УДК 004.85, 004.89

Ю.В. Крак, О.В. Бармак, Б.А. Троценко

ТЕХНОЛОГІЯ ОПТИМІЗОВАНОЇ ПЕРЕДАЧІ ЖЕСТОВОЇ МОВИ В МЕРЕЖІ ІНТЕРНЕТ

Запропоновано метод передачі змодельованого процесу анімації мови жестів глухих у мережі Інтернет, який дозволяє ефективно передавати інтерактивні медіа дані на клієнтський комп'ютер. Реалізовано принципи оптимізації передачі даних.

Вступ

Кількість людей з вадами слуху, для яких потрібно розробляти сучасні засоби навчання і спілкування відповідно до світового науково-технічного розвитку – становить мільйони. Розвиток сучасної науки, комп'ютеризація суспільства, використання мультимедійних та Інтернет технологій створили достатні умови для розробки комп'ютерних систем комунікації цих людей у формах і образах близьких і зрозумілих для них і для оточуючого середовища.

Основною формою спілкування глухих є мова жестів. На користь їм можуть стати комп'ютерні технології: у [1] обґрунтовується необхідність розробки та концепція комп'ютерної інформаційної технології невербального спілкування людей з вадами слуху (нечуючими та слабочуючими) з можливістю реалізації у її рамках крім іншої і такої функціональності:

- відображення на обличчі тривимірної комп'ютерної моделі людини мовленнєвого процесу з урахуванням емоційних складових;
- синтез рухів мови жестів глухих та дактильної абетки на трьохмірній моделі людини.

Подібні мультимедійні системи, що мають велику базу даних аудіо- та візуальної інформації, в обробленому та передобробленому виглядах, на основі якої будуть будуватись певні відповіді на запити клієнта, наприклад, такі як «показати фрагмент заданого тексту візуально». У випадку, коли потрібно «перекласти» певний фрагмент тексту зі звичайної мови на мову глухонімих, доцільно зберігати не відео-фрагмент відповіді, а лише таку інформацію, з якої дану відповідь можна отримати за розумний час алгоритмічно. Перевагами даного підходу, зокрема, є: відносна компактність бази даних, що стосується задачі в порівнянні зі збереженням готової інформації; можливість видозмінювати та доповнювати базу даних; можливість отримання медіа виводу різної якості в залежності від потужностей або потреб.

Зважаючи на соціальний запит до подібних систем, їх слід зробити якомога найбільш розповсюдженими, для чого доцільно використати можливості мережі Інтернет.

Оскільки обчислювальні можливості комп'ютерів широкого кола користувачів значно різняться, так само як і значно різняться можливості їх каналів зв'язку, даний аспект слід урахувати при продумуванні способу масового обслуговування, для передачі даних через мережу Інтернет.

Користувач, на загал, бажає або відразу почати використовувати технологію, або, у найгіршому випадку, завантажити певну розумну кількість інформації з сервера. Адже не має сенсу для потреби відображення жестовою мовою фрази «доброго дня, Україно», довжиною у 10 секунд, заважувати сотні мегабайт інформації бази даних для препроцесингу, а також спеціалізоване програмне забезпечення для його обробки та відображення. Зазначимо також таку рису задачі, як: потрібний результат відносно легко формується з бази даних на єдиному комп'ютері для кожного конкретного запиту, водночас як зберігання готового відео-ряду для множини всіх можливих запитів є зашироκою і нераціональною задачею.

Отже, узагальнюючи, потрібна технологія постачання інтерактивної медіа інформації на комп'ютер клієнта. Дана медіа інформація в існуючій системі не зберігатиметься у готовому вигляді, а будуватиметься з певної підготовленої до обробки інформації відповідно до вимог клієнта. Слід розробити розумний спосіб передачі відео-ряду з можливістю адаптації до потужностей цільової клієнтської машини та ширини каналу передачі даних. Слід використати компресію; при компресії надати можливість програмісту, який використовує дану технологію, враховувати природу побудови зображень для конкретної задачі.

Розробка форматів та методів швидкої передачі високоякісних зображень, які генеруються динамічно, в мережі Інтернет – досить актуальна задача. Існує ціла лінійка програмних Інтернет-технологій які призначені для 3D-візуалізації предметного та сценічного простору, для інтерактивного та динамічного відображення графічних об'єктів. До найбільш відомих варто віднести:

- Java [2] – внесла новий сенс інтерактивним здібностям Web. У Java багато областей застосування, тривимірність лише одна з них. На сайті Silicon Graphics (www.sgi.com/fun/java) є посилання на опис Java-системи VisAnimator, яка служить для інтерактивної візуалізації та анімації в Інтернеті. Результат роботи аплета VisAnimator – це вікно з 3D-об'єктом, який обертається, під яким знаходяться три кнопки керування: "Зупинити", "Запустити", "Змінити швидкість";
- технологія MetaStream [3] дозволяє дискретно наближувати та віддаляти об'єкт, переміщувати його на площині, виділяти область зображення з тим, щоб показати її загальним планом. Ця технологія зарекомендувала себе на рекламному ринку Інтернета завдяки простоті та функціональності;
- Flash [4] – технологія виробництва візуальних ефектів. Часто Flash застосовують для динамічних заставок, організації багатофункціонального та розгалуженого меню. З допомогою Flash робляться і тривимірні сцени.

Аналізуючи існуючі технології можна прийти до висновку, що вони або не мають інтерактивності, або для досягнення її слід використовувати спеціальні плагіни, які необхідно інсталювати на комп'ютер клієнта. Це пов'язано з тим, що чим простіший спосіб відтворення, тим більше інформації слід зберігати. І навпаки: чим компактніше представлена інформація, тим більше обчислювальних витрат потрібно для її використання. Ще недавно, при використанні анімації персонажа в мережі Інтернет або в інших комп'ютерних мережах потрібно було відповісти на питання: що більш критично для метода представлення руху – компактність чи швидкість обчислень? Тому чим компактніше буде представлена інформація, тим швидше вона передасться по мережі і, отже, користувач не побачить небажаних затримок у русі. Але з іншого боку швидкість обчислень також важлива. Який сенс швидко передавати інформацію про рух, якщо перед тим як відобразити рух, комп'ютер буде вимушений провести обчислення, які займають більше часу, чим передача інформації? На думку авторів теперішня швидкість каналів передачі інформації та швидкодія комп'ютерів дозволяє кінцевим користувачам комфортно користуватись інтерактивними медіаданими, для чого потрібно розробити правильний програмний засіб. Потреба в такому засобі продиктована наступними практичними задачами:

- навчальні задачі-тренажери (імітатори виробничих та інших процесів) – в цих задачах анімація генерується динамічно, в залежності від сценарію та команд користувача; для реалізації цих задач потрібно використовувати існуючі засоби (Flash, VRML [5] тощо) і в клієнта має бути завантажений відповідний плагін (не завжди безкоштовний);
- задачі, пов'язані з передачею відео-потоків, які формуються динамічно, в залежності від команд користувача.

У даній роботі пропонується підхід який дозволятиме вирішувати такого типу задачі без необхідності прив'язки до обмеженого списку стандартних засобів та

без необхідності у клієнта використовувати відповідні плагіни для браузера.

Постановка задачі

Слід запропонувати узагальнений засіб оптимальної передачі динамічного створеного зображення в мережі Інтернет з можливістю керування ним. Для цього слід розробити програмну реалізацію Web-застосування з наступною функціональністю:

1) серверна частина застосування має містити базовий метод створення зображення у пам'яті (для можливості наступного рендерінгу трьохмірним API у цю область);

2) мають бути реалізовані методи динамічної зміни цього зображення (зміна формату зображення, оконтурювання зображення тощо);

3) має бути реалізована можливість динамічного виводу цього зображення у стандартний елемент управління HTML-сторінки;

4) має підтримуватись протокол обміну між HTML-сторінкою та серверною частиною для керування процесом рендерінгу (динамічна зміна формату зображення, обертання сцени, зміна перспективи тощо).

Концептуально потрібна функціональність, яка показана на схемі (рис.1).

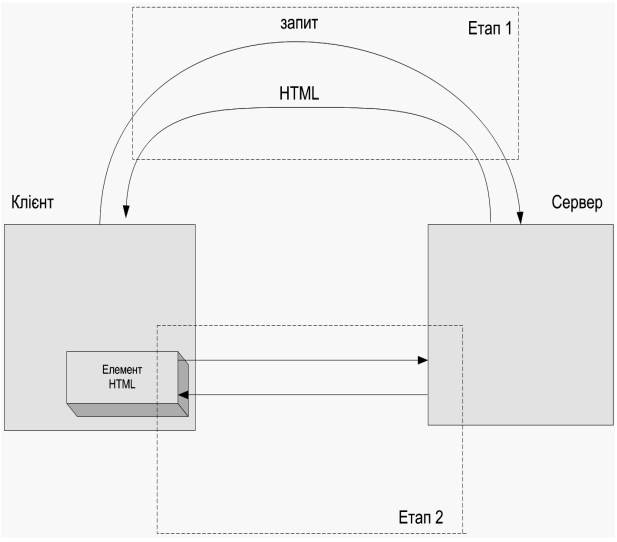

Рис. 1. Схема передачі анімаційних зображень у мережі Інтернет

Реалізація технології

Для реалізації технології запропоновано використовувати платформу DOT.NET, тобто серверну частину реалізовувати на ASP.NET. Для реалізації динамічної передачі зображень з сервера та отримання запитів від клієнта пропонується використовувати технологію Silverlight [6]. Вибір саме таких засобів обумовлено наступними перевагами:

- ASP.NET пропонує повністю об'єктно-орієнтовану модель програмування, яка включає у себе архітектуру, що керує подіями, яка базується на елементах управління; ця архітектура забезпечує інкапсуляцію та повторне використання коду;
- ASP.NET підтримує всі основні пристрої та браузери.

Але основною перевагою, з точки зору потрібної функціональності є поява в ASP.NET нової технології Silverlight. Вона працює з використанням підключеного до браузера модуля та надає скорочену версію бібліотеки класів .NET Framework. В цю скорочену версію входить міні-версія WPF технології, якою розробники користуються для створення інтерфейсів користувача наступного покоління для операційних систем сім'ї Windows. Silverlight дозволяє створювати більш функціональні сторінки, ніж можна створювати використанням лише за допомогою HTML, DHTML або JavaScript.

Для реалізації серверної та клієнтської частин використано об'єктно-орієнтовний підхід. Серверна частина реалізована на ASP.NET.

Клієнтська частина, очевидно, не може бути стандартним компоненом браузера. Реалізація поставленої задачі включатиме наперед непередбачувану поведінку обробки зображень за допомогою наперед незазначених алгоритмів. На сьогоднішній день, це звужує коло базових технологій клієнтської частини до Microsoft Silverlight, Adobe Flash, та Sun Java Applets. Для забезпечення найбільшої зручності розробки пари сериалізації та десериалізації потоку даних було вибрано Microsoft Silverlight для клієнтської части-

ни, та Microsoft ASP.Net для серверної частини.

Робота технології на клієнтській частині складатиметься з трьох етапів:

1) браузер клієнта завантажує Silverlight-код відтворення відео, що включає у себе спеціалізовані для конкретної задачі алгоритми десеріалізації потоку кадрів;

2) Silverlight-код отримує певну порцію загальних даних, якщо такі є, і здійснює препроцесинг, який закладений в алгоритмі. Разом з ними він отримує і 128-бітний ідентифікатор сесії передачі даних, який буде потрібен на наступному етапі;

3) Silverlight-код, по-перше, продовжує отримувати один за одним команди, за допомого яких будуються фрейми; по-друге, дає можливість серверу через окреме з'єднання передавати інформацію, яка визначає або дає можливість визначити швидкість передачі даних.

Оскільки велика кількість користувачів знаходиться у середовищі, захищеному проксі-сервером, то вибір було зроблено на користь HTTP-протоколу, який або використовується напряму, або підтримується всіма проксі-серверами. Альтернативою могло б бути пряме TCP-підключення.

Для двосторонньої передачі даних між клієнтом та сервером з використанням HTTP-протоколу, а зокрема, – передачі даних з сервера та нотифікації сервера про момент отримання пакета даних використовувався наступний підхід:

- клієнт (silverligth-код) надсилає HTTP-GET запит. У відповідь він отримує не наперед встановлену кількість даних, а потік, що передається фрагментами ("chunked");
- клієнт (silverlight-код) надсилає HTTP-PUT запит. Аналогічно, кількість даних, що буде передана, не встановлюється, потік передається фрагментами ("chunked").

На багатьох серверах, у тому числі на IIS, який потрібен для використання платформи APS.Net, у разі застосування даного підходу виникає проблема, пов'язана з віртуалізацією обробки запитів. Зокрема код, який буде обробляти перший, типу HTTP-GET, запит не має доступу та безпосереднього зв'язку до коду, що оброблятиме другий, типу HTTP-PUT, запит; їхні адресні простори розділені. Подібна віртуалізація, яка доцільно реалізо-вана з точки зору безпеки, у даному випадку ускладнює потрібну комунікацію.

Зважаючи на те, що нам потрібно всього лиш повідомляти перший код про доставлені дані, а щонайбільше передавати обмежену кількість команд, було використано об'єкти ядра, що дозволяють міжпроцесну комунікацію; найдоцільніший у даному випадку – іменований канал передачі даних: перший код (обробник HTTP-GET запиту) вибирає ім'я (текстове представлення 128-бітного випадкового числа), створює іменований канал і запускає потік читання даних з нього. Ім'я каналу передається у відповіді на HTTP-GET. Клієнт отримає ім'я і створить інше підключення типу HTTP-PUT. Обробник даного підключення прочитає ім'я іменованого каналу, відкриє такий канал і записуватиме туди отримані команди. Таким чином перший обробник буде проінформований.

Архітектура системи

Було створено і побудовано інформаційну модель процесу, що оптимізується.

На першому (найвищому) логічному рівні, відео інформація передається з серверної сторони на сторону клієнта.

На наступному, другому, логічному рівні, сервер закодовує відео інформацію фрейм за фреймом, а також може передбачати певну ініціалізацію клієнта, клієнт — проводить ініціалізацію, а також відтворює картинку на основі команд. На цьому логічному рівні програміст (інженер, розробник) може враховувати специфіку задачі і структруру медіа даних, тобто надати власну пару компресора-декомпресора даних. На цьому рівні окремий потік відповідає за обслуговування іменних каналів та передачу команд компресора.

Рекомендований рівень компресії виходить з розрахунку, що 10 відсотків часу канал має або простоювати, або резервується на непередбачену ситуацію,

наприклад, передачу даних іншим програмним забезпеченням на стороні клієнта.

На наступному, третьому, логічному рівні, команди сериалізуються на стороні сервера і десериалізуються на стороні клієнта. Це є останній логічний рівень даної технології. Інформація на даному рівні передається використанням HTTP (Silverlight/Asp.net).

Оскільки код, який буде виконуватися на стороні клієнта, завантажується або щоразу при передачі сторінки, або використовується з кешу при наявності (на відміну від постачання клієнта окремого програмного забезпечення), дана технологія не потребує передбачення версій протоколу: при зміні останнього (наприклад, розширення), модифікується також і код, що буде виконуватись на стороні клієнта.

Будемо вважати основною ту частину, яка генерує медіа потік. У розробленій технології є підтримка встановлення бажаних параметрів (розмір, частота кадрів), але остаточне рішення надасть клас, який генерує медіа потік.

Оскільки ми маємо можливість надсилати програвач кожного разу при завантаженні сторінки, відпадає необхідність підтримки версії протоколу.

Послідовність ініціалізації:

- користувач вводить дані, що визначають вміст медіа інформації (наприклад, фразу, яку слід показати на мові жестів);
- клієнтська частина запитує клас відображення про бажані параметри;
- бажані параметри та параметри ініціалізації надсилаються на сервер;
- на сервері створюється клас, що генеруватиме медіа потік;
- серверний клас здійснює ініціалізацію, валідує параметри;
- дані ініціалізації серверного класу і валідовані параметри надсилаються на клієнтську сторону;
- клієнтська сторона здійснює ініціалізацію, встановлює додаткове з'єднання для інформування про надіслані команди і отримання фреймів.

Класи реалізації

Загальні класи, які використовуються технологією, показані на рис. 2.

Статичний клас BinaryIOTools містить методи передачі даних найнижчого рівня для даної технології. Ці методи використовуються серверними та клієнтськими класами обслуговування передачі даних і включають в себе читання та запис масивів байт, рядків, асоціативних масивів рядків та глобальних ідентифікаторів.

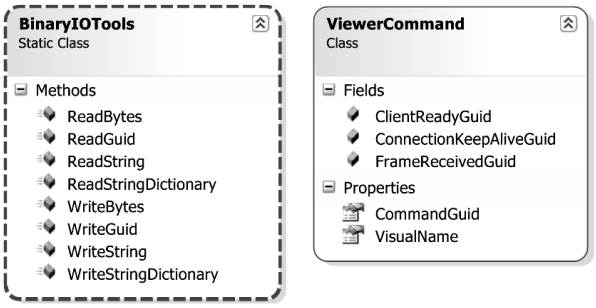

Рис. 2. Класи, які використовуються

Клас ViewerCommand інкапсулює команду, яку клієнт надсилає на сервер. Окремо виділені інформаційні команди підтримки сесії, повідомлення про закінчення ініціалізації клієнтської сторони, повідомлення про отримання фрейма.

Інтерфейси

Інтерфейс IFramesProducer (рис. 3) створено для роботи з класами, які транслюють медіа дані. За допомогою нього отримуються медіа дані з вищого (ніж дана технологія) рівня. Практично даними може бути довільний відео-потік, або клас, де зображено віртуальну людину, яка говорить мовою жестів.

Передбачено задання бажаних для клієнта розміру фрейма та частоти його зміни через методи SetPreferableFrameSize та SetPreferableFrameRate відповідно, але архітектурно останнє слово залишається за класом-наслідником IFramesProducer: дійсні розмір фрейма і частота його зміни, яку можна отримати через методи GetFrameSize, GetFrameRate. Передбачається що виклик цих методів є підготовчим етапом ініціалізації класу-транслятора.

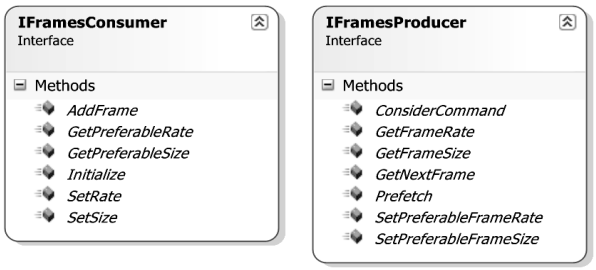

Рис. 3. Інтерфейси

Безпосередньо для ініціалізації передбачено метод Prefetch. У цьому методі клас-транслятор може, наприклад, створити підключення до бази даних і завантажити у відео-пам'ять базовий меш трикутників. Даний метод повертає масив байт, які будуть передані на сторону клієнта для ініціалізації. Конкретна реалізація цих інтерфейсів може, для прикладу, передавати базові налаштування голосу, спрайти — все, що буде використане для побудови фреймів.

Робочий режим реалізується через метод GetNextFrame. Через параметри він повертає асоційований масив рядків "ключ-значення" та масив байт. Передбачається, що саме масив байт визначатиме побудову фрейма, а асоційований масив містить опціональні параметри відображення. Для рапортування про кінець трансляції сервер повертає порожній масив байт.

Передбачено вплив на трансляцію у процесі. Асинхронно з викликами методів Prefetch і GetNextFrame може бути виклик методу ConsiderCommand. Цьому методу передається ідентифікатор команди, отриманої від клієнтської сторони.

Інтерфейс IFramesConsumer створено для роботи з класами, які приймають і відтворюють трансляцію. Передбачено отримання бажаних розміру фрейма та частоти його зміни через методи GetPreferableSize та GetPreferableRate, але зрештою буде виконане налаштування на дані транслятора методами SetSize та SetRate.

Ініціалізація клієнтської сторони проводиться у методі Initialize. Цей метод буде виконаний після ініціалізації серверної частини і йому буде передано масив байт, отриманий з методу Prefetch. Після ініціалізації рекомендується надсилати серверу інформаційну команду закінчення ініціалізації.

Робочий режим клієнта є серією викликів AddFrame. Даному методу передається асоціативний словник та масив байт, які отримані від виклику GetNextFrame серверної частини. Після отримання фрейма рекомендується надсилати інформаційну команду отримання фрейма.

На стороні сервера

Клас ServerSideStreamer (рис. 4) реалізує логіку передачі медіа даних на сторону клієнта. Його конструктору слід передати канал виводу інформації (наприклад, потік виводу відповіді на HTTP запит) і клас, який здійснюватиме трансляцію. Конструктор створює два потоки:

- перший потік створює іменований канал даних, у циклі отримує ідентифікатори команд та передає їх класу-транслятору;
- другий потік ініціалізує транслятор і у циклі генерує та надсилає фрейми.

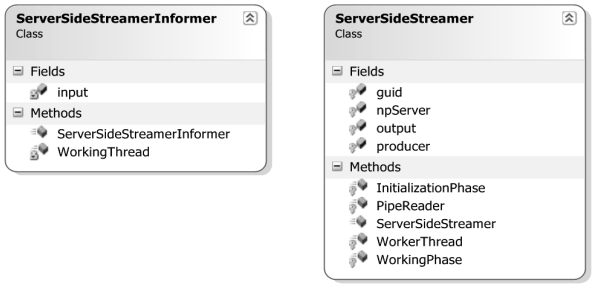

Рис. 4. Класи на стороні сервера

Таким чином, створення екземпляра класу ServerSideStreamer не блокує потік, який створюється.

Клас ServerSideStreamerInformer створено для інформування класу-транслятора про отримані команди. Конструктору класу передається потік, з якого приходять ідентифікатори команд. За домовленістю, перший ідентифікатор є іменем іменованого каналу, через який команди будуть передаватись класу ServerSideStreamer. Цей клас передбачено для обробника HTTP-put запиту.

На стороні клієнта

Клас ClientSideServerInformer (рис. 5) створений для зручного надсилання команд з клієнтської сторони на серверну. Конструктору даного класу передається потік виводу. Конструктор створює додатковий потік для надсилання даних, чергу команд і семафор – для інформування потоку про появу елемента в черзі. Керування повертається відразу, потік, який викликає конструктор, не блокується.

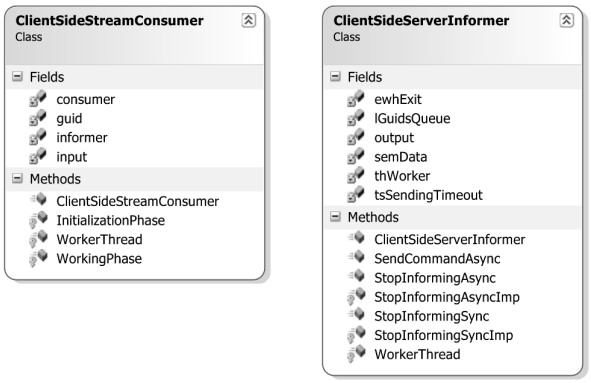

Рис. 5. Класи на стороні клієнта

Метод SendCommandAsync додає ідентифікатор команди в чергу і відразу повертає керування.

Клас ClientSideStreamConsumer є логічною парою до класу ServerSideStreamer і, відповідно, отримує з заданого протоколу дані та десеріалізує їх, ініціалізує і веде робочий цикл класу, який відображає дані, а також надсилає інформаційні команди серверній частині. Відповідно конструктор даного класу приймає потік, об'єкт класу, який реалізує IFramesConsumer і об'єкт ClientSideServerInformer.

Результати прототипного моделювання

Для тестування запропонованої технології було створене прототипне застосування (рис. 6). Як медіа дані для передачі використовувався потік змодельованого трьохмірним API анімаційного зображення морфем мови жестів. Тестування показало спроможність запропонованої технології.

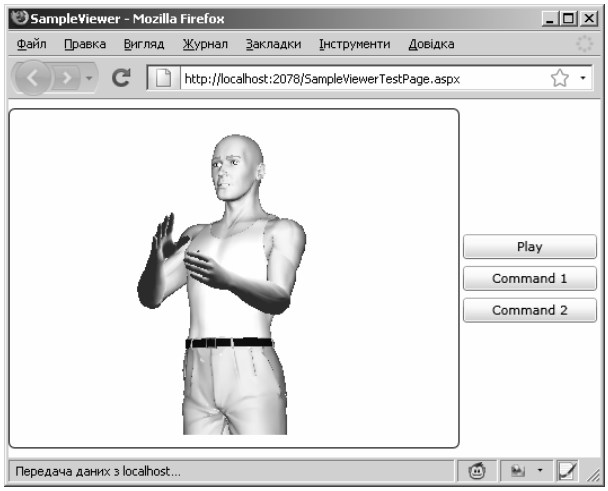

Рис. 6. Прототипне застосування технології

Висновки

У роботі запропоновано і розроблено метод передачі зображень у мережі Інтернет, який дозволяє ефективно передавати інтерактивні медіа дані на клієнтський комп'ютер. Результати прототипного моделювання показали життєспроможність технології. Розроблена технологія – універсальний засіб для передачі інформації як для універсальних, так і для спеціалізованих систем, наприклад, дистанційного навчання жестової мови.

Подальші дослідження будуть направлені на використання запропонованої технології для створення різного роду застосувань, для передачі синтезованої мови жестів у мережі Інтернет.

1. *Кривонос Ю.Г., Крак Ю.В., Бармак О.В., Тернов А.С.* Інформаційна технологія невербального спілкування людей з вадами слуху // Журн. Штучний інтелект. – 2008 – № 3. – С. 325–331.
2. www.java.com
3. www.bitek.ru/technologies/3d/metastream
4. www.adobe.com/flashplatform/
5. xml.coverpages.org/vrml-X3D.html
6. www.silverlight.net

Отримано 24.03.2009

Про авторів:

Крак Юрій Васильович,
доктор фізико-математичних наук,
професор,

Бармак Олександр Володимирович,
кандидат технічних наук,
старший науковий співробітник,

Троценко Богдан Анатолійович,
аспірант, інженер-програміст
другої категорії.

Місце роботи авторів:

Інститут кібернетики імені В.М. Глушкова
НАН України
03187, Київ-187,
Проспект Академіка Глушкова, 40.
Тел.: 8(067) 930 6752; 8(068) 172 9821;
8(068) 351 1074.
yuri.krak@gmail.com
alexander.barmak@gmail.com
modosansreves@gmail.com