STUDIA INFORMATICA 2011
Volume 32 Number 2B (97)

Dariusz R. AUGUSTYN, Łukasz WARCHAŁ, Piotr ŻELAŹNICKI
Politechnika Śląska, Instytut Informatyki

TWORZENIE NOWOCZESNYCH APLIKACJI Z TRÓJWYMIAROWYM, GRAFICZNYM INTERFEJSEM UŻYTKOWNIKA, OPARTYM NA TECHNOLOGII CUBE

Streszczenie. Współczesne systemy informatyczne powinny charakteryzować się zaawansowanym graficznym interfejsem użytkownika (GUI), dostosowanym do wyspecjalizowanych potrzeb wynikających z obsługiwanej dziedziny przedmiotowej. Istnieje wiele zastosowań, w których wymagany jest interfejs 3D. Implementacja rozszerzenia aplikacji o interfejs 3D nie jest zadaniem łatwym, biorąc pod uwagę, że na ogół programiści tworzący aplikacje są specjalistami w ramach konkretnej dziedziny przedmiotowej, a nie grafiki komputerowej. W takich przypadkach pomocna może być technologia CUBE. Dostarczając odpowiednie narzędzia i stosunkowo łatwy interfejs programowy – CUBE API – technologia CUBE pozwala na osiągnięcie tego celu. Niniejszy artykuł wprowadza czytelnika w podstawy technologii CUBE. Omawia też wykonaną przez autorów aplikację prototypową z zaawansowanym GUI, opartym na trójwymiarowym modelu ciała człowieka, wykorzystującym komponenty omawianej technologii. Doświadczenia związane z budową tej prototypowej aplikacji pozwalają na weryfikację przydatności technologii CUBE.

Słowa kluczowe: ergonomia obsługi systemów informatycznych, trójwymiarowy interfejs użytkownika, rozszerzanie aplikacji w zakresie obsługi 3D GUI, technologia CUBE

DEVELOPING MODERN APPLICATIONS WITH 3D GUI BASED ON CUBE TECHNOLOGY

Summary. A modern information system should have an advanced ergonomic graphical interface customized to a particular domain. There are many domains where a 3D GUI is required. The 3D GUI extension is not a trivial task, especially for programmers who are rather specialists in a concrete business domain than computer graphics. The CUBE technology may help to introduce the 3D GUI extension into standard information systems. Providing tools and handy CUBE API the described technology may allow to achieve this goal. This paper introduces into the CUBE technology. The paper describes the medical prototype application with a man-body-

model-based 3D GUI (based on the CUBE technology), developed by paper authors. This application allows to verify the usability of the CUBE technology.

Keywords: ergonomic information systems, 3-dimensional user interface, extending applications for enabling 3D GUI, CUBE technology

1. Wstęp

Klasyczne systemy informatyczne, wyposażone nawet w atrakcyjny interfejs użytkownika (ang. *Graphical User Interface*), mogą nie spełniać wymagań w specyficznych zastosowaniach, dotyczących operowania na obiektach przestrzeni trójwymiarowej. Rozszerzanie GUI aplikacji o elementy 3D wymaga specjalnych umiejętności z zakresu technologii obsługi grafiki 3D. Zespoły tworzące aplikacje biznesowe skupione są na realizacji wymagań klienta i nie są na ogół wyspecjalizowane w zakresie grafiki komputerowej.

W takich sytuacjach pomocna może być technologia CUBE, pozwalająca w stosunkowo prosty sposób rozszerzyć możliwości aplikacji dziedzinowych w zakresie 3D GUI. CUBE ukrywa pewne szczegóły programowej obsługi 3D, dostarczając łatwe w obsłudze API. Artykuł opisuje technologię CUBE, architekturę komponentów CUBE oraz przykłady ich wykorzystania.

Technologia CUBE została zaproponowana i opracowana przez firmę 2KMM Sp. z o.o. [1] (we współpracy z firmą Artifex Mundi – Game Development Studio Sp. z o.o.). Technologia CUBE jest własnością firmy 2KMM Sp. z o.o., natomiast autorzy niniejszego artykułu byli konsultantami w czasie tworzenia wymagań do systemu CUBE oraz bezpośrednimi wykonawcami aplikacji prototypowej[1], weryfikującej, w założonym zakresie, przydatność omawianej technologii.

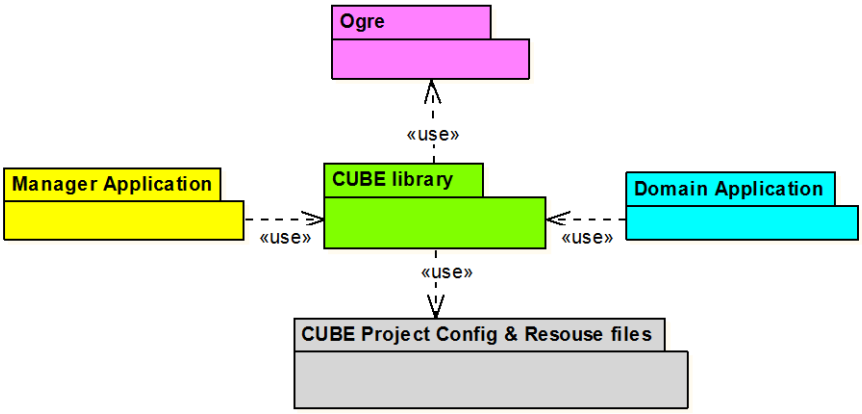
Artykuł omawia technologię CUBE za pomocą przykładów, wskazuje na potencjalne zalety takiego podejścia. Prezentuje pilotową aplikację dziedzinową, dotyczącą zastosowań medycznych, wyposażoną w 3D GUI, korzystającą z modułów CUBE.

2. Wprowadzenie do technologii CUBE

U podstaw technologii CUBE leży założenie o wykorzystaniu zewnętrznego silnika graficznego, służącego do renderowania scen trójwymiarowych. W obecnej, pierwszej wersji CUBE, zakłada się wykorzystanie silnika OGRE [2] (ang. *Object-Oriented Graphics Rendering Engine*), dostępnego na zasadach bardzo liberalnej licencji MIT, działającego na podstawie Direct3D lub OpenGL, na systemach Windows/Linux/Mac OSX. Zakłada się również

[1] Prace w ramach projektu naukowo-badawczego NB – 322/RAu2/2010.

możliwość wykorzystania innego silnika graficznego. Komponenty związane z OGRE zostały oznaczone na rys. 1 i 2 kolorem fioletowym[2].



Rys. 1. Zależności pomiędzy podstawowymi komponentami CUBE
Fig. 1. Dependences among main CUBE components

Moduły biblioteczne CUBE (kolor zielony na rys. 1 i 2) realizują zasadniczą funkcjonalność systemu CUBE. Moduły te wymagane są do uruchomienia zarówno programu przeznaczonego do zarządzania projektem CUBE (aplikacja narzędziowa Manager.exe, nazywana CUBE Designer, oznaczona kolorem żółtym), jak również dowolnej aplikacji dziedzinowej (aplikacji „biznesowej", rozszerzanej w zakresie 3D GUI, oznaczonej kolorem niebieskim).

Zestaw obiektów 3D może być utworzony za pomocą niezależnego programu, np. takiego jak narzędzie Blender [3] (narzędzie typu *free open source 3D content creation suite*), dostępne na licencji GNU. Po zainstalowaniu obsługi języka Python, możliwy jest eksport wykonanego w Blenderze projektu do pliku w uniwersalnym formacie Collada [5], tzn. pliku *.dae* (rys. 3), importowalnego później w narzędziu CUBE Designer. Czynność ta została pokazana na rys. 3 (czynność nr 1).

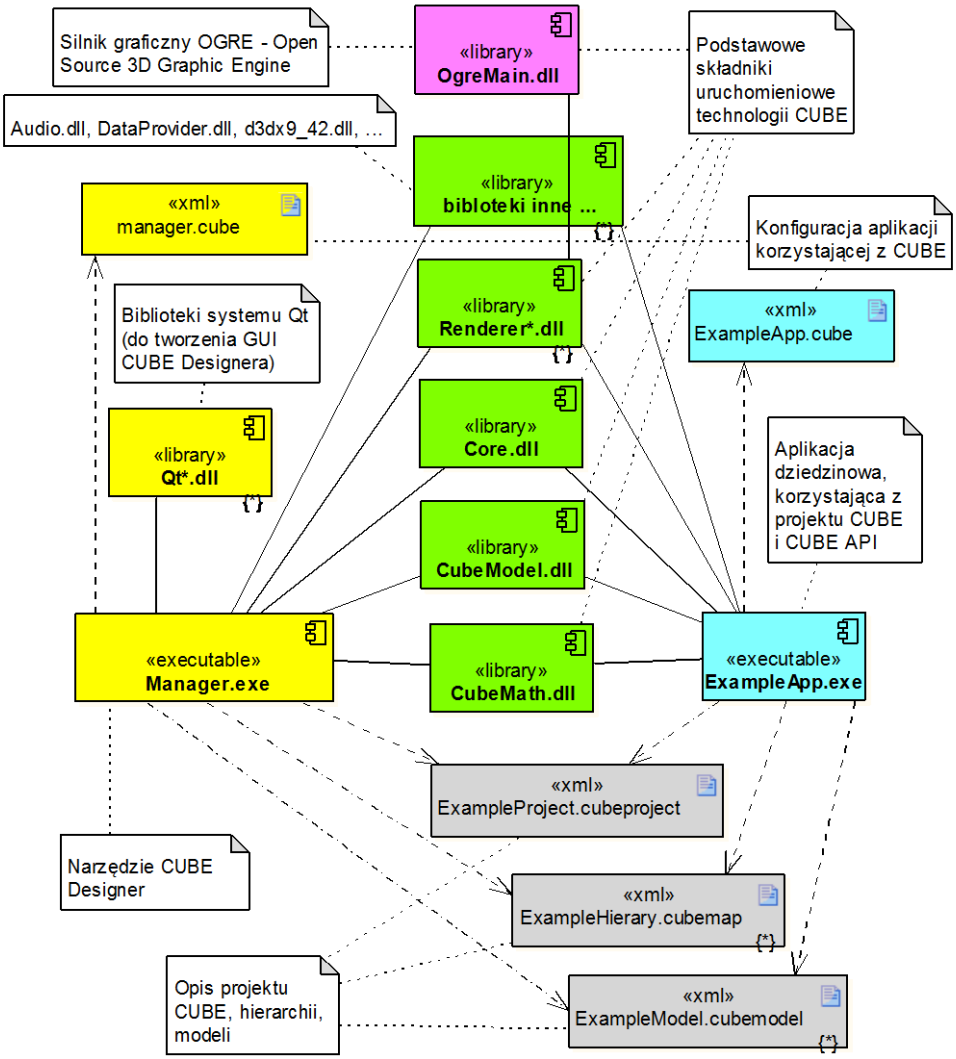
Program CUBE Designer powala na zdefiniowanie/modyfikowanie/zapisanie projektu CUBE, zawierającego hierarchie CUBE (zhierarchizowane zestawy scen, obiektów-modeli, okien renderowania, świateł, kamer, tzw. viewportów itd). Projekt CUBE zapisywany jest w pliku *.cubeproject*, a hierarchia w pliku *.cubemap* (oba elementy oznaczone kolorem szarym na rys. 1 – 3.). Czynność ta, oznaczona numerem 2, została pokazana na rys 3.

CUBE Designer powala na import obiektów-modeli, np. z pliku w formacie Collada (w wersji 1.4.1), tworząc odpowiednie opisy wczytywanych obiektów-modeli w plikach XML *.cubemodel* (czynność 4 z rys. 3).

CUBE Designer pozwala na interaktywne rozmieszczenie na scenie wczytanych modeli (czynność nr 6), czyli spełnia również rolę edytora sceny.

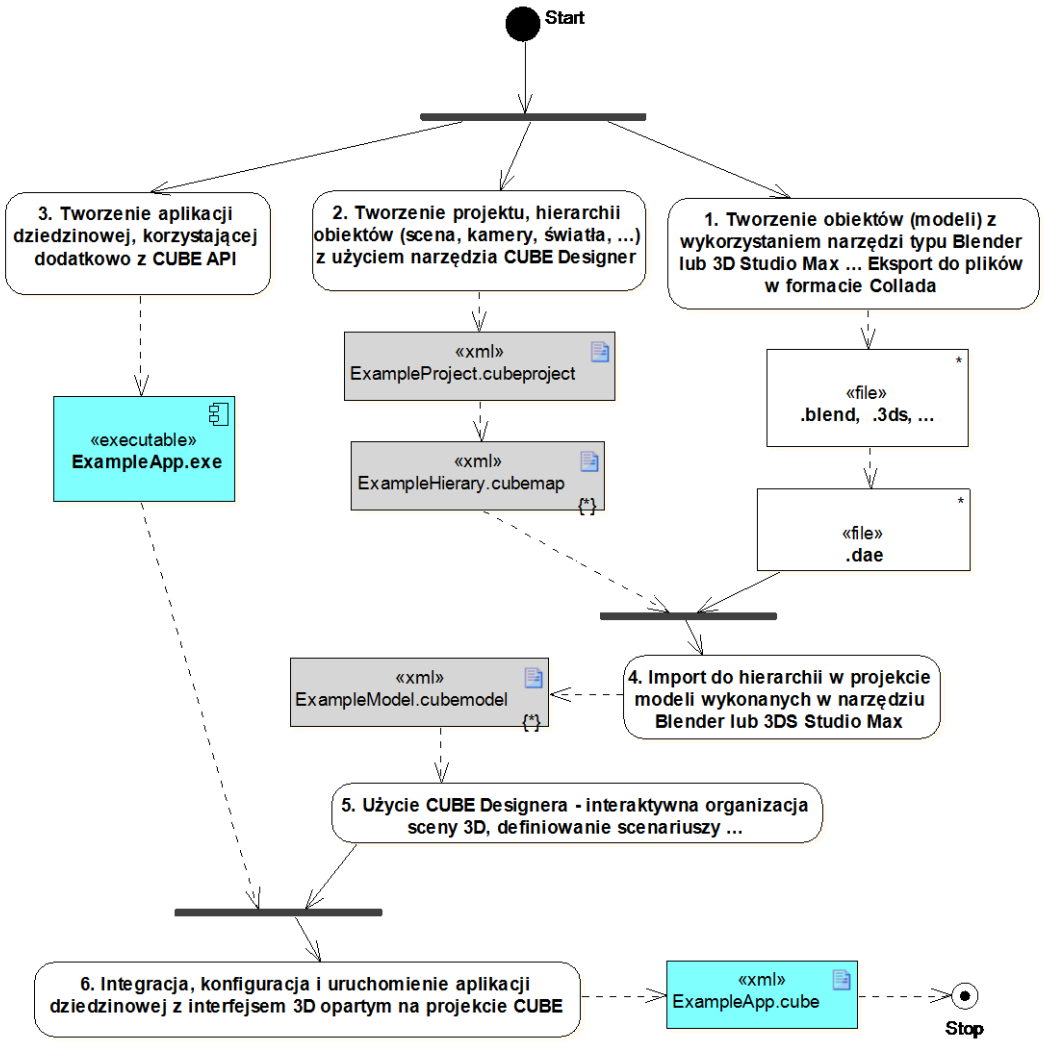
[2] Na portalu zeszytów Studia Informatica: http://znsi.aei.polsl.pl/, artykuł dostępny jest z rysunkami w kolorze.

Dzięki interfejsowi programowemu – CUBE API – możliwa jest integracja dowolnej aplikacji dziedzinowej (np. *ExampleApp.exe* na rys. 2 i 3) z systemem CUBE. W najprostszym przypadku, programista integrujący prostą aplikację z systemem CUBE powinien: zainicjować infrastrukturę CUBE, załadować projekt, załadować hierarchię obiektów (np. domyślną dla projektu), zainicjować wybrane okno aplikacji ze wskazaniem jako okno renderowania sceny. Ustawienia charakterystyczne dla aplikacji, ale związane z użyciem systemu CUBE, zapisywane są w pliku konfiguracyjnym *.cube* (*ExampleApp.cube* na rys. 2 i 3). Wszystkie te zadania zostały wyszczególnione na rys. 3, w ramach czynności nr 6.



Rys. 2. Komponenty CUBE i sposób ich wykorzystania
Fig. 2. Usage of CUBE components

Zadania, takie jak: tworzenie aplikacji dziedzinowej (czynność 3), utworzenie projektu CUBE (czynność 2) czy interaktywne tworzenie obiektów-modeli (czynność 1), mogą być realizowane niezależnie i zapewne będą wykonywane przez osoby o różnych kwalifikacjach.



Rys. 3. Proces wytwarzania aplikacji dziedzinowej, zintegrowanej z CUBE
Fig. 3. Process of CUBE-integrated domain application development

2.1. Ilustracja użycia narzędzia CUBE Designer do tworzenia prostej aplikacji wykorzystującej technologię CUBE

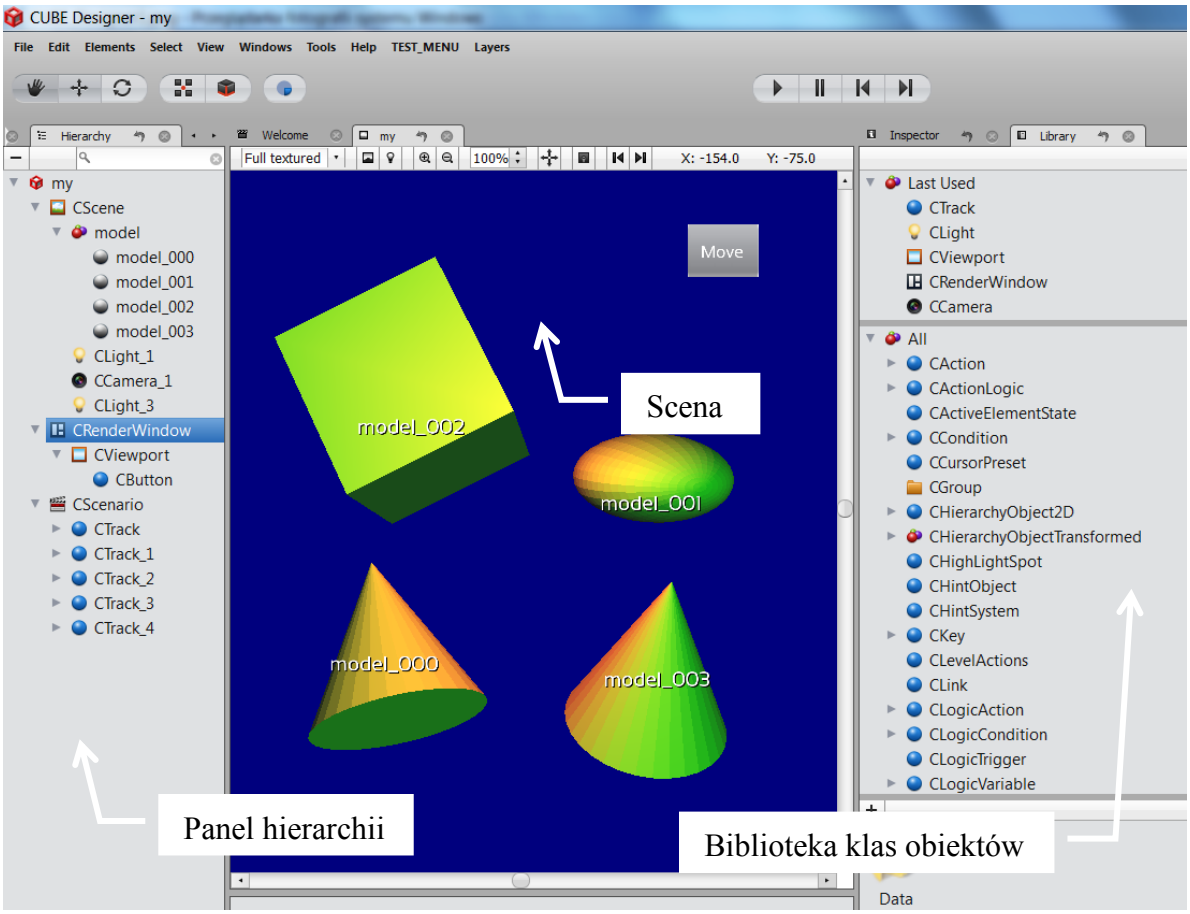
Rozdział stanowi ilustrację wybranych funkcjonalności CUBE Designera, których użycie umożliwi utworzenie przykładowego projektu CUBE. Projekt ten będzie mógł być wykorzystany w prostej, niezależnej aplikacji. Aplikacja będzie pozwalać na wizualizację sceny 3D i uruchomienie scenariusza – animacji dotyczącej przykładowego zestawu obiektów.

Przykładowy zestaw obiektów-modeli (stożek, sześcian, sfera) utworzono w programie Blender. W ramach projektu stworzono hierarchię (*my.cubemap*) zawierającą elementy:

- scena (*CScene*):
  - hierarchia obiektów-modeli (*model_000* i *model_004* – stożki, *model_002* – sześcian, *model_001* – sfera, różnie wyskalowana w każdym z trzech wymiarów),
  - źródła światła (*CLight_1* i *CLight_3*),

  - kamera (*CCamera_1*),
- okno renderowania (*CRenderWindow*):
  - viewport skojarzony przez odpowiednią właściwość z kamerą *CCamera_1*:
    - przycisk „dwuwymiarowy” z etykietą „Move” (*CButton*),
- scenariusz (*CScenario*) opisujący zaplanowane zachowania wybranych obiektów hierarchii:
  - pięć niezależnych ścieżek scenariusza (*CTrack*, *CTrack_1*,... , *CTrack_4*).

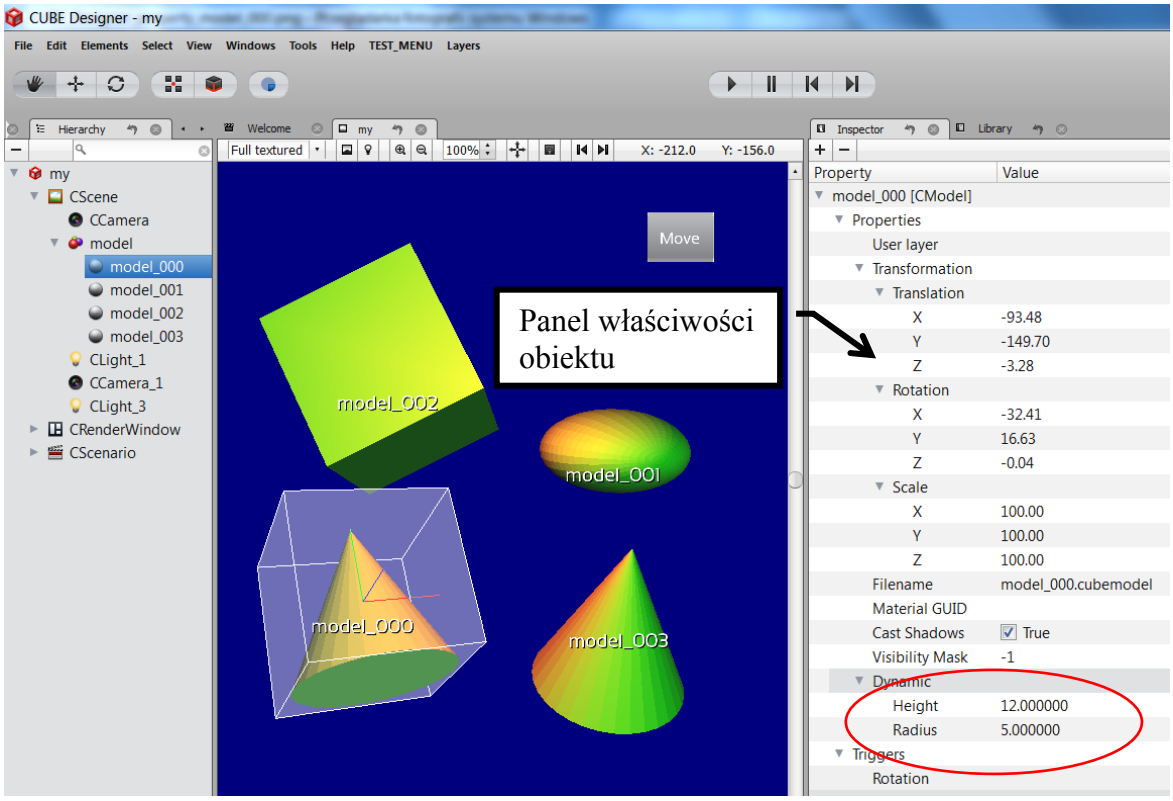
Ekran CUBE Designera w trakcie tworzenia hierarchii został pokazany na rys. 4. Tworzenie hierarchii obiektów odbywa się przez przeciągnięcie odpowiedniej nazwy z panelu biblioteki klas i upuszczeniu nad panelem hierarchii (lewy panel na rys. 4 – zakładka *Hierarchy*). Budowa hierarchii odbywa się na podstawie dostępnych w bibliotece klas obiektów (prawy dolny panel na rys. 4 – zakładka *Library*).



Rys. 4. Okno CUBE Designera i przykładowa scena 3D
Fig. 4. CUBE Designer window and the example of 3D scene

Każdy element hierarchii posiada swoje właściwości, pokazywane w panelu *Inspector* (prawy panel na rys. 5). Oprócz właściwości standardowych (charakterystycznych dla danego typu obiektu), obiekt może posiadać własności dziedzinowe, dynamiczne, definiowane przez użytkownika. Przykładowo, obiekt *model_000* (stożek) posiada dodatkowo zdefiniowane

właściwości: wysokość i promień podstawy (właściwości zaznaczone na rys. 5 czerwoną elipsą, w ramach kategorii *Dynamic*). Możliwość dynamicznych właściwości została wykorzystana w aplikacji prototypowej (opisanej w rozdziale 3.2), gdzie właściwości dynamiczne pozwalają na zakładanie filtru na „merytoryczne" obiekty sceny, tzn. związane bezpośrednio z dziedziną przedmiotową (kości szkieletu), pozwalając na ich programowe odróżnienie od elementów technicznych.



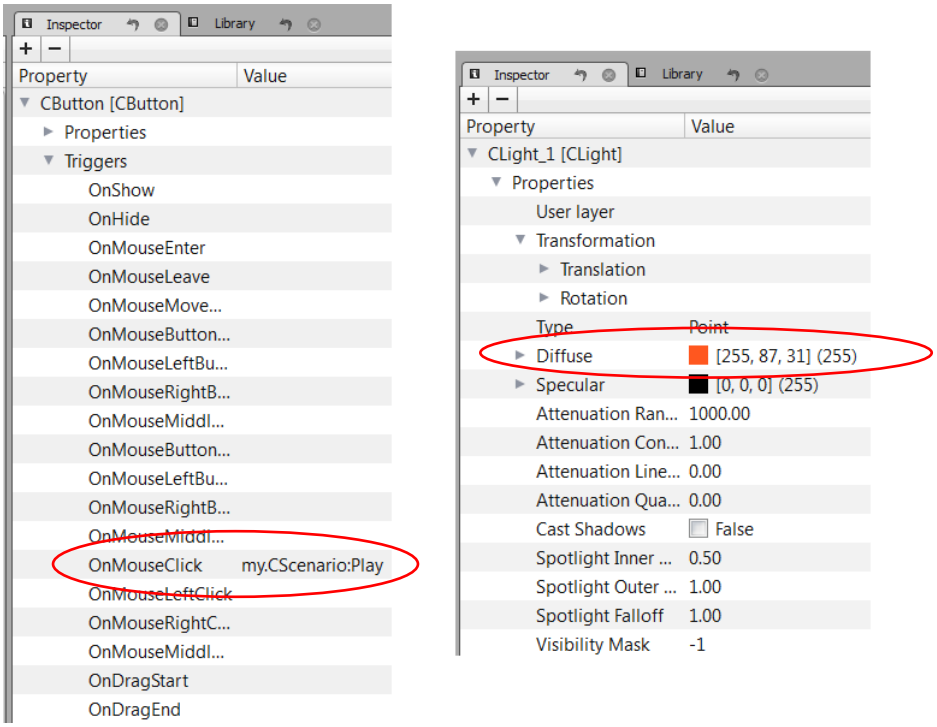
Rys. 5. Właściwości wybranego modelu – obiektu *model_000*
Fig. 5. Properties of the selected object – *model_000*

W scenie obsługiwanej przez przykładową prostą aplikację, umieszczony jest przycisk (*CButton*), którego naciśnięcie ma uruchomić scenariusz *CScenario*. Przypisanie wywołania scenariusza do zdarzenia kliknięcia myszy w ramach panelu właściwości *CButton* pokazano na rys. 6a).

W scenie umieszczone są dwa źródła światła, z czego właściwość koloru jednego z nich (*CLight_1*) będzie zmieniana w ramach realizacji scenariusza. Właściwości źródła światła *CLight_1*, w szczególności *Diffuse* (kolor), pokazano na rys. 6b.

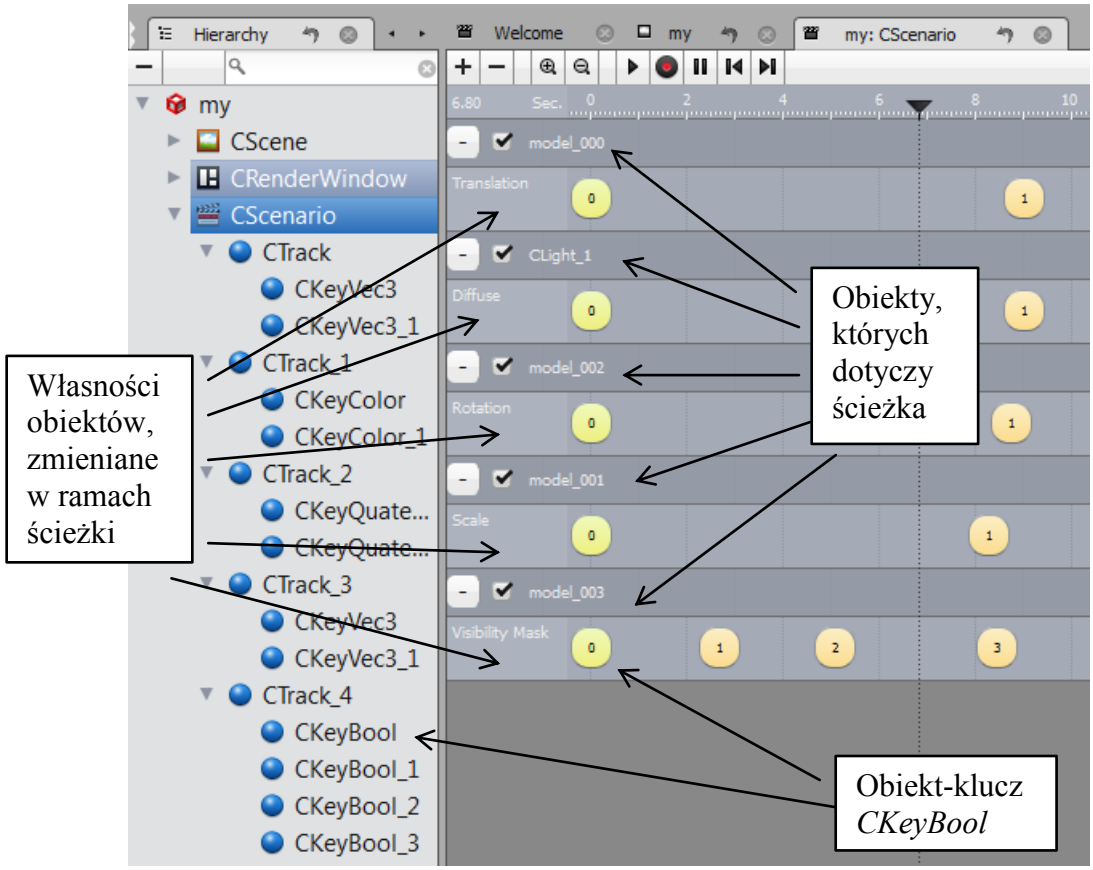
Scenariusze są mechanizmem CUBE, pozwalającym m.in. zdefiniować realizację akcji na scenie. Scenariusz (np. *CScenario* z rys. 7 – lewy panel *Hierarchy*) składa się ze ścieżek (np. *CTrack*, …, *CTrack_4*).

Ścieżki opisują zmianę wybranej właściwości pewnego obiektu w czasie (prawy panel *my:CScenario* na rys. 7).



a b

Rys. 6. Właściwości: a) przycisku *CButton*, b) źródła światła *CLight_1*
Fig. 6. Objects properties: a) *CButton*'s properties, b) *CLight_1*'s ones



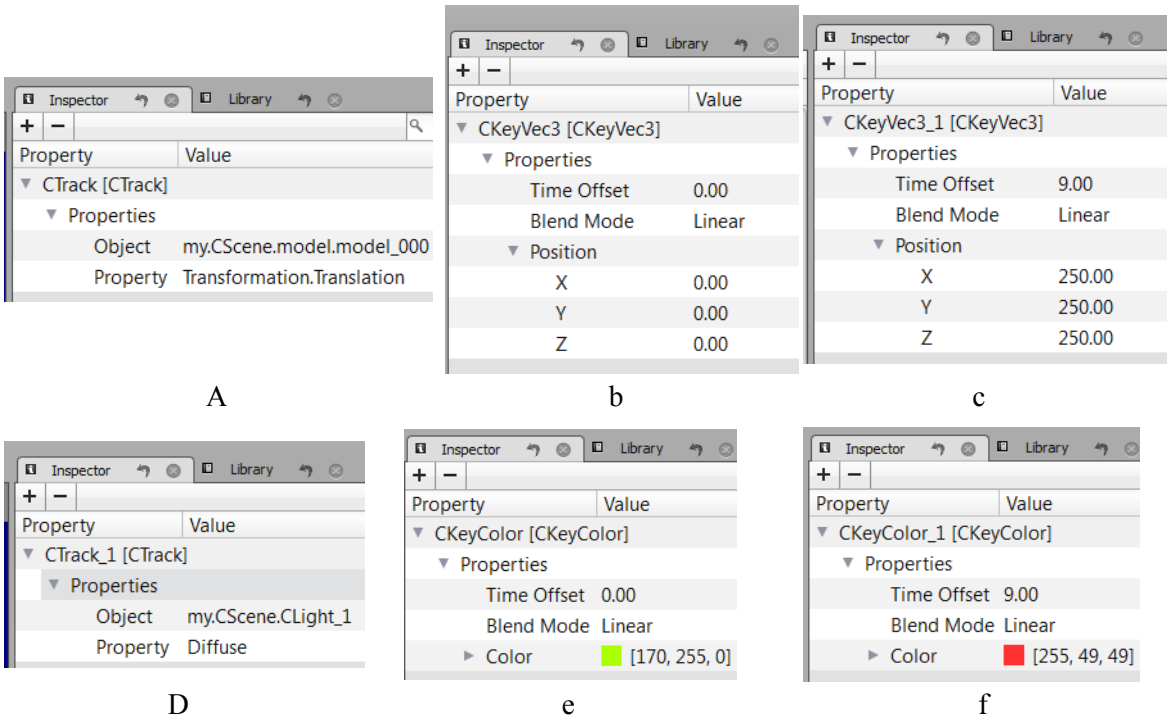
Rys. 7. Scenariusz z czterema ścieżkami
Fig. 7. The scenario with 5 tracks

We właściwościach każdej ścieżki należy określić obiekt sceny i wybrać jego właściwość podlegającą zmianie w ramach ścieżki, np. właściwość położenia (przesunięcia – *Translation*) obiektu *model_000* dla ścieżki *CTrack* (rys. 8a) czy właściwość koloru (*Diffuse*) źródła światła *CLight_1* dla ścieżki *CTrack_1* (rys. 8d).

W ramach ścieżek określa się obiekty konkretnych wartości, tzw. obiekty-klucze. Typy obiektów-kluczy zastosowane w przykładzie są następujące: *CkeyVec3* – wartość przesunięcia, *CKeyColor* – wartość koloru, *CKeyQuaternion* – określenie obrotu, *CKeyBool* – wartość boolowska (włącz/wyłącz).

Rozmieszczenie obiektów-kluczy na osi czasu określa momenty zmian śledzonej w ścieżce właściwości obiektu. Przykładowo, dla *CTrack* położenie zmienia się od wartości *CKeyVec3.Possition* (rys. 8b) do *CKeyVec3_1.Possition* (rys. 8c). Przykładowo, dla *CTrack_1* kolor zmienia się od wartości *CKeyColor.Color* (rys. 8e) do *CKeyColor_1.Color* (rys. 8f).

Sposób przejścia (zmiany) wartości właściwości śledzonej określa *BlendMode* w obiekcie-kluczu. W przykładzie, właściwości *BlendMode* (rys. 8b, c, e, f) są ustawione na *Linear*, co zapewnia płynną zmianę wartości śledzonych, w czasie pomiędzy momentami wyznaczonymi przez obiekty-klucze.



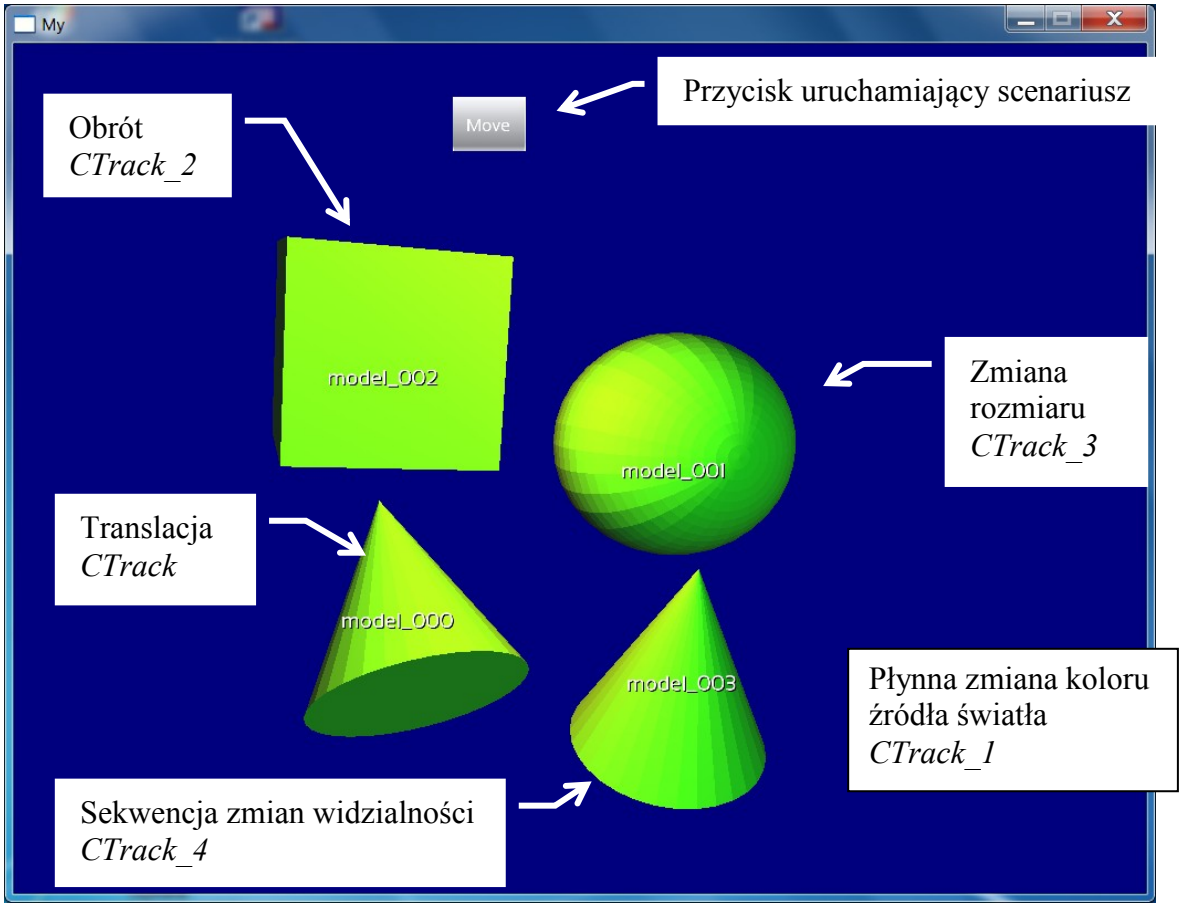
Rys. 8. Właściwości ścieżki *CTrack*: a) śledzona własność translacji obiektu *model_000*, b) wartość początkowa położenia, c) wartość końcowa. Właściwości ścieżki *CTrack_1*: d) śledzona własność koloru źródła światła *CLight_1*, e) wartość początkowa koloru, f) wartość końcowa

Fig. 8. Properties of *CTrack*: a) tracked *Translation* property of *model_000*, b) starting value of *Position*, c) final value of *Position*. Properties of *CTrack_1*: d) tracked *Diffuse* property of *CLight_1*, e) starting value of *Color*, f) final value of *Color*

Omówiony przykładowy projekt CUBE został wykorzystany w prostej, jednooknowej aplikacji. Inicjalny widok sceny 3D po uruchomieniu aplikacji pokazuje rys. 9. Naciśnięcie przycisku z etykietą „Move" spowoduje uruchomienie scenariusza, w ramach którego nastąpi obrót sześcianu, przesunięcie stożka, zmiana rozmiaru sfery, sekwencja czterech zmian widzialności drugiego stożka, zmiana koloru źródła oświetlenia (zielony na czerwony). Końcowy widok sceny 3D, po zakończeniu scenariusza, pokazuje rys. 10.

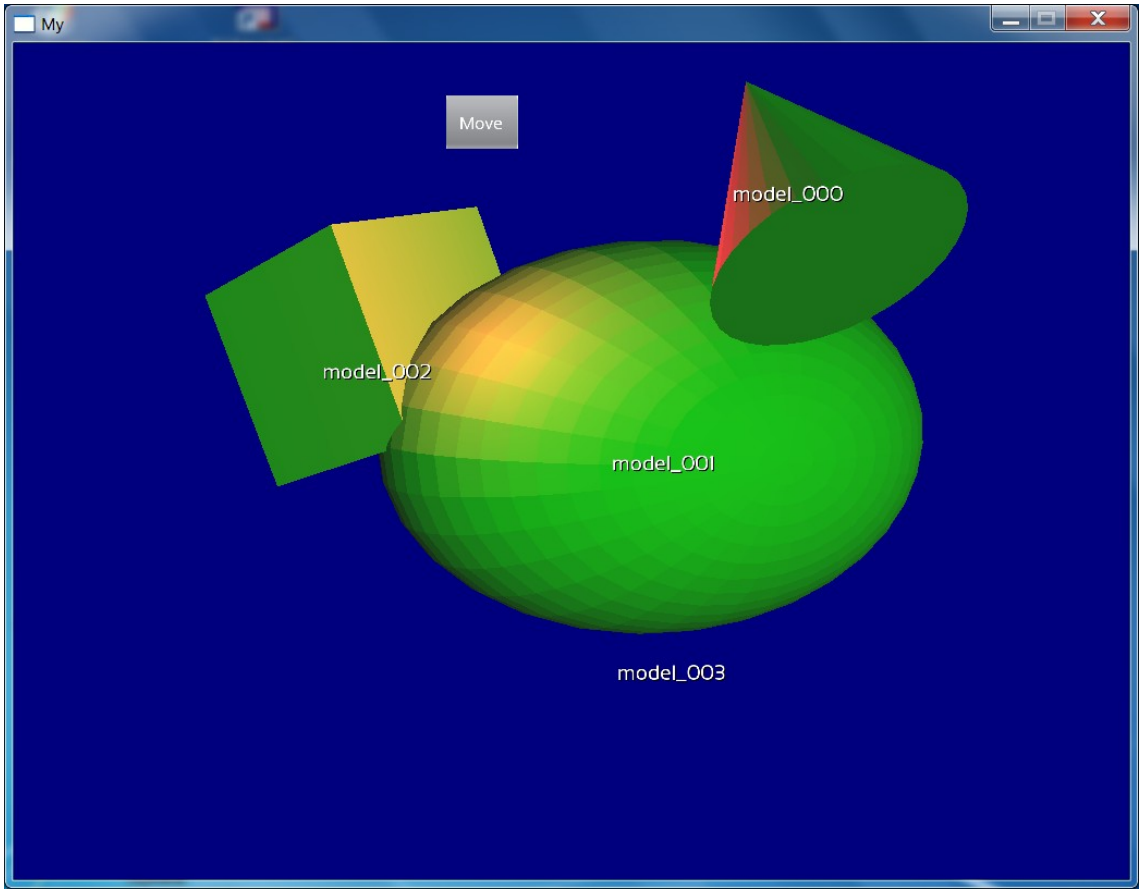
Przestawienie procesu utworzenia prostego programu do animacji zestawu obiektów miało m.in. za zadanie pokazanie możliwości CUBE Designera. Funkcjonalność programu CUBE Designer stanowi dobrą ilustrację możliwości wynikających z zastosowania CUBE API (program CUBE Desinger jest klientem usług dostarczanych przez CUBE API).

Prosty program (napisany w języku C++, skompilowany do wersji wykonywalnej Win32) jest przykładem, który praktycznie prawie nie wymaga programowania (oprócz czynności inicjujących). Akcje realizowane przez ten program są skonfigurowane już na etapie budowy projektu CUBE. Kod kolejnego, bardziej zaawansowanego przykładu aplikacji, będzie korzystał z CUBE API w celu programowej obsługi obiektów sceny.



Rys. 9. Ekran prostej przykładowej aplikacji – początkowy widok sceny 3D przed uruchomieniem scenariusza *CScenario*

Fig. 9. Simple example application screen – starting view of 3D scene before running the *CScenario*



Rys. 10. Ekran prostej przykładowej aplikacji – końcowy widok sceny 3D po wykonaniu scenariusza *CScenario*
Fig. 10. Simple example application screen – final view of 3D scene after running the *CScenario*

3. Zastosowanie technologii CUBE – aplikacja prototypowa – prezentacja danych dotyczących procesu terapeutycznego z użyciem modelu ciała ludzkiego

Prototypowa aplikacja to system wspomagający wstępną ocenę urazów kostnych pacjentów celem alokacji zasobów niezbędnych do wykonania planowanych procedur medycznych. Aplikacja ta (system wspomagania planowania działań diagnostyczno-leczniczych) spełnia pewien wybrany zakres wymagań funkcjonalnych, dotyczących systemu SOR (Szpitalny Oddział Ratunkowy).

Na etapie analizy wymagań określono kluczowe kryterium użyteczności systemu – przedstawienie dostępnych informacji w sposób umożliwiający użytkownikowi systemu (lekarzowi) szybkie podjęcie wymaganych decyzji. Zastosowanie standardowych metod prezentacji informacji, np. wierszy danych w układzie tabelarycznym, okazało się rozwiązaniem niewystarczającym. Przedstawienie informacji dotyczących urazów pacjentów z wykorzystaniem

trójwymiarowego modelu szkieletu umożliwia szybką, orientacyjną ocenę liczby oraz umiejscowienia urazów. Wykorzystanie modelu szkieletu jako interaktywnego menu, w którym po kliknięciu na wybrany element (kość), uzyskiwany jest dostęp do bardziej szczegółowych informacji o stwierdzonym urazie, zostało uznane za rozwiązanie intuicyjne i przyjazne użytkownikowi.

3.1. Fragment analizy wymagań dotyczących aplikacji dziedzinowej

Analiza wymagań dotyczących aplikacji prototypowej obejmuje: rozpoznanie i opisanie procesu biznesowego wspomaganego przez aplikację, zdefiniowanie modelu analitycznego, opisującego pojęcia będące przedmiotem procesu, oraz opracowanie modelu przypadków użycia tworzonego systemu.

3.1.1. Model procesu biznesowego

Proces biznesowy dotyczy wstępnej oceny stwierdzonych u pacjentów urazów kostnych celem zaplanowania działań diagnostyczno-leczniczych na oddziale szpitalnym o profilu ortopedycznym.

W przypadku wystąpienia określonych urazów u pacjenta, informacja o stwierdzonym rozpoznaniu wstępnym przekazywana jest do oddziału szpitalnego przygotowującego się do jego przyjęcia. Informacja może być przekazana w dowolny sposób, w szczególności w postaci elektronicznego komunikatu z zewnętrznego systemu, np. z systemu wspomagającego działania ratowników medycznych. Pozyskana informacja jest automatycznie lub ręcznie rejestrowana w systemie, po czym fakt jej dostępności jest komunikowany zainteresowanym osobom – lekarzom. Uprawnieni lekarze oceniają na postawie pozyskanych informacji stan pacjenta oraz rezerwują zasoby niezbędne do obsługi oczekiwanego przypadku.

Po zakończeniu realizacji procedur diagnostyczno-leczniczych zwalniają się zasoby w nich uczestniczące.

Kluczowym elementem procesu jest ocena stanu pacjenta. Konieczne jest podjęcie w krótkim czasie, na podstawie niepełnych informacji, decyzji o rezerwacji zasobów niezbędnych do realizacji wymaganych procedur medycznych.

Proces wstępnej oceny stanu pacjenta na podstawie przekazanych informacji o rozpoznaniu oraz alokacja zasobów niezbędnych do przeprowadzenia planowanych procedur diagnostyczno-leczniczych, mogą być wspomagane przez dedykowany system informatyczny do wizualizacji stanu pacjenta.

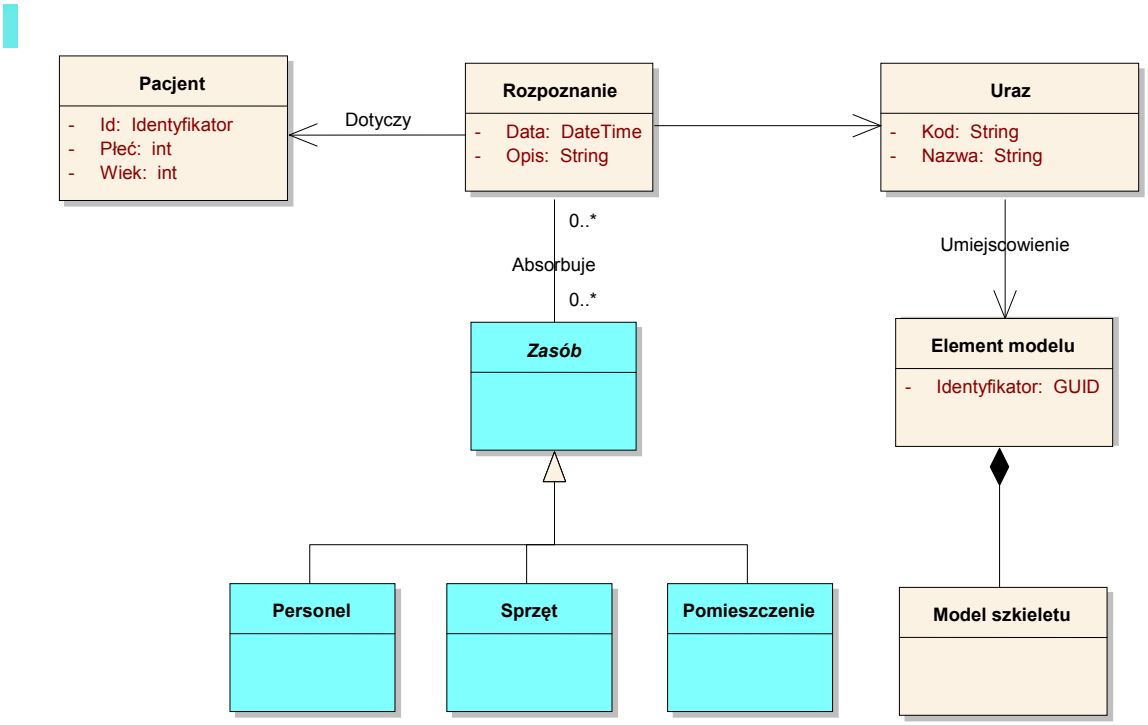
3.1.2. Model analityczny

Model analityczny opisuje m.in. podstawowe pojęcia występujące w procesie wspomaganym przez system.

Wyróżniono następujące pojęcia w ramach procesu oceny i planowania obsługi pacjentów w przypadku wystąpienia urazów kostnych:

1. Pacjent – osoba poszkodowana, u której stwierdzono określony uraz ortopedyczny, dla pacjenta określa się płeć oraz orientacyjny wiek.
2. Uraz – określony przez kod Międzynarodowej Klasyfikacji Chorób i Problemów Zdrowotnych ICD-10 [7].
3. Model szkieletu – trójwymiarowy model szkieletu człowieka, umożliwiający prezentację umiejscowienia urazów.
4. Element modelu – kość lub część kości, której może dotyczyć określony uraz.
5. Rozpoznanie – informacje o pierwszej ocenie urazów pacjenta, zawierające datę uzyskania informacji, stwierdzony uraz oraz dodatkowe informacje przekazane przez osobę stwierdzającą uraz pacjenta, użyteczne w procesie planowania opieki nad pacjentem.
6. Zasób – osoba, sprzęt lub pomieszczenie niezbędne do wykonania planowanych procedur medycznych (zaznaczone kolorem niebieskim na rys. 11).

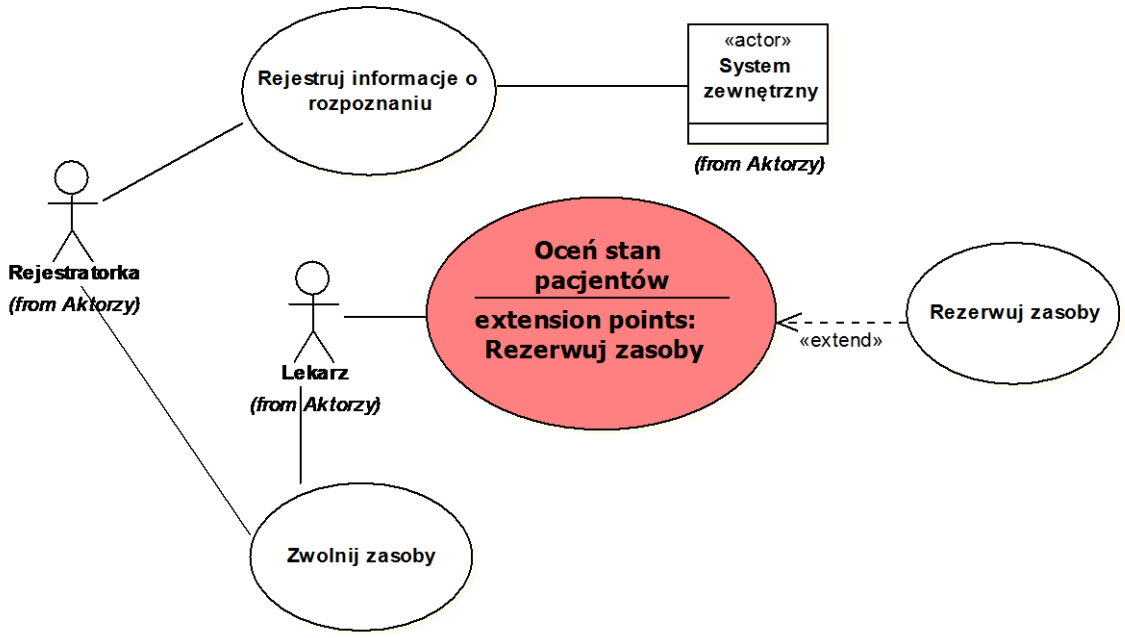
Zależności pomiędzy podstawowymi pojęciami przedstawia diagram klas na rys. 11.



Rys. 11. Analityczny model danych
Fig. 11. Analytical data model

3.1.3. Model przypadków użycia systemu

Model przypadków użycia określa aktorów korzystających z systemu oraz sposób użycia systemu, opisany w postaci przypadków użycia (rys. 13) i scenariuszy.



Rys. 12. Model przypadków użycia
Fig. 12. Use case model

Aktorzy:

1. Lekarz – lekarz oceniający stan pacjentów i podejmujący decyzję o rezerwacji zasobów niezbędnych do wykonania planowanych procedur medycznych.
2. Rejestratorka – osoba odpowiedzialna za rejestrację informacji o stwierdzonych urazach pacjentów, przekazywanych telefonicznie lub w formie papierowej dokumentacji dostarczonej na oddział.
3. System zewnętrzny – zewnętrzny system informatyczny, generujący informacje o stwierdzonych urazach w postaci komunikatów w formie elektronicznej.

Przypadki użycia:

1. Rejestruj informacje o rozpoznaniu

Rejestracja informacji o urazach stwierdzonych u pacjentów.

2. Rezerwuj zasoby

Przypadek użycia opisuje sposób rezerwacji zasobów niezbędnych do wykonania planowanych działań diagnostyczno-terapeutycznych.

3. Zwolnij zasoby

Zapis w systemie informacji o zakończeniu określonych działań diagnostyczno-terapeutycznych, co wiąże się ze zwolnieniem zasobów zaangażowanych w ich wykonanie.

4. Oceń stan pacjentów

Przypadek użycia opisuje sposób, w jaki system informatyczny wspomaga ocenę stanu zdrowia pacjentów. Do realizacji przypadku użycia zastosowano prezentację umiejscowienia urazów na trójwymiarowym modelu szkieletu człowieka.

- Scenariusz podstawowy: prezentacja pacjentów i ich urazów.

  Realizacja scenariusza rozpoczyna się, gdy użytkownik uruchamia opcję przeglądu urazów pacjentów:

  1. System prezentuje listę pacjentów, u których rozpoznano wystąpienie określonych urazów.
     Na liście prezentowane są: identyfikator pacjenta, płeć, wiek pacjenta.
  2. Użytkownik wskazuje wybranego pacjenta z listy.
  3. System prezentuje trójwymiarowy model szkieletu pacjenta, na którym oznacza elementy związane z dotychczas zarejestrowanymi rozpoznaniami wskazanego pacjenta.
  4. Użytkownik wskazuje element szkieletu związany z wybranym urazem.
  5. System prezentuje szczegółowe informacje na temat rozpoznania: datę rozpoznania, kod i nazwę choroby, opis rozpoznania.

  <<extension point>> Rezerwuj zasoby

- Scenariusz alternatywny: prezentacja rozpoznań wszystkich pacjentów.

  Jeżeli w pkt. 2 scenariusza podstawowego użytkownik chce zobaczyć jednocześnie urazy wszystkich aktualnie obsługiwanych pacjentów:

  2a.1. Użytkownik wybiera opcję „zaznacz wszystko” (tzn. pokaż rozpoznania wszystkich pacjentów).

  2a.2. System prezentuje model szkieletu, na którym wyróżnia elementy związane z zarejestrowanymi w systemie, aktywnymi rozpoznaniami.

  Następuje przejście do pkt. 4 scenariusza podstawowego.

- Scenariusz alternatywny: zakończenie oceny stanu pacjentów.

  Jeżeli w pkt. 4 scenariusza podstawowego użytkownik zakończył wstępną ocenę stanu pacjentów:

  4a.1. Użytkownik zamyka okno przeglądu stanu pacjentów.

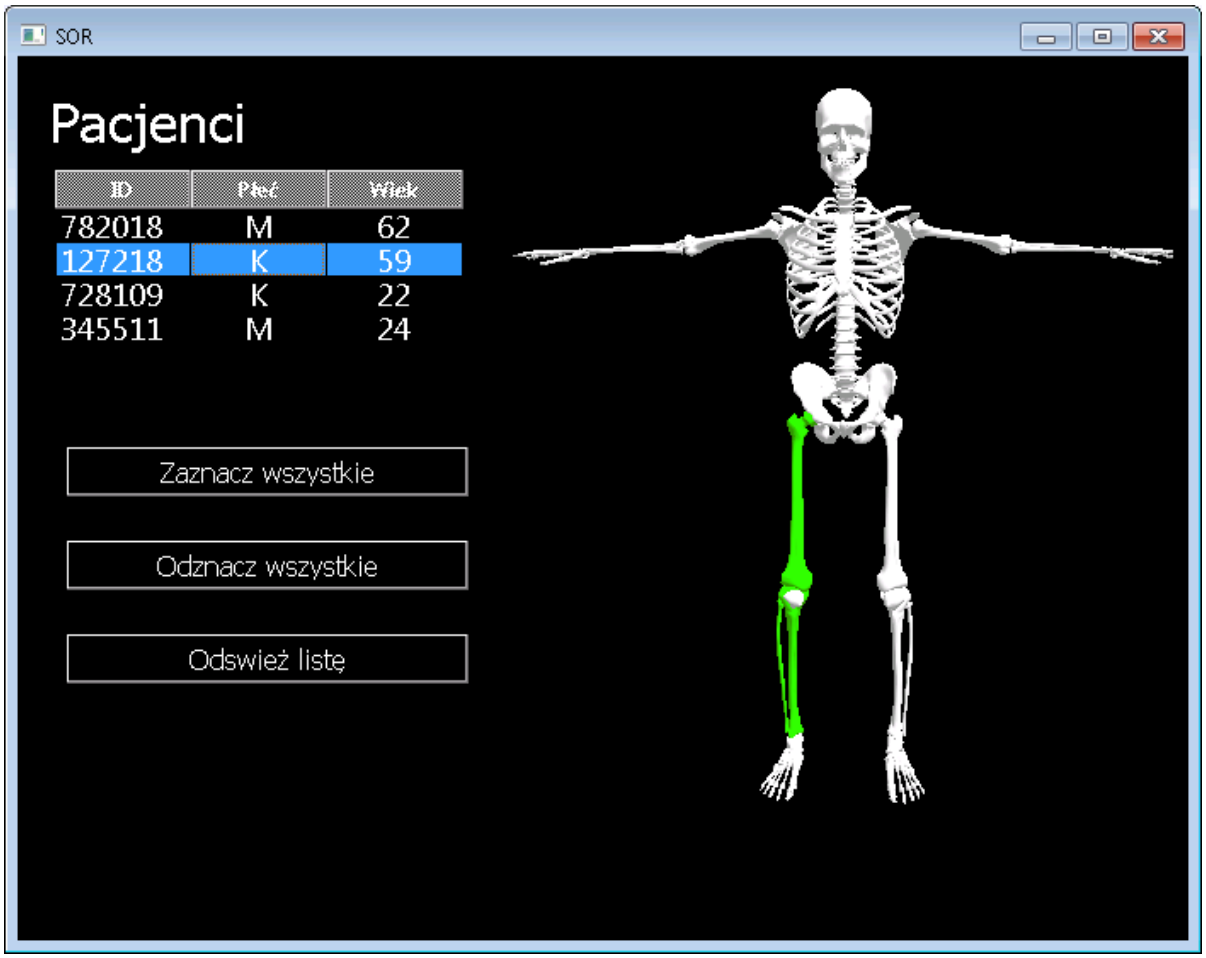
  Następuje zakończenie realizacji przypadku użycia.

- Warunki i ograniczenia:
  Warunki początkowe:
  – użytkownik zalogowany do systemu,
  – użytkownik posiada uprawnienie do przeglądu pacjentów.

3.2. Realizacja aplikacji prototypowej, wykorzystującej technologię CUBE

Prototypowa aplikacja została utworzona dla środowiska uruchomieniowego Win32 i napisana w języku C++, dzięki czemu możliwe było skorzystanie wprost z komponentów CUBE, tzn. bez konieczności pisania warstwy pośredniej (ang. *wrapper*). Do stworzenia graficznego interfejsu użytkownika wykorzystano bibliotekę QT [8] w wersji 4.6.2.

W niniejszym rozdziale omówiono działanie aplikacji prototypowej w zakresie realizacji przypadku użycia „Oceń stan pacjenta", dotyczącego wizualizacji urazów. Pokazano fragmenty kodu wykorzystującego CUBE API.



Rys. 13. Główne okno aplikacji z zaznaczonymi urazami kości wybranego pacjenta (kolor zielony)
Fig. 13. Main application window with selected patient bone injuries (green color)

Po uruchomieniu aplikacji główne okno ma inicjalną postać, pokazaną na rys. 13. Główne okno omawianej aplikacji składa się z trzech obszarów (jak na rys. 14). Po lewej stronie znajduje się lista pacjentów przyjętych na SOR. Informacje w niej zawarte pochodzą z zewnętrznego źródła – w szczególności z systemu wspomagającego ratowników medycznych. Każdy pacjent określony jest jednoznacznie kodem identyfikującym (ID). System prezentuje także płeć poszkodowanego oraz jego wiek. W centralnej części ekranu znajduje się trójwy-

miarowy model szkieletu ciała ludzkiego[3]. Za pomocą myszy i klawiatury można go przesuwać, obracać, powiększać itp. Po prawej stronie pojawia się panel ze szczegółowymi informacjami dotyczącymi wybranej kości (szczegóły wstępnego rozpoznania) – rys. 14.

Wiązaniem danych z kontrolkami, obsługą sygnałów i zdarzeń w oknie aplikacji zajmuje się specjalizowana klasa *SorMainWindow*.

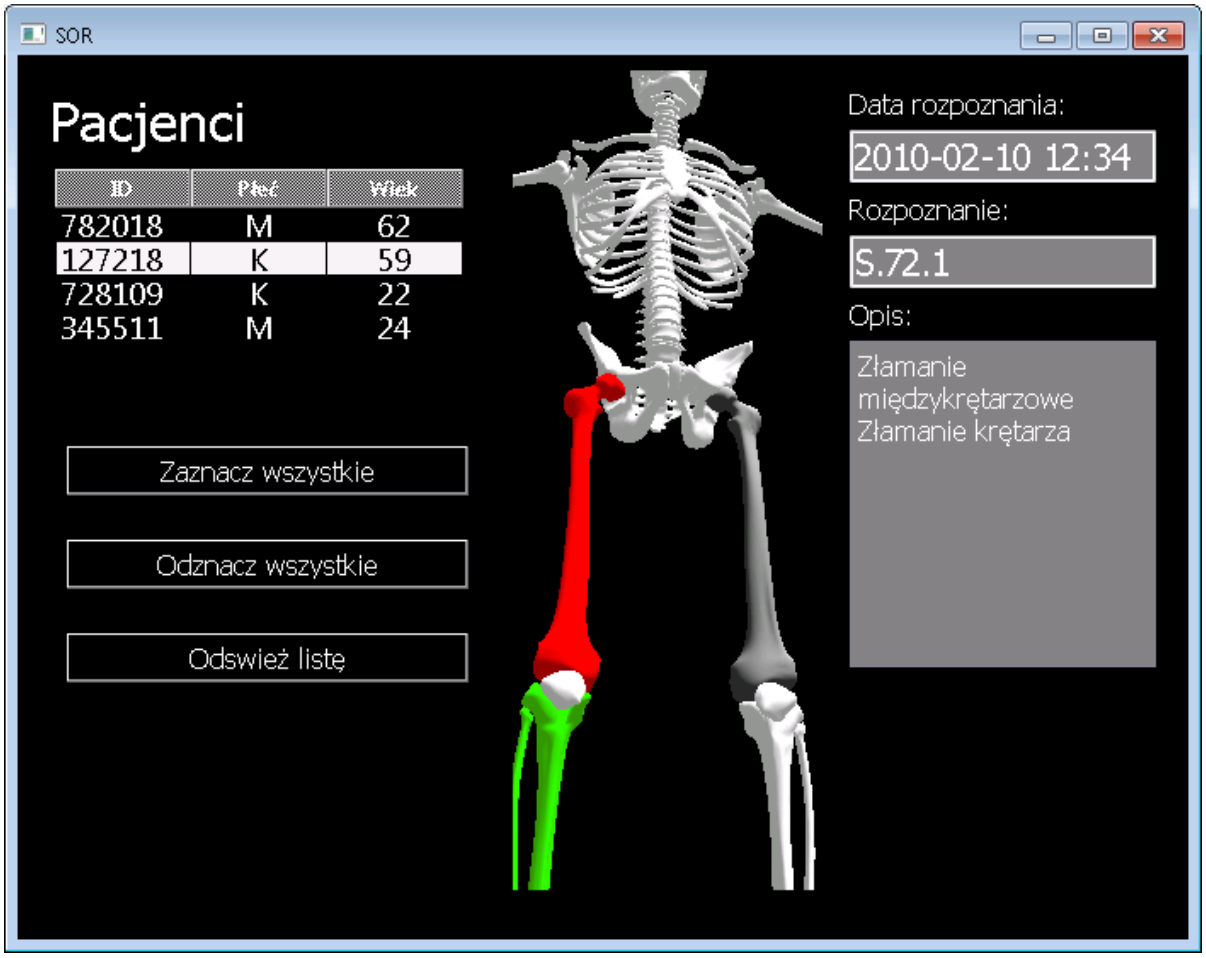
Zmiana zaznaczenia w liście pacjentów powoduje wysłanie sygnału (ang. *signal*), który znajduje swoje ujście (ang. *slot*) w metodzie *PatientSelected()* klasy *SorMainWindow*. W metodzie następuje odczytanie szczegółowych danych wstępnego rozpoznania i naniesienie tych informacji na model szkieletu (zaznaczenie uszkodzonych kości kolorem zielonym). Pobranie z hierarchii odpowiednich instancji obiektów (reprezentujących kości) odbywa się za pomocą metody *FindObjects()*. Następnie w każdym obiekcie ustawiana jest właściwość określająca „materiał" użyty do wyrysowania danej kości.

Fragment kodu źródłowego opisanej metody zamieszczono poniżej:

```
// pobranie hierarchii z projektu
IHierarchyPtr hierarchy = m_Project->GetCurrentHierarchy();
// utworzenie filtru wyszukującego obiekty klasy CModel
ISchemaPtr schema = m_CubeCore->CreateSchemaFilter("","CModel");
//pobranie listy znalezionych obiektów
IHierarchyObjectCollectionPtr collection = hierarchy->FindObjects(schema);
// iterowanie po liście
for(INT i=0; i< (INT)collection->GetObjectCount(); i++) {
    // pobranie konkretnego obiektu z kolekcji
    model = dynamic_pointer_cast<IModel>(collection->GetObjectAt(i));
    // sprawdzenie czy kość występuje w opisie urazu pacjenta
    // (na podstawie bazy danych systemu dziedzinowego)
    if(isInjuredBone(model))
    {
        // określenie materiału do wyrysowania - kolor zielony
        model->GetSchema()->GetProperty("Material GUID","")
            ->Set(m_Material_Broken_Bone->GetGUID());
    }
}
```

Jeżeli osoba obsługująca aplikację potrzebuje dodatkowych informacji, wskazuje myszą wybraną kość. Aplikacja sprawdza, czy w miejscu, w którym wciśnięto przycisk myszki, znajduje się jakiś obiekt. Odbywa się to za pomocą wirtualnego promienia, wysłanego z punktu, w którym umieszczona jest kamera. Aplikacja sprawdza, które obiekty z nim kolidują i wybiera ten najbliższy.

[3] Model szkieletu został wykonany przez zewnętrzną firmę i zakupiony przez 2KMM Sp. z o.o.



Rys. 14. Okno aplikacji z zaznaczoną kością (kolor czerwony) i szczegółowymi informacjami o wstępnym rozpoznaniu
Fig. 14. Application window with marked bone (red color) and details information about diagnosis

Poniżej przedstawiono fragment kodu źródłowego metody realizującej ww. zadanie:

```
// utworzenie promienia przechodzącego przez punkt kamery i miejsce kliknięcia
ray cameraRay =
    m_Camera.m_Ptr.lock()->CameraToViewportRay(
        m_Viewport.lock(),vec2(mouse.x(),mouse.y()),FALSE);
// iteracja po elementach modelu 3D
for(UINT i=0; i< m_Application->GetModelsNb(); i++)
{
    // pobranie obiektu
    model3D = m_Application->GetModel(i).lock();
    // sprawdzenie "kolizji" obiektu i promienia wirtualnego
    if(model3D->Collide(cameraRay,tmp_distance))
    {
        //...
    }
}
```

Jeśli wynikiem tej operacji jest uszkodzona kość, pobierane są dla niej szczegółowe informacje o rozpoznaniu i wyświetlane są w panelu po prawej stronie. Kość zostaje zaznaczona kolorem czerwonym (rys. 15) – zostaje użyty odpowiedni materiał, tak jak w poprzedniej metodzie.

4. Podsumowanie

Celem prac opisanych w niniejszym artykule była realizacja aplikacji prototypowej, pozwalającej na ocenę przydatności technologii CUBE. Ze względu na zainteresowania autorów medycznymi systemami informatycznymi, utworzona aplikacja dotyczyła prezentacji rozpoznań na trójwymiarowym modelu ciała ludzkiego.

Realizacja aplikacji potwierdziła oczekiwania co do łatwej obsługi, związanej z przygotowaniem projektu CUBE (użycie narzędzia CUBE Designer) oraz nieskomplikowanego rozszerzenia aplikacji dziedzinowej, jako klienta podsystemu CUBE (oprogramowanie wywołań CUBE API). Technologia CUBE (podejście, narzędzie, komponenty) w założonym zakresie została zweryfikowana pozytywnie. Stwierdzono pewne usterki w działaniu pierwszej dostarczonej wersji CUBE Designera. Zwrócono uwagę na pewne niedostatki aktualnej dokumentacji w zakresie CUBE Designera i CUBE API, a najprawdopodobniej ten aspekt będzie jednym z istotnych czynników decydujących o skali powodzenia projektu CUBE, mierzonej stopniem łatwości integracji z dziedzinowymi aplikacjami.

Rozważana jest realizacja innych aplikacji pilotowych, dotyczących następujących zastosowań:

- systemy edukacyjne (np. nauka podstaw stereometrii),
- aplikacje instruktarzowe (interaktywne instrukcje obsługi złożonych urządzeń, np. układanie kostki Rubika, demontaż karabinu maszynowego, wkładanie karty SIM i bateryjek do urządzenia mobilnego),
- proste gry (2- i 3-wymiarowe),
- systemy z prostą wizualizacją elementów architektury (np. wizualizacja położenia pacjenta w budynku szpitala czy droga dotarcia do pacjenta dla odwiedzających, począwszy od wejścia do budynku).

Zmiany dotyczące omawianej technologii zapewne będą dotyczyć uporządkowania, rozbudowy i ułatwień w zakresie udostępnianego CUBE API, tak aby docelowi klienci – programiści firm trzecich, tworzący oprogramowanie aplikacyjne dla biznesu – odnajdywali technologię CUBE jako bardzo łatwą w użyciu i niezawodną.

W zakresie technicznym, rozwój systemu zakłada stworzenie implementacji komponentów CUBE w technologii przeglądarkowej (obecnie możliwa jest jedynie budowa tzw. aplikacji desktopowych). Według informacji pochodzących od producenta (firma 2KMM Sp. z o.o.) , obecnie tworzona jest implementacja wykorzystująca metody akceleracji, wynikające z użycia technologii CUDA.

Zagadnienie tworzenia aplikacji z 3D GUI jest jak najbardziej aktualnym problemem technologicznym. Światowe firmy software'owe starają się proponować rozwiązania w tym zakresie. Przykładem może być działanie firmy Google, która ostatnio zaczęła udostępniać

technologię WebGL [5], pozwalającą na tworzenie aplikacji Web z interfejsem 3D, w niektórych typach przeglądarek (z odpowiednią wtyczką programową). Wśród przykładowych zastosowań WebGL znalazła się również aplikacja działająca na podstawie modelu ciała ludzkiego [6].

Zapewne podstawowymi cechami technologii CUBE, na których jej twórcy powinni budować przewagę konkurencyjną, jest prostota interfejsu CUBE API i łatwość edycji sceny oraz tworzenia projektu CUBE (za pomocą narzędzia CUBE Designer), czyli w konsekwencji minimalizacja prac programistycznych, wykonywanych przez firmy trzecie, tworzące oprogramowanie aplikacyjne.

BIBLIOGRAFIA

1. 2KMM | EU Programs | Innovative Economy (2010), http://www.2kmm.pl/EUPrograms/ /InnovativeEconomy.aspx.
2. OGRE – Open Source 3D Graphics Engine (2011), http://www.ogre3d.org.
3. blender.org – Home (2011), http://www.blender.org.
4. COLLADA – Digital Asset and FX Exchange Schema (2011), https://collada.org/ mediawiki/index.php/COLLADA_-_Digital_Asset_and_FX_Exchange_Schema.
5. Planet WebGL (2011), http://planet-webgl.org.
6. Google Body – Google Labs (2011), http://bodybrowser.googlelabs.com.
7. WHO – International Classification of Diseases (ICD) (2011), http://www.who.int/ classifications/icd/en.
8. QT – Cross Platform Application and UI Framework, http://qt.nokia.com/products.

Recenzenci: Dr Ewa Romuk
Prof. dr hab. inż. Konrad Wojciechowski

Wpłynęło do Redakcji 31 stycznia 2011 r.

Abstract

Modern information systems often require advanced GUI corresponding to handled domains. Some domains impose usage of 3D GUI of developed application. The problem is that programmers are specialists in business domain but commonly not in computer graphics. The CUBE technology form 2KMM Company facilitate the process of making 3D GUI

extension. The handy tool – CUBE Designer and the simple programming interface – CUBE API allow to incorporate 3D GUI into a standard application.

This paper introduces into the CUBE approach. The paper describes main CUBE components (libraries and the tool for designing a 3D scene). It also shows in details the method of preparing the simple animation application based on the CUBE project with scenarios. This allow to emphasise the functionality of CUBE Desinger.

The verification of CUBE technology usability was the main goal of the work described in the article. The verification was made by successful preparing (by the paper authors) the prototype application with man-body-model-based 3D GUI. The prototype application affects medical domain.

Adresy

Dariusz R. AUGUSTYN: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, draugustyn@polsl.pl.

Łukasz WARCHAŁ: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, lukasz.warchal@polsl.pl.

Piotr ŻELAŹNICKI: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, pzelaznicki@gmail.com.