrotu wygląda następująco:

```
b.getLocalRotation().fromAngles(0f, 0f, 0f);
```

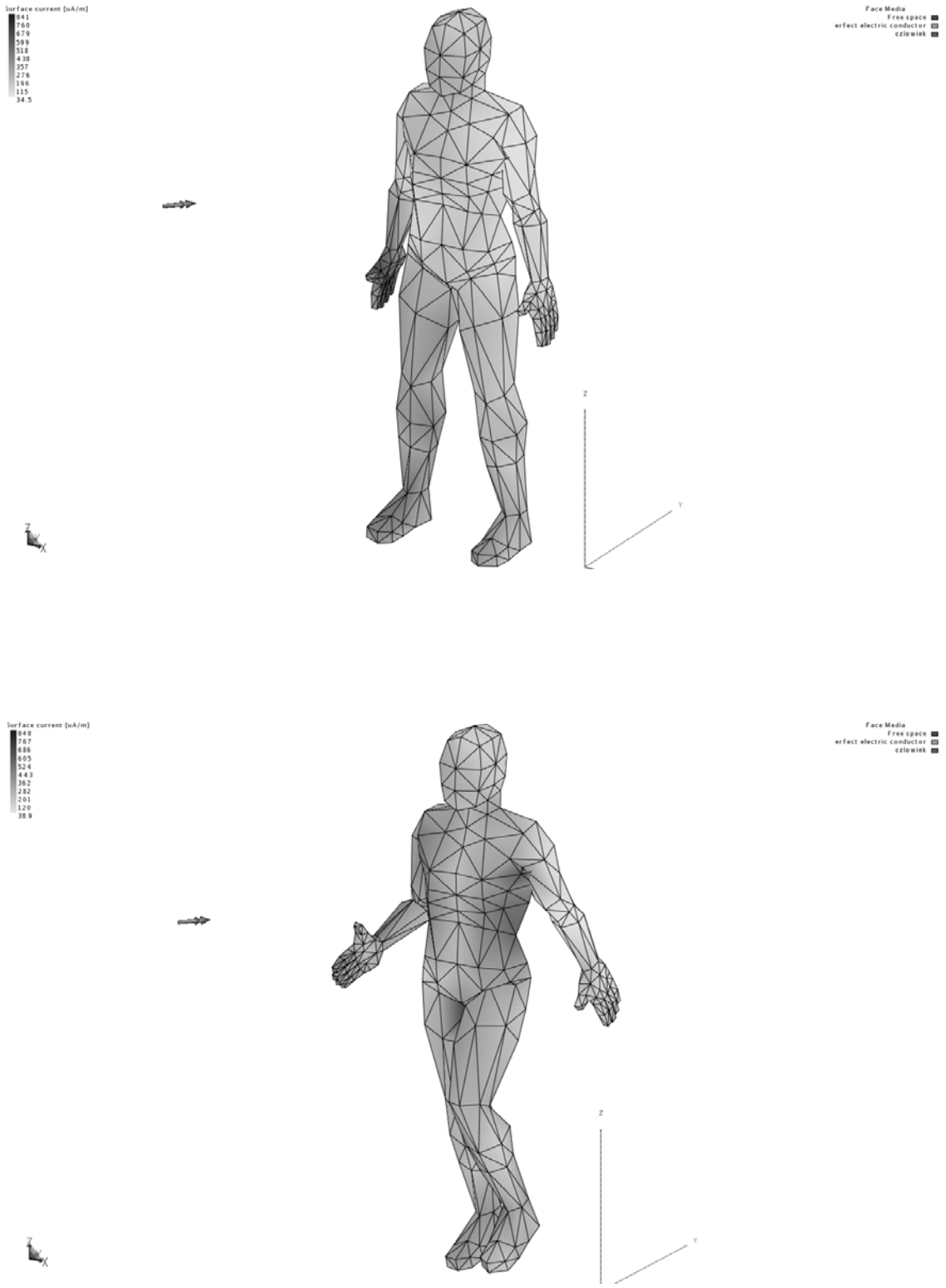
4. WYKORZYSTANIE MODELU DO ANALIZY PRĄDÓW INDUKOWANYCH

Szczegółowo przygotowana w edytorze Blender siatka 3D postaci ludzkiej może być wykorzystana także do analizy pola elektromagnetycznego za pomocą metod MES lub MEB w takich programach jak FEKO, Opera3D czy Flux3D. Pole elektromagnetyczne generowane jest przez różne urządzenia do diagnostyki medycznej jak MRI oraz urządzenia użytku domowego, np. do komunikacji bezprzewodowej. Pole to powoduje indukowanie w ciele ludzkim prądu, który może wpływać niekorzystnie na zdrowie oraz w ekstremalnych przypadkach zagrażać życiu. W związku z tym bardzo istotne jest modelowanie wpływu pola na ciało ludzkie [11, 12].

Wszystkie wyżej wymienione systemy do analizy numerycznej pola posiadają funkcje importu siatek obiektów 3D w formacie AutoCAD, czyli plików DXF. Siatki tego typu można zbudować w Blenderze i wyeksportować do formatu DXF za pomocą funkcji *Export/Autodesk .dxf* znajdującej się w menu *File*. Przy eksporcie można określić parametry eksportu modelu, np. czy ma to być siatka dwu czy trójwymiarowa.

Zaletą przygotowania siatki w edytorze 3D takim jak Blender jest możliwość dodania do postaci szkieletu co było opisane wcześniej. Dzięki temu możemy zmieniać położenie kończyn i analizować numeryczne prądy indukowane przy różnych położeniach ciała w stosunku do źródła pola bez konieczności budowy modelu od nowa w modułach CAD wymienionych aplikacji.

Powyżej na rysunku 6 pokazano dwie siatki przedstawiające postać ludzką w spoczynku oraz ruchu wygenerowane w Blenderze i zaimportowane do pakietu FEKO [13]. Dla modelu ciała ludzkiego określono materiał dielektryczny na podstawie publikacji [4, 5] o względnej przenikalności magnetycznej równej 1, względnej przenikalności elektrycznej równej 75 i przewodności właściwej ciała równej 0,85 S/m. Dla obu siatek przeprowadzono symulację gęstości prądu indukowanego w ciele człowieka pod wpływem punktowego źródła pola magnetycznego oznaczonego strzałką oddalonego od ciała o ok. 1 m. Źródło generowało pole o częstotliwości 70 MHz i amplitudzie 10 Vm. Częstotliwość 70 MHz zawiera się w zakresie częstotliwości 70-110 MHz, który określa częstotliwości rezonansowe dla ciała mężczyzny o wzroście 170 cm [4]. Zmiana położenia modelu w stosunku do źródła pola spowodowała niewielki wzrost maksymalnej wartości gęstości prądu indukowanego z 841 $\mu A/m^2$ do 848 $\mu A/m^2$ oraz inny rozkład prądów, co przedstawia rysunek 6.



Rys. 6. Wizualizacja prądów indukowanych w ciele człowieka w aplikacji FEKO

5. PODSUMOWANIE

Budowa aplikacji wizualizujących ruch postaci w celu jego analizy jest w znacznym stopniu ułatwiona dzięki istniejącym narzędziom do modelowania postaci i ich animacji takim jak Autodesk 3d Max, Maya, czy Blender. Przygotowane w edytorze grafiki 3D modele mogą być wydajnie animowane w samym środowisku edytora lub w wirtualnym środowisku zewnętrznych aplikacji wizualizacyjnych. Możliwe jest to dzięki licznym interfejsom importu i eksportu obiektów.

Własne, wydajne aplikacje wizualizujące ruch obiektów trójwymiarowych można tworzyć szybko dzięki zastosowaniu wysokopoziomowych języków programowania, takich jak język Java oraz silników gier takich jak jMonkeyEngine. Silniki gier oferują nam znacznie więcej możliwości niż tylko zestaw funkcji do tworzenia animacji postaci. Dzięki silnikom gier możliwe jest także wykorzystanie gotowych implementacji licznych algorytmów przetwarzania grafiki 3D jak: dynamiczne cieniowanie, teksturowanie, czy wygładzanie krawędzi. Dzięki wbudowanym mechanizmom w silnikach gier możliwa jest interakcja użytkownika i wizualizowanej sceny, m.in. dzięki wbudowanej obsłudze różnych urządzeń wejściowych (manipulatorów). Silniki gier posiadają także zaimplementowaną obsługę kolizji obiektów i implementacje różnych algorytmów fizycznych, np. uwzględniających grawitację podczas ruchu obiektów. Wymienione funkcje mogą posłużyć do różnego rodzaju badań symulacyjnych, np. związanych z oddziaływaniem utworzonego wcześniej animowanego obiektu na jego otoczenie.

W naszym przypadku wykorzystanie darmowego edytora Blender umożliwiło szybkie przygotowanie bazowego modelu postaci, który posłużył do wizualizacji danych pozyskanych z systemu przechwytywania ruchu. Dzięki dodaniu szkieletu do obiektu w edytorze 3D oraz zaimplementowanym zaawansowanym funkcjom importu modeli w silniku gier możliwe było także szybkie przygotowanie wydajnego środowiska animacyjnego, które posłużyło do wizualizacji pozyskanych danych o rzeczywistym ruchu człowieka. Wykorzystany silnik jMonkeyEngine doskonale się nadaje do tego typu zadań. Dodatkowo dzięki wykorzystaniu w procesie programowania uniwersalnego języka, jakim jest język Java, uzyskano przenośność międzysystemową stworzonego oprogramowania wizualizacyjnego.

Przygotowane w Blenderze siatki postaci ludzkich mogą być także wykorzystane w aplikacjach MES do analizy wpływu pola elektromagnetycz-

nego na człowieka. Dzięki funkcjom animacji edytora 3D oraz importu danych z systemów rejestracji ruchu rzeczywistego (ang. Motion Capture) możliwa jest analiza prądów indukowanych w warunkach maksymalnie zbliżonych do rzeczywistych.

LITERATURA

1. Kitagawa M., Windsor B.: MoCap for Artists. Workflow and Techniques for Motion Capture., Elsevier, 2008.
2. Holden J.P., Selbie W.S., Stanhope S.J.: A proposed test to support the clinical movement analysis laboratory. Gait and Posture 17., Elsevier, str. 205-213, 2003.
3. Milner C.E., Hamill J., Davis I.S.: Distinct hip and rearfoot kinematics in female runners with a history of tibial stress fracture. Journal of Orthopaedic & Sports and Physical Theraphy, Volume 40, No. 2, str. 59-66, 2010.
4. Przytulski A.: Efekty termiczne w ciele człowieka wywołane szybkozmiennym polem elektromagnetycznym., Pomiary Automatyka Robotyka 12/2010, str. 145-149, Warszawa, 2010.
5. Zradziński P., inn.: Zasady modelowania zagrożeń elektromagnetycznych. Modelowanie ciała pracownika., Bezpieczeństwo Pracy 10/2006, Centralny Instytut Ochrony Pracy – Państwowy Instytut Badawczy, Warszawa, 2006.
6. jMonkeyEngine Homepage, <http://www.jmonkeyengine.com/>, 23.06.2010.
7. Kuklo K., Kolmaga J.: Blender. Kompendium., Helion, Gliwice, 2007.
8. Mullen T.: Blender. Mistrzowskie Animacje 3D., Helion, Gliwice, 2010.
9. 3D Model Cyborg ninja, <http://t3ds.com/66879>, 23.06.2010.
10. jMonkeyEngine User's giude, <http://www.jmonkeyengine.com/wiki/doku.php/user_s_guide>, 23.06.2010.
11. Scorretti R., Siauve N., Burais N.: Induced currents into the human body by cooking induction systems: The new european standard EN 50366., International Journal of Applied Electromagnetics and Mechanics, IOS Press, Volume 25, Number 1-4 / 2007, str. 413-417, 2007.
12. Dughiero F., Forzan M., Sieni E.: A numerical evaluation of electromagnetic fields exposure on real human body models until 100?kHz., COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering, Vol. 29 Iss: 6, str.1552 – 1561, 2010.
13. FEKO – Comprehensive Electromagnetic Solution, <http://www.feko.info/product-detail/overview-of-feko>, 20.01.2010.

Rękopis dostarczono dnia 19.10.2010 r.

BUILDING OF 3D MODELS FOR JMonkeyEngine GAME ENGINE

Piotr KOPNIAK

ABSTRACT *The aim of the article is a presentation of an import mechanism of 3D character models made in Blender editor for jMonkeyEngine game engine. It has got implemented import functions for 3D models. Meshes, textures, skeletons and whole animations are imported by this engine. The engine has got functions for import and export models with utilization of its own file format based on XML (jME XML), too. A file of this type for jMonkeyEngine may be generated by freeware Blender editor thanks to additional export plug-ins. Models made with Blender we may use for FEM analysis of an electromagnetic field influence to a human body in applications like: FEKO, Opera3D or Fulx3D.*

This paper presents a process of: developing 3D character model in Blender editor, adding a skeleton and exporting into jME XML file and AutoCAD DXF, which may be used for field analysis aplications. The article contains a description of developing process of an application in Java language with utilization of jMonkeyEngine for animation visualization of character models imported from XML files.

Dr inż. Piotr KOPNIAK jest adiunktem w Zakładzie Ochrony Informacji Instytutu Informatyki na Wydziale Elektrotechniki i Informatyki Politechniki Lubelskiej. Ukończył studia wyższe na tym samym wydziale w 1999 roku broniąc pracę magisterską ma temat: „Zestaw apletów i aplikacji Java do zastosowań biznesowych". Od 2004 roku zajmował się tematyką ochrony informacji, a w szczególności ukrywaniem informacji w obrazie cyfrowym. Prace badawcze zakończył w 2007 roku obroną pracy doktorskiej p.t.: „Wykorzystanie metod cyfrowego przetwarzania sygnału w steganologii komputerowej". Od tego czasu zajmuje się badaniami związanymi z przetwarzaniem obrazu i przechwytywaniem ruchu (ang. Motion Capure). Dr inż. Piotr Kopniak obecnie prowadzi w Instytucie Informatyki zajęcia dydaktyczne z przedmiotów: Programowanie w języku Java, Przetwarzanie mobilne oraz Programowanie współbieżne i rozproszone. Dodatkowo jest założycielem i opiekunem koła naukowego Pentagon Café, zajmującego się tworzeniem internetowego systemu wspomagania pracy Instytutu Informatyki.