УДК 004.8

РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ІНТЕРАКТИВНОГО НАВЧАННЯ ЖЕСТОВІЙ МОВІ

Ю.Г. Кривонос, Ю.В. Крак, А.С. Тернов, М.П. Лісняк

Інститут кібернетики імені В.М. Глушкова НАН України,
03680, Київ, проспект Академіка Глушкова, 40

У статті пропонується підхід до розробки програмної системи для інтерактивного навчання жестовій мові. Розглянуто базову архітектуру та складові такої системи, формат та протокол передачі даних між ними. Також описано основний модуль системи, що відповідає за синтез та анімації жестів, емоційно-мімічних проявів на обличчі людини з використанням тривимірних моделей тіла людини. Описуються основні вимоги до модуля, проблеми та інструменти їх вирішення, його архітектура та контролери, що відповідають за генерацію анімації та її обробку на графічних процесорах.

The article suggests an approach to the software system development, dedicated to the interactive learning of sign language. The article shows a basic architecture of the system, its modules, format and transmission protocol of the data. The article also describes the main module of the system responsible of the synthesis and the animation of the gestures and facial expressions of the emotions, based on the three-dimensional pattern of a human body. It describes main requirements for the module, analyzing basic problems and the implements for their solution, architecture and components, especially these ones that are responsible of the generation of the animation, its structure, the algorithm of gestures' combination and their graphical processing.

Вступ

Невербальна артикуляційна і мімічна передача інформації людиною за допомогою жестової мови стала предметом інтенсивних досліджень із середини XX ст. внаслідок виникнення світових тенденцій до інтеграції людей з вадами слуху в суспільство та у зв'язку із прогресом і загальнодоступністю комп'ютерної техніки.

У суспільстві важливою є проблема полегшення інтеграції людей з фізичними вадами. Це також стосується людей з вадами слуху, глухонімих. Такі люди переважно спілкуються жестовою мовою. Відмінності між жестовими та звуковими мовами визначає канал, по якому передається інформація. У жестових мовах інформація кодується рухами рук, тіла, обличчя, очей і сприймається за допомогою зору. Це зумовлює також спроможність жестової мови виражати емоції і почуття глухої людини за допомогою різноманітних мовних засобів ритміко-інтонаційного рисунку, емотивної лексики через експресію руху та емоційну міміку на її обличчі. Звідси випливають основні фундаментальні властивості жестових мов:

1) провідну роль відіграє простір навколо людини, яка говорить;

2) елементи жесту виконуються і сприймаються одночасно, в той час коли звукове мовлення поступово досягає вуха людини. Це дозволяє кодувати більший обсяг інформації порівняно зі словом усного мовлення.

Основною структурною одиницею жестової мови є жест. Жест у жестових мовах є аналогом звукового слова, тому деякі лінгвістичні методики можна успішно застосувати і до нього. Але жесту притаманні унікальні властивості, для опису яких такі методики не підходять, наприклад, фіксований порядок для визначення граматичного слова не потрібний, оскільки граматичні елементи в жестах виконуються одночасно [1].

У роботі [1] виділено три параметри, необхідні для опису структури жесту:

- місце виконання жесту за відношенням до тіла людини, яка промовляє;
- форма кисті руки, що виконує жест;
- траєкторія руху руки.

Зазначимо, що жестова мова є справжньою мовою з усіма її складнощами, також вона принципово і фундаментально відрізняється від розмовної мови правилами, принципами побудови речень і т. д.

На сучасному етапі розвитку сурдопедагогічної теорії виділено дві форми мови жестів – розмовну і калькуючу [1]. Кожній із них властиві певні граматичні закономірності.

Розмовна жестова мова – самостійна кінетична система. У ній наявна така кількість жестів, яка необхідна для невимушеного, неофіційного спілкування глухих людей між собою. За допомогою розмовної жестової мови передається інформація на побутові теми, про повсякденне життя. Ця мова безпосередньо пов'язана з ситуацією, на тлі якої відбувається розмова.

Калькуюча жестова мова – це система спілкування, жести в якій супроводжують усну мову того, хто говорить. Жести при цьому виступають певним еквівалентом слів, а їх розташування, в основному, відповідає будові речення у словесній мові. Цей вид жестової мови не має власної граматичної будови мови, а тільки калькує її.

Для вирішення проблеми швидкого і ефективного вивчення жестової мови необхідне створення технологій для комп'ютеризації цього навчання в силу доступності та сучасності такого підходу з урахуванням бурхливого розвитку графічних процесорів та технологій для побудови тривимірної графіки.

Важливим етапом розвитку графічних процесорів та посилення їх значення для персональних комп'ютерів було створення високорівневих шейдерних мов, які дозволяли створювати складні алгоритми для відображення тривимірної графіки, підвищуючи реалістичність зображення [2]. Завдяки такому розвитку відеопроцесорів стала можливою розробка аватарів – віртуальних агентів для взаємодії людини з комп'ютером (human-computer interaction). Фундаментальними характеристиками аватарів є реалістичний вигляд та рухи, емоційна міміка та артикуляція, природнє промовляння слів та фраз. Основними проблемами при розробці аватарів є моделювання якісної тривимірної моделі, анімація реалістичних рухів, виразів обличчя, артикуляції, синтез природньої мови, синхронізація анімації зі звуком. Аватари використовуються в різних сферах, особливо в галузі розробки програмного забезпечення для людей з вадами слуху [3, 4]. У даній роботі розглядаються проблеми та особливості реалізації аватара для відтворення української жестової мови для побудови системи інтерактивного навчання жестовій мові.

Основні вимоги до програмного комплексу інтерактивного навчання жестової мови

Програмний комплекс для інтерактивного навчання українській жестовій мові складається з декількох підсистем, що відповідають за анімацію, звук, розпізнавання тощо (рис. 1).

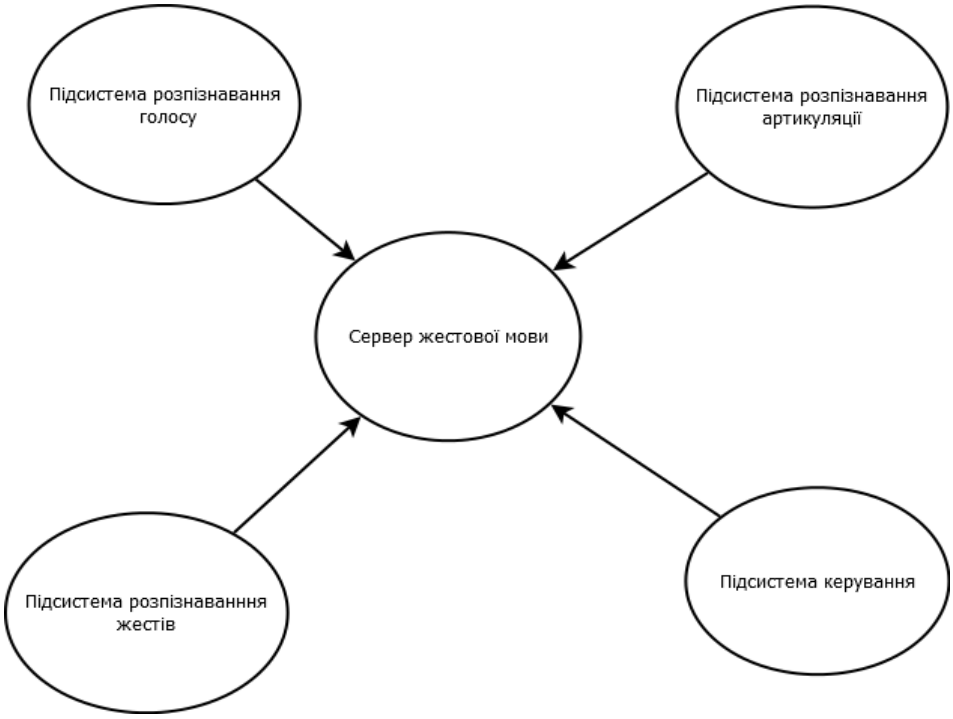

Рис. 1. Високорівнева структура програмного комплексу

У цій статті розглянуто моделі даних та протоколи їх передачі для цього програмного комплексу, а також більш детально описується сервер жестової мови, який, у свою чергу, теж розбитий на окремі слабко зв'язані між собою модулі.

Оскільки кожна з цих підсистем є своєю окремою досить об'ємною як науковою, так і алгоритмічно-програмною задачею, основна вимога до побудови їх у комплексі – зробити їх максимально незалежними один від одного та від основної підсистеми, забезпечити можливість їх розробки різними людьми, на різних мовах програмування, але при цьому передбачити можливість легкої та швидкої інтеграції без великих зусиль. Отже, для того щоб задовольнити вище описані вимоги, було вирішено, що кожна з підсистем буде окремою незалежною програмою, що дозволить будувати архітектуру кожної з підсистем найкращим для неї способом, робити її клієнт-серверною, виносити повністю на сторону клієнта, якщо в цьому є необхідність, та працювати максимально автономно. Для інтеграції вирішено розробити та реалізувати модуль, що буде відповідати за комунікацію з іншими підсистемами.

Для реалізації такого модуля необхідно визначити:

1) модель даних;
2) формат передачі даних;
3) транспорт для передачі даних.

Найбільш очевидним є вибір транспорту передачі даних. Оскільки різні підсистеми можуть бути написані, базуючись на різних технологіях, а також можуть бути фізично розміщені на різних комп'ютерах, альтернативи мережевій комунікації немає.

Основними транспортними протоколами для мережевої комунікації є TCP (Transmission Control Protocol) та UDP (User Datagram Protocol). Розглянемо основні відмінності між ними.

Отже UDP – це простий протокол, що базується на повідомленнях.

Основними характеристиками UDP є:

ненадійність – невідомо чи повідомлення досягло пункту призначення;

невпорядкованість – порядок отримання повідомлення необов'язково такий самий, як порядок їх відправлення;

датаграми – пакети відправляються як цілісна одиниця та мають певні межі.

Водночас TCP – протокол, орієнтований на з'єднання, яке підтверджене з обох сторін. Його характеристиками є:

надійність – протокол управляє підтвердженням, повторним пересиланням даних і певним тайм-аутом для повідомлень;

впорядкованість – порядок відправлення та отримання повідомлень однаковий;

потоковість – дані читаються, як потік байтів без певних меж, обробку потоку потрібно реалізовувати на вищому рівні.

Для програмного комплексу для інтерактивного навчання українській жестовій мові досить суттєвою проблемою є умова обмеженості пакета у протоколі UDP, оскільки обсяг даних може перевищувати ці обмеження, також важливим є надійність надходження повідомлень. Тому для реалізації модуля комунікації між підсистемами в якості транспорту передачі даних було вибрано мережевий протокол TCP.

Щодо формату передачі даних, на даний момент широко використовуються наступні формати XML (Extensible Markup Language), JSON (JavaScript Object Notation) та Google Protobuf. Ці формати, які підтримуються різними платформами, для яких є бібліотеки практично для кожної мови програмування.

Отже XML і Json є форматами, які зрозумілі людям, в той час як Protobuf є бінарним протоколом, але в той же час він має найбільшу швидкість десеріалізації та використовує найменшу кількість пам'яті. Водночас XML і Json є форматами, які дозволяють не мати чіткої схеми повідомлення, на відміну від них Protobuf вимагає чіткої схеми повідомлення.

Враховуючи високу швидкість серіалізації та десеріалізації, низьке використання пам'яті, а також реалізацію зручних для програмістів бібліотек, практично під кожну популярну мову програмування було вирішено використовувати Protobuf формат даних. Але проблему чітко детермінованої схеми даних доведеться вирішувати за допомогою моделі даних.

Для моделі даних основною вимогою є її масштабованість, тобто коли можна додати якісно нове повідомлення з новими даними, абсолютно не змінюючи програмного коду, який уже написаний, а лише додаючи новий. У такій моделі кожна з підсистем знатиме тільки ті повідомлення, що відносяться тільки до неї і не знатиме повідомлень сторонніх систем, з якими вона не комунікує. Таку модель даних можна описати за допомогою наступного proto файлу:

```
message Message {
   required int32 id;
   required int32 correlationID;
   optional bytes data;
}
```

Поле *id* є ідентифікатором формату даних внутрішнього повідомлення, тобто воно фактично задає яким чином серіалізувати/десеріалізувати поле *data*, визначене, як масив байтів, але, насправді, є уже серіалізованим protobuf повідомленням. Тобто поле *id* визначає, яким чином потрібно десеріалізовувати поле *data*. Поле *correlationID* відповідає за кореляцію запиту і відповіді, тобто система гарантує, що на запит з певним *correlationID* клієнт отримає відповідь з тим самим значенням цього поля.

Отже, за такої моделі даних при необхідності додати нове повідомлення всього лише потрібно додати парсер даних для цього конкретного *id* та обробник подій для нього ж. Водночас, якщо відокремити реалізацію обробника подій від парсера, то з'являється можливість легко реалізувати і інші типи формату передачі даних (Json, xml), – для цього потрібно буде лише реалізувати парсер під конкретний формат.

На основі вищеописаної моделі даних був проведений експеримент для порівняння цих форматів за допомогою тестової утиліти, написаної на мові програмування Java. Для цього серіалізували та десеріалізували 1000 повідомлень. Основні характеристики, які порівнювались – це швидкість серіалізації, швидкість десеріалізації, кількість витраченої пам'яті на одне повідомлення (рис. 2).

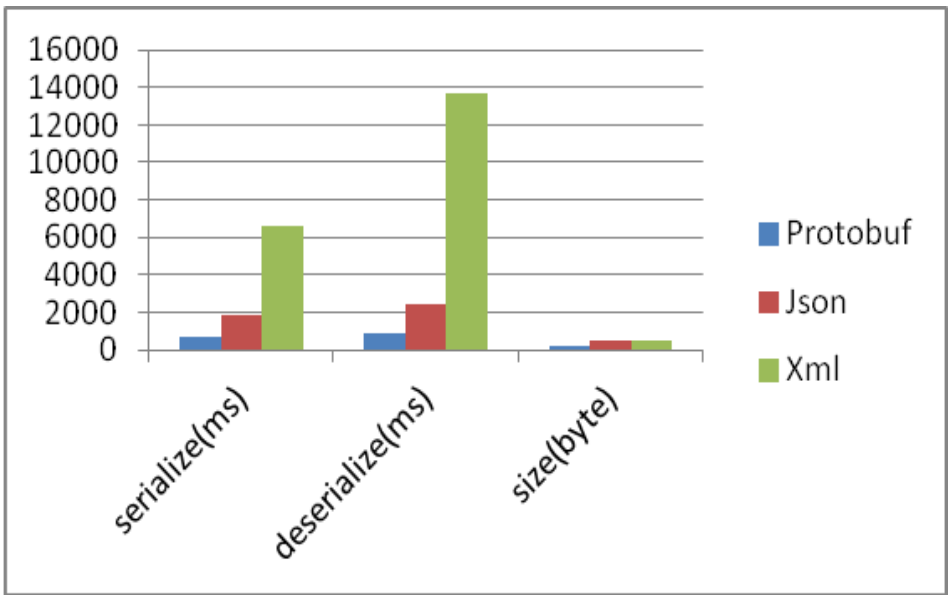

Рис. 2. Результати експерименту швидкодії форматів даних

Архітектура та деталі реалізації модулів програмної системи

Розглянемо детальніше основні вимоги, функції та структуру основної підсистеми. Функціями основної підсистеми є: генерація анімації жестів, генерація мімічних, артикуляційних та емоційних проявів на обличчі аватара, генерація звукового супроводження жесту (голосового промовляння слова, якому відповідає даний жест), синхронізація артикуляції та голосу, а також вивід зображення та звуку за допомогою графічного рушія та звукової бібліотеки. Реалізацію усіх цих функцій, окрім генерації звукового супроводження, об'єднаємо в один модуль — модуль “жестівник”. Водночас для інтеграції з іншими підсистемами повинен бути наявний модуль зв'язку. Високорівнева структура основної підсистеми показана на рис. 3.

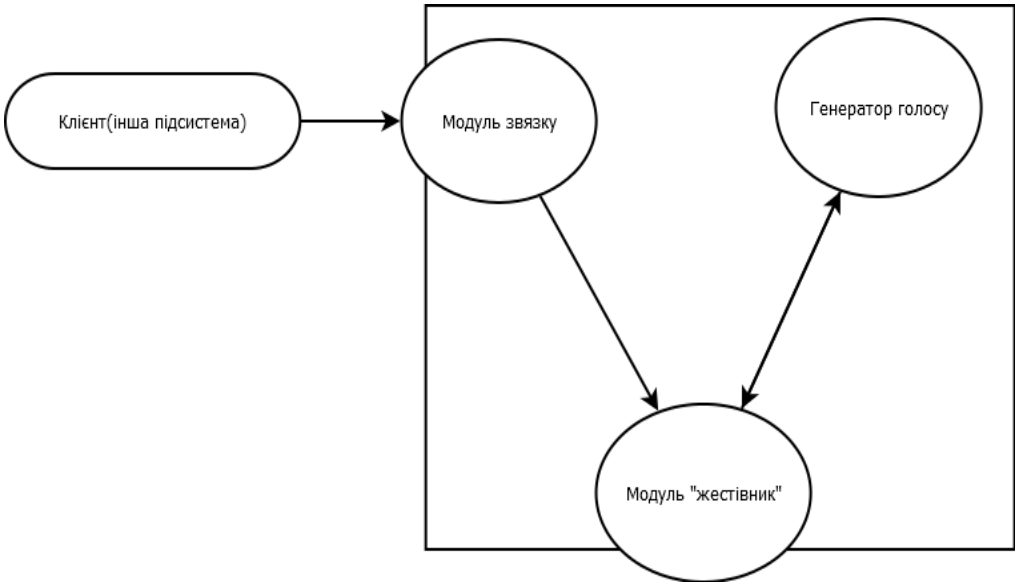

Рис. 3. Структура сервера жестової мови

Модуль “жестівник” є одним з найважливіших компонентів усієї системи, оскільки сам він забезпечує генерацію жестів та їх відображення, відповідає за реалістичну анімацію жестів, з одночасною анімацією артикуляції та різних емоційних проявів і звуковим супроводженням [5].

Основними архітектурними вимогами для такої системи є:

1) обчислення в режимі реального часу;
2) підтримка сучасних підходів і технологій тривимірної графіки для реалістичного зображення аватара;
3) кросплатформеність;
4) модульна структура;
5) організація бази жестів з великою кількістю даних.

Надзвичайно важливо, щоб система могла підтримувати обчислення в режимі реального часу, оскільки вона є досить ресурсозатратною. Для реалістичного відображення аватара, його рухів, емоційних проявів необхідно використовувати тривимірні моделі тіла людини з великою кількістю полігонів та скелетом, що складається з більш ніж 80 кісток. Також важливо, щоб тривимірний аватар виглядав максимально реалістично, був схожий на звичайну людину.

Для реалізації всього функціоналу модуль «жестівник» повинен складатись з наступних компонент:

1) контролер міміки та емоційних проявів;
2) контролер анімації рухів та жестів;
3) контролер синхронізації звуку та міміки;
4) графічний рушій;
5) звукова підсистема.

Ці компоненти формують певну послідовність перетворення вхідних даних, якій на вхід подається ідентифікатор жесту і через яку дані проходять у чіткому порядку, та результат виконання попередньої компоненти подається, як вхідні дані на наступну. Кожна з цих компонент повинна бути незалежною та мати можливість бути заміненою іншою реалізацією без зміни решти компонент. Схема архітектури та послідовність виконання показана на Рис. 4.

Рис. 4. Структура модуля «жестівник»

Модель даних для модуля подається наступними даними:

- множина вершин, нормалей та координат текстур тривимірної моделі в початковому положенні;
- множина станів обличчя для базових візем та емоційних проявів;
- множина кісток, до яких прив’язані вершини, а також вагові коефіцієнти для кожної такої прив’язки;
- множина базових жестів, що представлені за допомогою скелетної анімації.

Контролер міміки відповідає за генерацію артикуляції промовляння та мімічних артикуляційних проявів за допомогою позиційної анімації (pose animation). Для позиційної анімації використовуються множини станів облич, які зображуються у вигляді пар: номер вершини – вектор зміщення. Таке подання дозволяє зменшити потребу в пам’яті, оскільки незміщені вершини у ньому не зберігаються. На вхід контролеру подається масив з часовими значеннями, масив ідентифікаторів візем, що відповідають даному часовому значенню, та ідентифікатор віземи емоційного прояву.

Контролер анімації відповідає за генерацію анімації жесту за допомогою скелетної анімації. Моделлю даних для контролера анімації є множина кісток та прив’язка вершин. Анімація жесту подається, як кути повороту кожної кістки відносно батьківської кістки в певний момент часу. Завдяки тому, що кожна вершина прив’язана з певними ваговими коефіцієнтами до деякої кількості кісток, відпадає необхідність зберігати положення кожної точки окремо, достатньо лише зберігати інформацію про рух кісток, а рух вершин обчислюється відносно них.

На Рис. 5 показана діаграма послідовностей модуля.

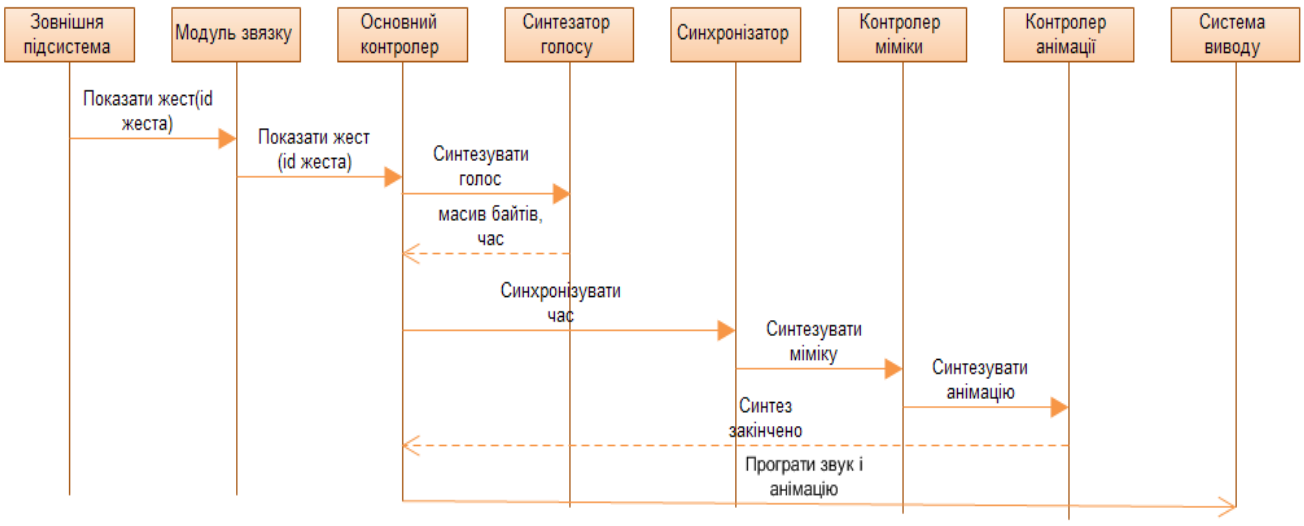

Рис. 5. Діаграма послідовностей

Важливим аспектом роботи контролера є механізм для управління анімаціями та переходами між ними. Було розроблено наступну модель, що підтримує три типи переходів між анімаціями:

1) моментальне завершення першої анімації та старт другої;
2) плавний перехід від першої анімації до другої;
3) очікування закінчення першої анімації і після цього старт другої.

Оскільки перший та третій пункти є досить простими, зупинимось детальніше на другому пункті.

Для забезпечення плавного переходу між двома анімаціями застосовувався такий алгоритм. Спочатку була поставлена умова, що наступна анімація буде стартувати за 30% часу до закінчення стартової анімації. Після цього для кожної анімації встановлювався свій ваговий коефіцієнт, що визначався за наступною формулою: $weight_1$ = timeLeft/(0.3*duration), $weight_2 = 1 - weight_1$, де $weight_1$ – ваговий коефіцієнт першої анімації, timeLeft – час, що залишився до кінця анімації, duration – довжина анімації.

Ці завдання можливо вирішити шляхом виконання деяких частин програми на графічному процесорі за допомогою програмованих шейдерів. Шейдер (англ. Shader) – це програма для одного зі ступенів графічного конвеєра, що використовується в тривимірній графіці для визначення остаточних параметрів об'єкта чи зображення. Вона може містити у собі довільної складності описи поглинання та розсіювання світла, накладення текстури, віддзеркалення і заломлення, затінення, зміщення поверхні і ефекти пост-обробки. Програмовані шейдери гнучкі та ефективні. Складні на вигляд поверхні можуть бути візуалізовані за допомогою простих геометричних форм.

Характеристика основних графічних рушіїв

Важливим етапом у розробці системи є аналіз наявних технологій для відображення тривимірної графіки, так званих графічних рушіїв, основним завданням яких є візуалізація тривимірної графіки. Основними критеріями для таких рушіїв є відкритий доступ, підтримка програмних шейдерів на мовах GLSL, HLSL, CG, підтримка DirectX та OpenGL та можливість змінити графічну підсистему без зміни програмного коду, висока оптимізація, підтримка експорту тривимірних моделей із спеціалізованих програм. Також для реалістичного відображення тривимірної моделі тіла людини важлива підтримка різних типів освітлення, можливість по-різному відбивати світло від різних поверхонь, можливість одночасного відображення кількох анімацій.

Наведеним критеріям повністю відповідають три графічних рушії – Irrlight, Ogre, та Unity. Окрім того, вони є кросплатформеними, підтримують як OpenGL так і DirectX графічні бібліотеки, що в майбутньому може дозволити без проблем портувати систему на інші платформи. Основними перевагами Irrlight є його проста і прозора архітектура, невеликий, але достатній основний функціонал, сумісність з деякими аудіо бібліотеками [6]. Ogre має більш складну архітектуру, потужний функціонал для управління матеріалами, підтримку позиційної анімації (pose animation), систему генерації шейдерів у реальному часі на основі скриптів матеріалів. Також великою перевагою Ogre є його гнучкість та можливість інтеграції з фізичними рушіями та аудіобібліотеками. Обидві системи мають досить розвинену спільноту, яка надає підтримку в разі проблем з їх використанням [7]. Ogre також підтримує мобільну систему iOS, але, на жаль, не підтримує мобільної системи Android без додаткових модифікацій. Unity ж, у свою чергу, є не графічним, а ігровим рушієм, тобто він має вбудовану підтримку звуку, фізики, мережі та інших речей, необхідних для тривимірних ігор. Водночас він є простим у користуванні та підтримує мобільні системи, такі як Android та iOS, що, в час стрімкого зросту мобільних технологій, є дуже важливим. Основним недоліком Unity є те, що це система з закритим кодом і безкоштовною є лише урізана версія її для персональних комп'ютерів, для мобільних систем Unity є платною[8]. Описані характеристики наведені в таблиці.

Таблиця. Характеристики кросплатформених рушіїв

	Ogre	Irrlight	Unity
Підтримка шейдерів	+	+	+
Підтримка DirectX/ Opengl	+	+	+
Кросплатформеність на ПК	+	+	+
Підтримка мобільних пристроїв	Тільки iOS	-	+
Підтримка звуку	Сторонні бібліотеки	Сторонні бібліотеки	Вбудована
Складність рушія	Середня	Легка	Складна
Бесплатність/Відкритість	+	+	-

Програмна реалізація

Використовуючи Ogre 3D, було реалізовані компоненти 2, 4 та 5 модуля «жестівник» та побудовано програмне забезпечення для відображення тривимірної моделі з простою скелетною анімацією, керуванням камерою та освітленням. Для його побудови використовувалась тривимірна модель з 26337 вершинами, 53992 полігонами та 70 кістками у скелеті.

У системі було реалізовано 3 програмних шейдери:

1) шейдер для підтримки карт нормалей, що дозволить досягти більш високої деталізації аватара, не збільшуючи полігональність вхідного меша;

2) шейдер для підтримки карт відображень для більш реалістичного відтворення і відбиття освітлення;

3) шейдер для підтримки обчислення скелетної анімації на графічному процесорі, що дозволив прискорити виконання цієї ресурсомісткої операції. Тестування проводилось на відеопроцесорі NVIDIA GeForce G102M, при цьому FPS (frames per second) кількість кадрів відтворення зросла від 30 до 120 за рахунок цього шейдера.

Робоче вікно «жестівника» показане на Рис. 6.

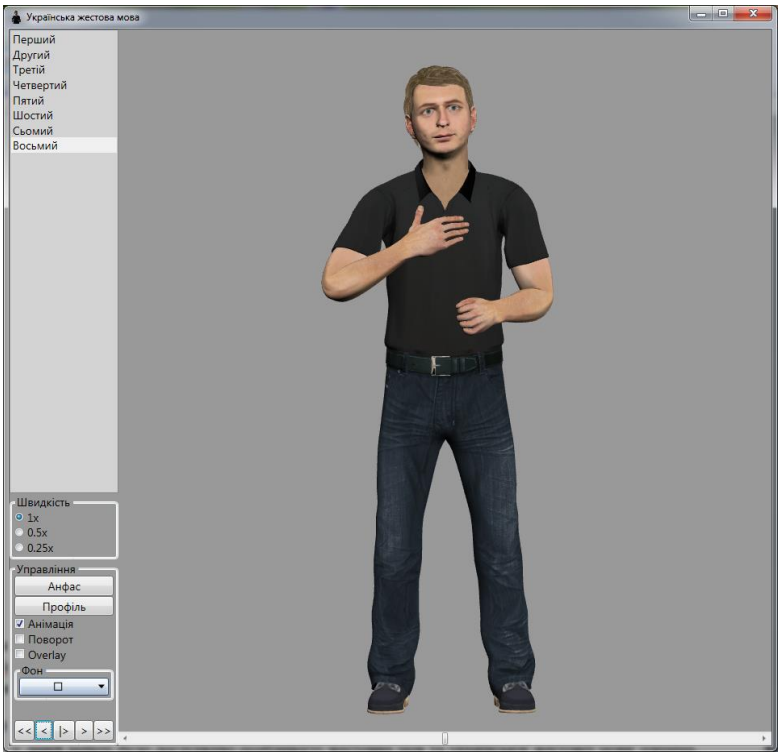

Рис. 6. Робоче вікно тестового програмного забезпечення

Також реалізовано модуль зв'язку, який реалізує протокол передачі даних між підсистемами, на мові C++ і Java. За допомогою цього модулю також реалізовано віддалений клієнт на платформі Android, який дозволяє керувати основною підсистемою.

Висновки

Досліджено жестові мови та зокрема, українську жестову мову, виділено її основні особливості. Проаналізовані основні вимоги до систем комп'ютеризації навчання жестових мов, розроблено базову високорівневу архітектуру для реалізації підсистеми синтезу жестової мови. Розроблено протокол передачі даних між різними підсистемами що можуть входити у програмний комплекс. Розглянуто моделі даних для відображення основних типів жестів та міміки, моделі генерації та відтворення анімації, а також основні інструменти для реалізації програмного забезпечення – сучасні бібліотеки для роботи з тривимірною графікою та звуком.

Реалізовані два основні модулі – генератор анімації та графічний рушій на основі Ogre та певний користувацький інтерфейс для цих модулів. Дане програмне забезпечення може бути використано для реалізації віртуального вчителя та перекладача жестової мови, для створення мультимедійних тематичних словників жестових одиниць, інтерактивних підручників та інтелектуальних інтерфейсів. Розробка експериментальної системи навчання жестовому мовленню продемонструвала ефективність і перспективність запропонованого підходу.

Подальші дослідження будуть спрямовані на програмну реалізацію модуля для анімації міміки та емоційних проявів і її інтеграцію в наявну систему, створення програмного шейдера для проведення обчислення

цієї анімації на графічному процесорі. Водночас для реалістичного відображення очей необхідно додати до системи шейдер, що відповідатиме за відображення зовнішнього світу в очах аватара.

1. *Прозорова Е.В.* Российский жестовый язык как предмет лингвистического исследования // Вопросы языкознания. – М., 2007. – № 1. – С. 44–61.
2. *Боресков А.* Разработка и отладка шейдеров – С.Петербург: БХВ, 2006. – 496 с.
3. *Smith R., Morrissey S., Somers H.* HCI for the Deaf community: Developing human-like avatars for sign language synthesis // Proccedings of the 4th Irish Human Computer Interaction Conference, 2-3 September 2010, Dublin. – P. 129–136.
4. *Hurdich J.* Utilizing Lifelike, 3D Animated SigningAvatar Characters for the Instruction of K-12 Deaf Learners. // Exploring Instructional and Access Technologies. Abstracts of International Symposium. – 23-25 June 2008, New York: RIT – P. 40–41.
5. *Кривонос Ю.Г., Крак Ю.В., Бармак О.В. та ін.* Інформаційна технологія для моделювання української мови жестів // Штучний інтелект. – 2009. – № 3. – С. 186–197.
6. *Електронний* ресурс Irrlicht. Режим доступу: http://irrlicht.sourceforge.net/docu.
7. *Електроний* ресурс Ogre. Режим доступу: http://ogre3d.org/tikiwiki/.
8. *Електронний ресурс* Unity. Режим доступу: http://unity3d.com/learn/.