Rozdział 3.9
Zarządzanie złożonością scenerii na bazie reguł logiki rozmytej

Gabriyel Wong, Nanyang Technological University
gabriyel@gmail.com

Jianliang Wang, Nanyang Technological University

Jest rzeczą zrozumiałą, że projektując grę komputerową, należy mieć na uwadze ograniczoną moc obliczeniową dostępnych procesorów potencjalnej platformy docelowej i wynikające stąd ograniczenia w zakresie złożoności renderowania grafiki. Autorzy grafiki trójwymiarowej często korzystają z edytorów sprzężonych wprost z silnikami graficznymi, zyskując w ten sposób możliwość sprawdzenia, czy złożoność ta daje się pogodzić (na konkretnym sprzęcie) z żądaną częstotliwością wyświetlania ramek. Podejście to, które stało się już normą w stosunku do gier rozgrywanych przez pojedynczego gracza — głównie dzięki dużej przewidywalności poczynań tego ostatniego — staje się jednak prawdziwym wyzwaniem w odniesieniu do środowisk dynamicznych, w scenerii współdzielonej przez wielu graczy. W niniejszym rozdziale opiszemy wychodzące naprzeciw temu wyzwaniu podejście oparte na logice rozmytej, umożliwiające pogodzenie efektywności wyświetlania z jakością generowanego obrazu.

Koncepcja

W obliczu ustalonej mocy dostępnych zasobów sprzętowych podstawowym celem grafiki czasu rzeczywistego jest maksymalizacja ostrości (jakości) obrazu odtwarzanego z określoną częstotliwością ramek (wydajność). Mimo iż powszechnie znane rozwiązania w tym zakresie, w rodzaju kontroli poziomu szczegółowości (ang. *Level of Details — LOD*) czy redukcji zbioru widocznych obiektów, przyczyniają się wydatnie do redukcji geometrycznej złożoności obrazu, to niezaprzeczalnie stopień tej redukcji zależny jest od konkretnej postaci scenerii i rozwiązania te nie najlepiej nadają się do ogólnej kontroli obciążenia silnika renderującego.

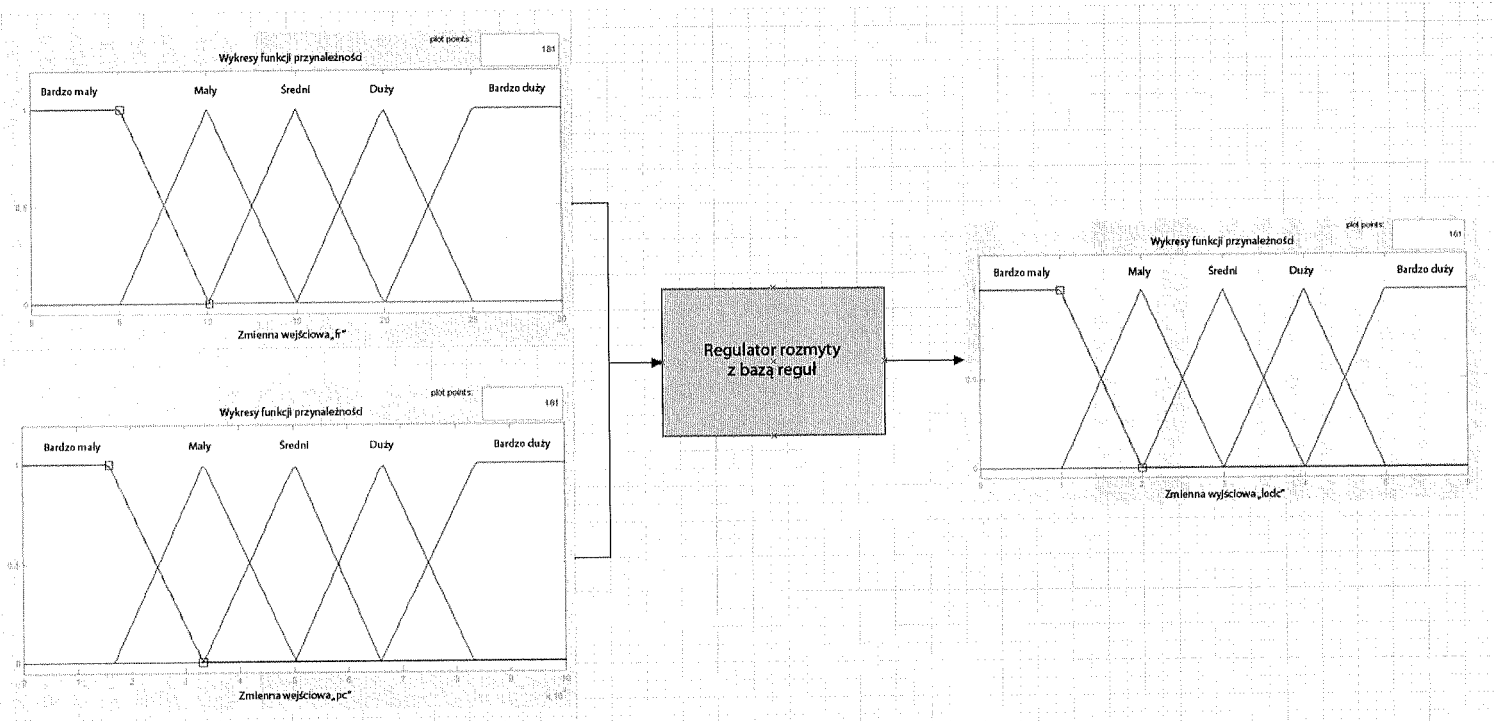
W niniejszym rozdziale nie proponujemy niczego w rodzaju nowego algo... unikalnej heurystyki redukcji geometrycznej, prezentujemy w zamian nowatorskie po... do zarządzania poziomem szczegółowości sceny (LOD) bazujące na zasadach lo... mytej (ang. *fuzzy logic*). Utożsamiając proces renderowania grafiki z procesem pr... cyjnym, można ustanowić coś na kształt globalnego kontrolera (regulatora) do... przepływ mocy „produkcyjnej" (obliczeniowej) do poszczególnych elementów ... cesu. W połączeniu z mechanizmem sprzężenia zwrotnego kontroler ten, wyk... wbudowane reguły wywodzące się z wiedzy projektanta, zdolny jest regulow... żenie procesorów drogą rozstrzygania wspomnianego kompromisu „jakość-wy...

Regulacja rozmyta

Zaadaptowanie reguł logiki rozmytej [FIDE1] do potrzeb kontrolowania renderingu ... ze sobą rozliczne korzyści. Przede wszystkim, w przeciwieństwie do konwencjo... metod sterowania, logika rozmyta dostarcza metodologii szybkiej i wygodnej. Sam pr... renderowania jest wystarczająco złożony, by jego ścisłe modelowanie uznać za ... wysoce niebanalne; techniki rozmyte dostarczają alternatywy dla modelowania ... matycznego w postaci reguł wejścia-wyjścia budowanych na bazie wiedzy projek... Implementacja tych reguł upraszcza się do postaci instrukcji warunkowych IF-... organizujących relacje między danymi wejściowymi a wyjściowymi. „Lingwistyczna" ... owych reguł czyni je bardziej przystępnymi zarówno dla projektantów, jak i użytko... bowiem wywodzą się one z opisów formułowanych w języku naturalnym, a nie z zw... reguł matematycznych. Ponadto logika rozmyta daje możliwość bardziej efekty... opisywania wszelkiego rodzaju zjawisk nielinearnych zachodzących w systemie, ... ogólnie pojmowany jest przez projektanta w sposób bardziej intuicyjny. Wszyst... sprawia, że logika rozmyta staje się atrakcyjną opcją w zakresie podstaw sterowania s... plikowanymi procesami, z renderowaniem grafiki w czasie rzeczywistym na czele.

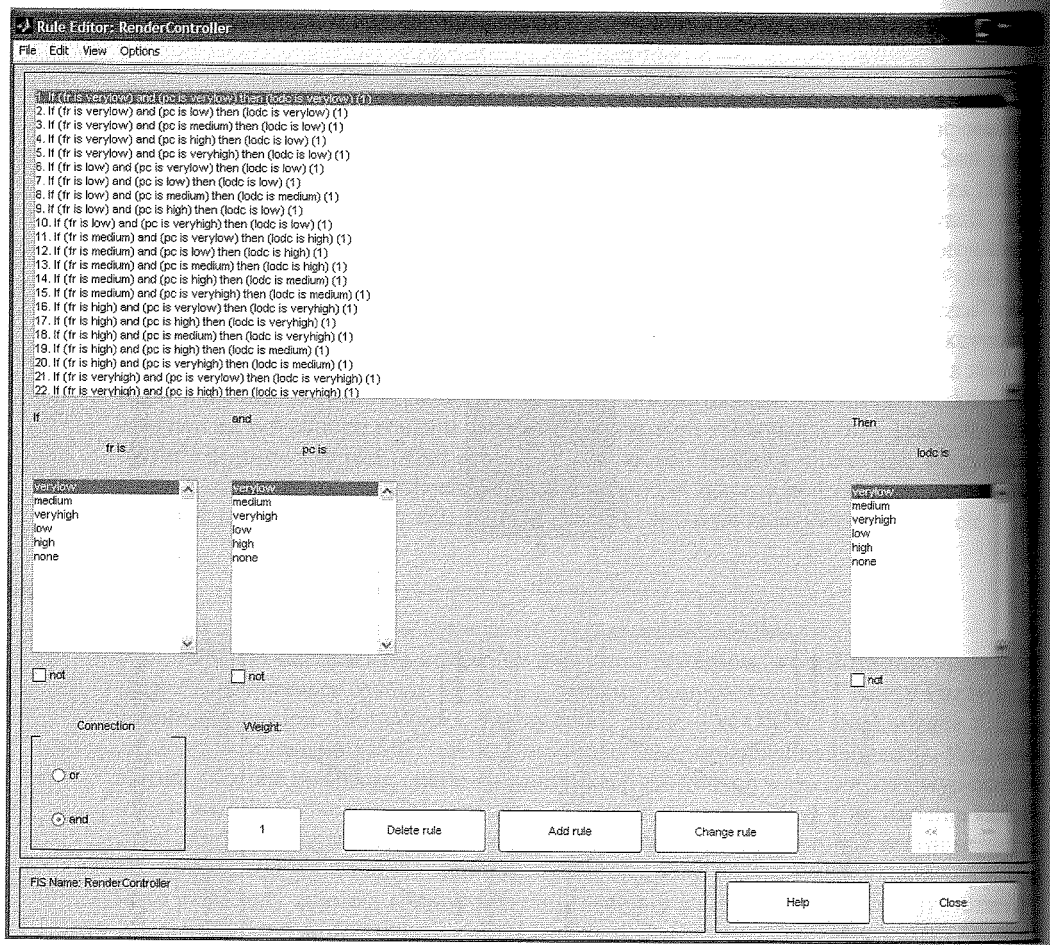
Narzędzia

Istnieje wiele narzędzi do konstruowania tzw. systemów wnioskowania rozmy... (ang. *Fuzzy Inference Systems* — FIS), z bardziej znanych wymienić należy Fuzzy L... Toolbox z pakietu MathWorks, FuzzyTech® firmy INFORM i pochodzące z fir... Aptronix środowisko FIDE™ (Fuzzy Inference Development Environment). Wszyst... oferują podobne podejście, oparte na identyfikacji zmiennych wejściowych i wyjścio... oraz konstrukcji zbioru reguł odzwierciedlających relacje między tymi dwiema kate... riami. Zmienne wejściowe i wyjściowe FIS są obiektami „rozmytymi" w tym sensie, ... odzwierciedlają wynik mapowania prostych pojęć języka naturalnego (stanowiących ... facto przykłady zbiorów rozmytych) na przedziały wartości liczbowych; wspomn... „mapowanie" ma postać funkcji, zwanej funkcją przynależności (ang. *membership functi...* określającej prawdopodobieństwo przynależności konkretnej wartości do konkretnego ... zbioru rozmytego (czyli do konkretnej kategorii pojęć). Na rysunku 3.9.1 widzimy przy... kłady mapowania różnych poziomów szczegółowości renderowania (BARDZO MAŁY, ... MAŁY, ŚREDNI, DUŻY, BARDZO DUŻY) na konkretną wartość, jaką może być ... liczba pikseli wyświetlanych na obszarze 1 mm^2 powierzchni wielokąta, czy też liczba ... wielokątów uwzględnianych w renderowaniu. Wymienione powyżej narzędzia oferują ... różne postaci funkcji przynależności — alternatywą dla funkcji „piłokształtnej", wi... docznej na rysunku 3.9.1, może być np. krzywa Gaussa.



Rysunek 3.9.1. *Funkcje przynależności dla zmiennych wejściowych i wyjściowych systemu FIS*

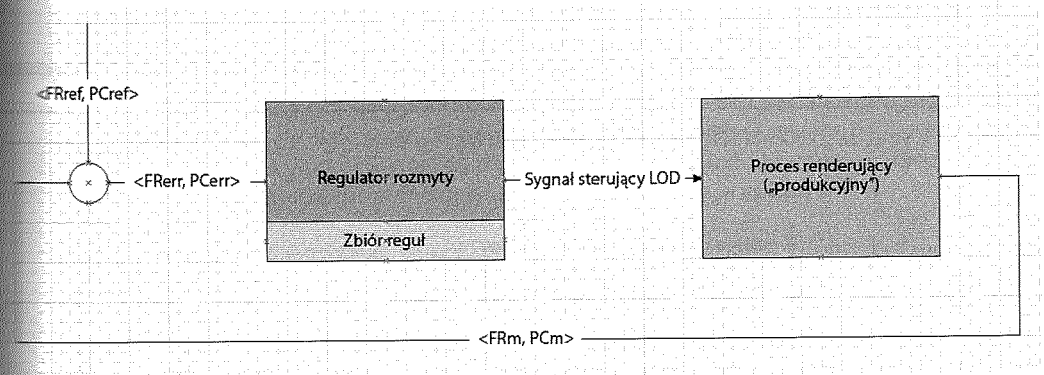
Zadaniem zbioru (bazy) reguł jest opis (w oparciu o wiedzę projektanta) zależności między „poprzednikami" (jakimi są warunki IF) a „konsekwencjami", czyli encjami THEN. Wysokiej jakości narzędzia do projektowania FIS powinny oferować edytory ułatwiające konstruowanie i edytowanie (dostosowywanie) takich reguł; zrzut ekranu jednego z takich edytorów widoczny jest na rysunku 3.9.2. Należy przy tym zaznaczyć, że skonstruowanie nie najbardziej nawet wyszukanego zbioru reguł nie jest bynajmniej tożsame ze skonstruowaniem systemu FIS: reguły te muszą zostać poddane serii testów (i przypuszczalnie wielokrotnym modyfikacjom), zanim ostatecznie zweryfikowana zostanie ich przydatność i użyteczność.



Rysunek 3.9.2. *Przykładowy zbiór reguł wiążących zmienne wejściowe systemu FIS ze zmiennymi wyjściowymi*

Projekt systemu

Na rysunku 3.9.3 widoczny jest ogólny schemat systemu wykorzystującego regulator rozmyty (ang. *Fuzzy Logic Controller — FLC*). Regulator ten sprzężony jest z procesem renderowania („produkcyjnym"), dzięki czemu uzyskuje informacje o liczbie widocznych na ekranie wielokątów i częstotliwości wyświetlania ramek. Niezwykle istotną kwestią związaną z projektowaniem takiego systemu jest jego konfigurowalność: preferowana częstotliwość odtwarzania ramek i maksymalna liczba renderowanych wielokątów to jedne z najważniejszych parametrów regulujących funkcjonowanie aplikacji. Wartości tych parametrów konfrontowane są z informacją otrzymywaną w wyniku wspomnianego sprzężenia, a wynik tej konfrontacji przekazywany jest w postaci zmiennych wejściowych do regulatora.



Rysunek 3.9.3. *System renderujący wykorzystujący regulator rozmyty*

Regulator rozmyty można potraktować jak czarną skrzynkę, obliczającą wartości zmiennych wyjściowych na podstawie wartości zmiennych wejściowych. Informacja wyjściowa regulatora jest następnie przekazywana do procesu renderującego jako sygnał sterujący wyborem LOD, w konsekwencji czego może zmienić się obciążenie silnika graficznego w związku z renderowaniem wielokątów.

LOD jest podstawowym mechanizmem, za którego pośrednictwem FLC steruje obciążeniem systemu renderującego; sam FLC jako taki nie reprezentuje wprawdzie żadnego algorytmu ani heurystyki związanej z redukcją złożoności geometrycznej, ale jego działanie ukierunkowane jest właśnie na tę redukcję poprzez sprzężenie z procesem renderującym. W książce [Luebke03] znajdą czytelnicy obszerny przegląd różnych technik LOD, zagadnień pokrewnych i ich zastosowań w aplikacjach graficznych czasu rzeczywistego. W tym miejscu ograniczymy się tylko do stwierdzenia, że do najpowszechniej stosowanych technik LOD należą techniki dyskretna i ciągła. Istotą techniki dyskretnej jest różnicowanie w czasie *liczby* uwzględnianych wielokątów, przy niezmienionym (predefiniowanym) sposobie renderowania konkretnego wielokąta; kryterium ustalania wspomnianej liczby może być odległość („głębokość") poszczególnych wielokątów lub ich zagęszczenie na ekranie. W efekcie otrzymujemy metodę nieskomplikowaną i łatwą w implementacji, jednakże jej skuteczność uwarunkowana jest w dużym stopniu zdolnościami projektanta-grafika, bowiem zbyt duże różnice między kolejnymi poziomami mogą prowadzić do wyraźnych zakłóceń w płynności odtwarzania. W metodzie

ciągłej unika się tego problemu za pomocą „wygładzania" procesu przełączania mów poprzez *stopniową* redukcję szczegółów w miarę wzrostu odległości obiekt dodatkowo zastosowanie „siatkowego" renderowania o zróżnicowanych rozdzielc także przyczynia się do zmniejszenia złożoności geometrycznej, bez zauważalnych gatywnych konsekwencji, nawet przy małej liczbie wielokątów.

System opisywany w niniejszym rozdziale wykorzystuje ciągły mechanizm LOD dla trójwymiarowych obiektów świata wirtualnego. Wyjście regulatora — zmienna sterująca — pełni rolę mnożnika dla ustalenia LOD i mnożnik ten czynnik dodatkowy, oprócz wspomnianej już odległości wielokąta od obs Dzięki owemu dodatkowemu czynnikowi można uzyskać inną niż liniową — kładniczą — zależność liczby wielokątów redukowanych (ignorowanych) odległości. Fragment kodu na listingu 3.9.1 pokazuje, jak można coś takiego plementować.

Listing 3.9.1. *Nieliniowy zanik poziomu szczegółowości uzyskany dzięki sprzężeniu silnika ren z regulatorem rozmytym*

```
void setCLODFalloff(float fallOFFMultiplier)
{
   // Dla każdego węzła siatki w scenerii
   {
      // Obliczenie odległości obiektu od kamery
      // Wyrażenie odległości w kategoriach względnych
      float fRatio = (fDepth-fNear)/(fFar-fNear);

      // Gdy żądamy nieliniowego zaniku szczegółowości
      fRatio = Mathf::Pow(fRatio, 2.0f);

      int newIndexCount = (int)(maxIndex * fRatio * (1.25/fallOffmultiplier));

      CLODObject->SetIndexCount() = newIndexCount;

      // ...
   }
}
```

Wykorzystując ciągłą technikę redukcji LOD, można osiągnąć efekt „gładkiego" przejścia między kolejnymi poziomami szczegółowości, bez niepożądanych artefaktów wizualnych w czasie wyświetlania animacji. A to spełnia warunki podstawowej zasady projektowania systemów renderujących, zgodnie z którą najważniejsze jest zachowanie poziomu jakości obrazu. Opisywany system skonstruowany został za pomocą narzędzi typu *open-source*; regulator rozmyty zaimplementowano na bazie biblioteki FIS (w języku C) dostarczonej przez Matlab®.

Zastosowanie do gier

Fuzzy Logic Toolbox z pakietu MathWorks [Matlab1] stanowić może doskonały punkt wyjścia jako narzędzie do konstrukcji systemu FIS wykorzystywanego w grach komputerowych. Przy tworzeniu tego systemu należy jedynie pamiętać o dwóch następujących rzeczach.

- Tworzony system wyposażony jest w predefiniowane reguły wiążące ze sobą obciążenia w zakresie renderowania wielokątów i częstotliwość odtwarzania ramek. Reguły te wydają się fundamentalne, co jednak nie wyklucza ich optymalizacji pod kątem odmiennych zastosowań i różnych platform sprzętowych.
- Konieczne jest uzyskanie pewnego punktu odniesienia, w postaci rezultatów początkowych testów systemu, na potrzeby przyszłego dostosowywania zbioru reguł, aż do uzyskania zadowalających wyników.

Pakiet MathWorks zawiera wygodne narzędzie do prototypowania aplikacji opartych na logice rozmytej — jest nim środowisko Simulink® [Matlab2]. Kreowanie aplikacji w tym środowisku sprowadza się podczas tworzenia odrębnych bloków konstrukcyjnych (ang. *building blocks*) i łączenia ich następnie w spójny system. Nawet prosty plik modelu Simulink ilustruje koncepcję renderowania kontrolowanego przez FLC. Możliwe jest używanie różnych generatorów sygnałów symulujących zróżnicowane obciążenie procesora — zróżnicowane wskutek różnych czynników zewnętrznych, takich jak ruch kamery, kreowanie nowych encji czy włączanie do scenerii nowej kategorii obiektów. Alternatywą dla użycia generatorów sygnałów wejściowych jest zastąpienie abstrakcyjnego procesu „produkcyjnego" rzeczywistą aplikacją (grą) w celu oceny prawdziwej wydajności FLC; umożliwiają to tzw. S-funkcje służące do kreowania „otoczek" (ang. *wrappers*) dla aplikacji zewnętrznych.

Warto również zauważyć, że nie sposób utworzyć FLC, który zdolny będzie do uporania się z obciążeniem przekraczającym możliwości platformy sprzętowej. Co prawda, projektanci i użytkownicy mają możliwość określania preferowanej częstotliwości odtwarzania ramek i maksymalnej liczby wielokątów (co pokazano na rysunku 3.9.3), lecz wartości te nie mogą wykraczać poza zagregowaną moc obliczeniową wspomnianej platformy. Stąd zalecenie, by na początku przeprowadzić testy umożliwiające ocenę tej mocy i nadanie rozsądnych wartości wspomnianym parametrom.

Założenia

W trakcie dotychczasowych rozważań przez cały czas przyjmowaliśmy milcząco założenie, iż zapotrzebowanie aplikacji na moc obliczeniową wyznaczone jest głównie przez geometryczną złożoność scenerii. Założenie to jest jednak o tyle uproszczone, iż na typową aplikację składa się znacznie więcej procesów — łączność sieciowa, sztuczna inteligencja, generowanie dźwięku, symulowanie zjawisk fizycznych itp. — także mających swój udział w budżecie czasowym każdej odtwarzanej ramki. Co więcej, nowoczesne rozwiązania sprzętowe (takie jak układy cieniujące — „shadery") sprawiają, iż związek między liczbą wielokątów a poziomem szczegółowości ich renderowania staje się bardziej skomplikowany, niż wynikałoby to z prostego mapowania pikseli i cieniowania powierzchni piksel po pikselu.

Nie zmienia to jednak ogólnej zasady, że zasady logiki rozmytej mogą być użyteczne w kontekście sterowania rozmaitymi procesami, także niewiele mającymi wspólnego z grafiką i jej renderowaniem. Wystarczy jedynie dostarczyć do FLC informację (w postaci zmiennych wejściowych) o tychże procesach i utworzyć stosowny zbiór reguł wiążący wartości zmiennych wyjściowych z określonymi kombinacjami zmiennych wejściowych, jak to wcześniej opisywaliśmy. Niezaprzeczalnie wymaga to adekwatnej wiedzy projektanta i należytej staranności w zakresie formułowania rzeczonych reguł.

Uwagi implementacyjne

Mimo iż mechanizm ciągłego dostosowywania LOD dostarcza środki dla bardziej [illegible] nego sterowania obciążeniem geometrycznym silnika, to jednak nie należy zap[illegible] iż same obliczenia związane z wyznaczaniem i przełączaniem LOD także wymaga[illegible] obliczeniowej i jako takie mogą mieć negatywny wpływ na płynność odtwarzania [illegible] Zalecane jest więc wykonywanie wspomnianych obliczeń nie dla każdej ramki, [illegible] dłuższych okresów czasowych. Utrzymywanie niezmienionego LOD w dłuższym [illegible] sie ma także tę zaletę, iż duże (zazwyczaj) podobieństwo sąsiednich ramek (wy[illegible] z ich logicznej spójności w kontekście scenerii) można wykorzystywać do opty[illegible] ich wyświetlania. Beztroskie przełączanie LOD co kilka ramek (lub nawet w [illegible] ramce) mogłoby natomiast powodować szkodliwe migotanie obrazu zwłaszcza [illegible] kiedy obiekty poruszałyby się w niewielkiej odległości od „oka" kamery.

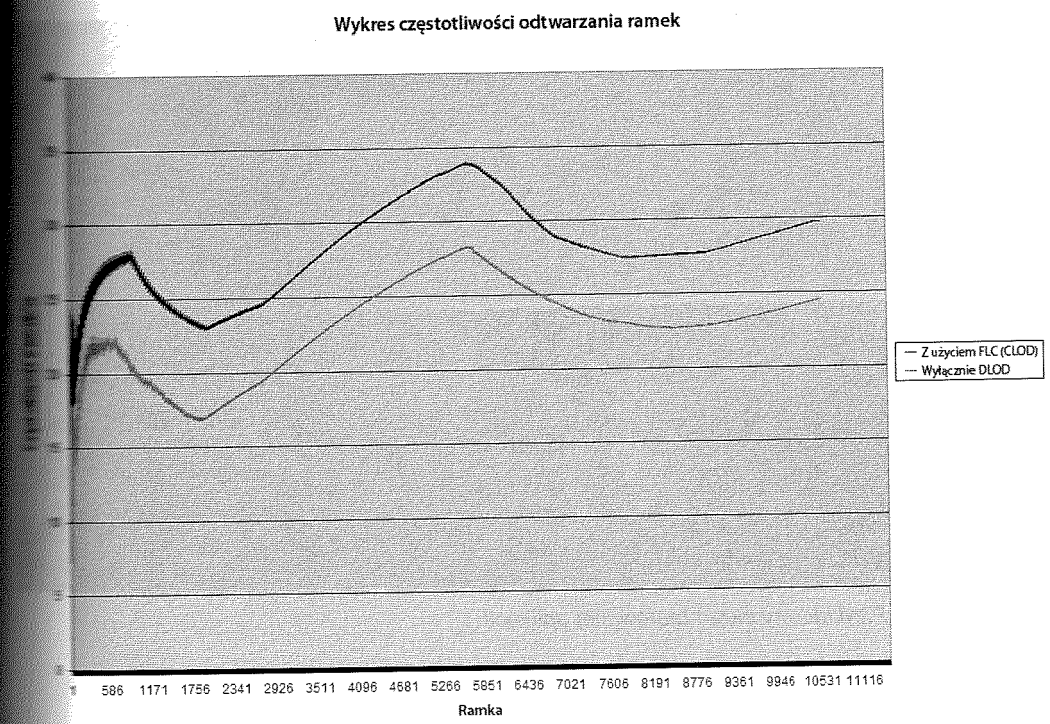
Ujemną stroną ciągłej metody dostosowywania LOD jest konieczność zbudow[illegible] formacji na temat spójności (wierzchołkowej) grafu obiektów 3D przed wysła[illegible] silnika polecenia renderowania tychże obiektów. Mimo iż informację tę można [illegible] struować offline, to i tak wpływa negatywnie na produktywność projektantów, zw[illegible] w warunkach częstych modyfikacji i restartów gry. Ponadto, ponieważ same obl[illegible] związane z ciągłą metodą LOD stanowią znaczące obciążenie dla procesora, ws[illegible] jest stosowanie tej metody jedynie dla wybranych obiektów scenerii.

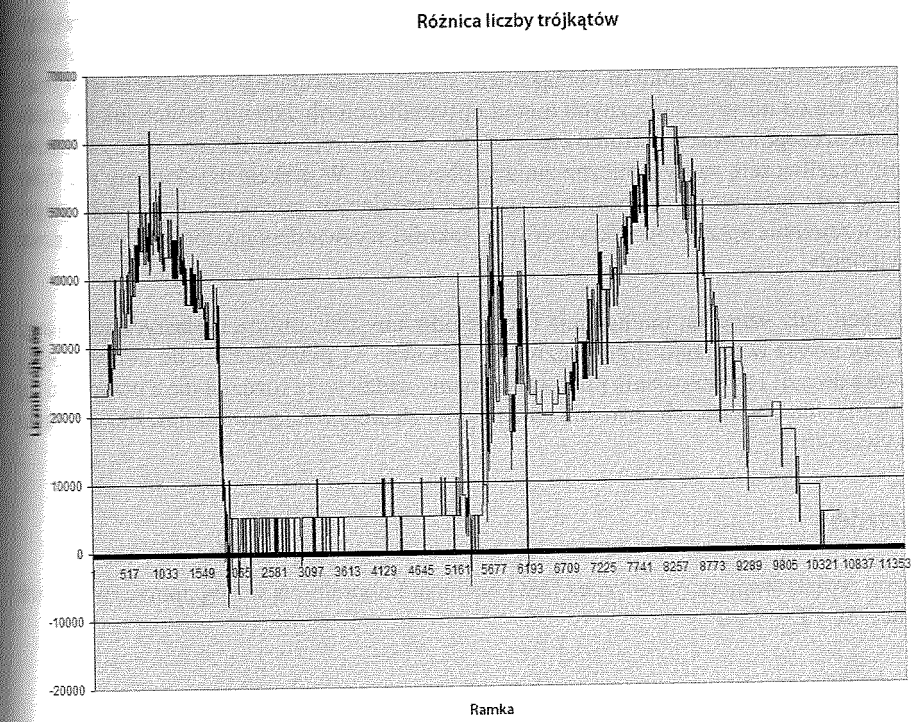
Testy i ich rezultat

Opisywane koncepcje zilustrujemy prostą aplikacją: w przykładowym świecie 3D [illegible] ściliśmy na otwartej przestrzeni populację bydła (krów), z których ok. 40% zarz[illegible] jest przez ciągłą metodę dostosowywania LOD (CLOD) — wskaźnik ów określa je[illegible] cześnie udział regulatora FLC w zarządzaniu całą sceneriią i może być — oczywiście [illegible] zmieniony stosownie do konkretnej aplikacji i kontekstu scenerii. Przykładowo o[illegible] mie, złożone obiekty najeżone szczegółami lepiej poddają się kontroli LOD niż [illegible] licznych, małych i prostych obiektów — te ostatnie wykazują znacznie większe [illegible] trzebowanie na moc obliczeniową.

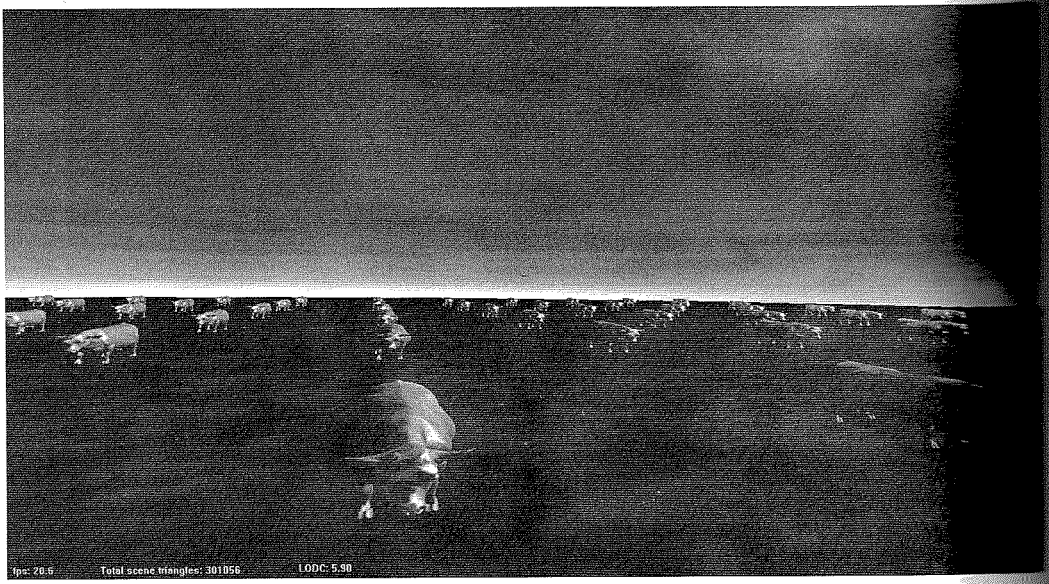
Celem przeprowadzonych testów było zarówno potwierdzenie tezy, że zastosowanie FLC istotnie przyczynia się do poprawy wydajności aplikacji, jak i przetestowanie funkcjonalności oraz efektywności FLC w zakresie sterowania sceneriią, zwłaszcza w porównaniu z przypadkiem wykorzystywania jedynie dyskretnej metody LOD (DLOD). Wyniki porównawcze dla obu opisanych podejść widoczne są na rysunku 3.9.4.

Spoglądając na rysunek, można zauważyć, że zastosowanie FLC skutkuje wyższą (o ok. 25%) częstotliwością odtwarzania ramek, w porównaniu z systemem wykorzystującym wyłącznie metodę DLOD. Oba systemy poddane zostały na początku testu identycznemu, silnemu obciążeniu wynikającemu z konieczności odtwarzania ramek w tempie 30 na sekundę. Diagram w dolnej części rysunku przedstawia względną różnicę liczby trójkątów w obu systemach dla każdej ramki — ewidentnie system wykorzystujący FLC wykazywał większą zdolność zachowywania szczegółów, mimo iż rzeczywista częstotliwość odtwarzania spadała miejscami do ok. 22 ramek/s. Przewaga ta konsekwentnie utrzymywała się w dalszym ciągu testu, w stopniu zależnym od geometrycznej komplikacji scenerii. Przykładowy zrzut ekranu z testowanej aplikacji widoczny jest na rysunku 3.9.5 oraz na kolorowej ilustracji nr 6.





Rysunek 3.9.4. *Porównanie efektywności dwóch różnych systemów renderujących*



Rysunek 3.9.5. *Przykładowy ekran testowanej aplikacji*

Konkluzja

W niniejszym rozdziale opisaliśmy zalety regulacji rozmytej zastosowanej do zarz
nia złożonością scenerii gier; dzięki tym zaletom możliwe okazało się skonstruo
regulatora sterującego ogólnym obciążeniem silnika renderującego. Podejście to w
nie różni się od innych klasycznych metod optymalizacji obciążenia, takich jak ko
poziomu szczegółowości i redukowanie zbioru widocznych obiektów, istotnym s
gółem: wrażliwością na dynamiczne zmiany obciążenia. W efekcie uzyskujemy nie
redukcję szczegółowości wyświetlania na zadowalającym poziomie, lecz także
dzenie wzajemnie sprzecznych kryteriów — jakości wyświetlanego obrazu i wydaj
wyświetlania.

W sytuacji, gdy sztuczna inteligencja i różne odmiany tzw. obliczeń elastycznych (
soft-computing) w rodzaju logiki rozmytej stosowne są powszechnie do celów p
mowania decyzji i symulowania pola walki, mamy nadzieję, iż rozdział nasz przyczy
do poszerzenia obszaru ich zastosowań na optymalizację renderowania grafiki w c
rzeczywistym.

Podziękowanie

Opisany projekt wspierany jest przez DSO National Laboratories w Singapurze, w ra
grantu DSOCL01144.