Krassimir Markov, Vitalii Velychko, Oleksy Voloshin
(editors)

Information Models of Knowledge

I T H E A®
K I E V – S O F I A
2010

Krassimir Markov, Vitalii Velychko, Oleksy Voloshin (ed.)

Information Models of Knowledge

ITHEA®

Kiev, Ukraine – Sofia, Bulgaria, 2010

ISBN 978-954-16-0048-1

First edition

Recommended for publication by The Scientific Concil of the Institute of Information Theories and Applications FOI ITHEA

ITHEA IBS ISC: 19.

This book maintains articles on actual problems of research and application of information technologies, especially the new approaches, models, algorithms and methods fot information modeling of knowledge in: Intelligence metasynthesis and knowledge processing in intelligent systems; Formalisms and methods of knowledge representation; Connectionism and neural nets; System analysis and sintesis; Modelling of the complex artificial systems; Image Processing and Computer Vision; Computer virtual reality; Virtual laboratories for computer-aided design; Decision support systems; Information models of knowledge of and for education; Open social info-educational platforms; Web-based educational information systems; Semantic Web Technologies; Mathematical foundations for information modeling of knowledge; Discrete mathematics; Mathematical methods for research of complex systems.

It is represented that book articles will be interesting for experts in the field of information technologies as well as for practical users.

General Sponsor: Consortium FOI Bulgaria (www.foibg.com).

Printed in Ukraine

ISBN 978-954-16-0048-1

C\o Jusautor, Sofia, 2010

ИССЛЕДОВАНИЕ ИНФОРМАЦИОННЫХ ПРОЦЕССОВ ДЛЯ ЭФФЕКТИВНОГО ВОСПРОИЗВЕДЕНИЯ ДАКТИЛЬНОГО ЖЕСТОВОГО ЯЗЫКА

Юрий Крак, Юрий Кривонос, Богдан Троценко

Abstract: *Complex information technology for visualization communications gesture and mimicry is created.*

Keywords*: modelling, sign language, computer system.*

ACM Classification Keywords*: I.2.8 Problem Solving, And Search H.1.1 Systems and Information*

Введение и постановка задачи

С созданием независимого украинского государства и признанием на государственном уровне жестового языка как средства межличностного общения и средства обучения глухих лиц возникла насущная необходимость в изучении национального жестового языка и создании современных словарей и средств обучения этому языку.

Языкам свойственны различные диалекты, жестовый язык не является исключением. Жестовый язык, как и любой другой, имеет различные диалекты. Жестовый язык редко используется на телеканалах, к тому же в отличии от устной речи не имеет распространенного письменного эквивалента, как результат, он имеет диалектические различия даже в пределах области. Наличие некоторого «эталона» жестовых единиц (дактилем) в записи позволит облегчить проблему диалектов.

Сейчас жестовый язык можно изучать с помощью учителя, книги или видеозаписи. Но каждый учитель имеет «свой» дактильный жестовый язык, его аудитория обучения ограничена, учитель как «медиа носитель» не копируется, нельзя повторить / увидеть жест вне учебы. При обучении с помощью книги [1]: двумерное изображение не передает динамики и деталей жестовых единиц, книга не является интерактивным медиа-носителем. Даже более современный способ представления жеста с помощью видеоизображения [2] имеет свои недостатки: большой объем данных, невозможность рассмотреть жест с разных сторон, кроме того, из отдельно записанных жестов нельзя составить предложение, которое бы выглядело естественно, так как видео не является интерактивным медиа-носителем.

Современное развитие компьютерной техники делает возможным использование более эффективного подхода к обучению жестовому языку, в частности дактильному жестовому языку, который рассматривается в данной работе. Использование компьютерных технологий будет способствовать уменьшению количества диалектов, благодаря распространению «эталонных» жестов. Более того, легкость копирования и использования программ, а также возможность работы по сети Интернет [3], которая позволяет отображать дактилемы на компьютерах различной мощности, будет способствовать распространению знаний и навыков владения дактильным жестовым языком.

Постановка задачи: предложить алгоритм и реализацию оптимального по времени расчета состояния пространственной модели при ее высокой размерности, наличия скелета, близкого к скелету руки человека, и обычных мультимедийных требований к аппаратной части компьютера, рассмотреть алгоритмы расчета поверхности модели с целью выявления наиболее эффективного, предложить метод изображения дактилированного слова на основе отдельных дактилем, для проведения вычислений задействовать все ядра процессора. Под оптимальным по времени расчетом подразумевается время, которое соответствует частоте смены 30 и более кадров в секунду.

Математическая модель

Для решения задачи используем скелетную модель руки. Скелет руки математически изобразим в виде дерева (V, E), где V – множество вершин, E – множество соединений. Каждая вершина соответствует некоторому сочленению костей (звеньям скелета). Каждый элемент отвечает за связь между звеньями,

при этом звено e_i назовем старшим по отношению к звену e_j, а звено e_j – *дочерним* по отношению к e_i. Каждое звено может иметь не более одного старшего и произвольное количество дочерних звеньев. Звено, не имеющее соответствующего ему старшего, назовем *основным*.

Через $P(e)$ будем обозначать старшее по отношению к e звено, если такое существует, и Ø в противном случае.

Для каждого звена поставим в соответствие упорядоченную пару $h(e) = (x, \theta)$, где $x \in R^3$ положение сочленений звена со старшим в системе координат, связанн с $P(e)$. Параметр θ задает порядок применения углов поворота для преобразования системы координат, связанной со звеном, $h(V)$ задает исходное положение руки для математической модели (V, E).

Для каждого звена $e \in V$ поставим в соответствие вектор $r(e) \in R^3$, состоящий из углов поворота вокруг осей *OX*, *OY*, *OZ*. Таким образом, совокупность $r(V)$ задает положение руки, данную совокупность также будем называть параметрами модели. При наличии отношения порядка между элементами множества вершин (*V*), например, по индексу, $r(V)$ можно единственным образом представить в виде матрицы размерности $3 \times |V|$, где *i*-й столбец является вектором $r(e_i), e_i \le e_{i+1}$. Данную матрицу будем называть *матрицей параметров модели.*

Основной операцией в данной математической модели является получение положений точек соединений звеньев на основе параметров модели, а также получение матриц перехода систем координат относительно системы координат, которая соответствует основному звену.

Матрицы перехода систем координат определяются формулой 1:

$$M(e) = \begin{cases} M(P(e)) \cdot f(\theta, M_X^{r(e)}, M_Y^{r(e)}, M_Z^{r(e)}), P(e) \neq 0 \\ I, \ P(e) = 0 \end{cases}, \quad (1)$$

где $M_X^{r(e)}$ – матрица поворота системы координат вокруг оси *OX* на угол, который записан в первой строке вектора $r(e)$, $M_Y^{r(e)}$ матрица поворота системы координат вокруг оси *OY* на угол, который записан во второй строке вектора $r(e)$, $M_Z^{r(e)}$ – матрица поворота системы координат вокруг оси *OZ* на угол, записанный в третьей строке вектора $r(e)$, I – единичная матрица размерности 3x3, $f(\theta, M_1, M_2, M_3)$ – произведение матриц в порядке, определенном θ.

Положение звена в глобальной системе координат определяется по формуле 2:

$$p'(e) = x(e) \cdot M(e), \quad (2)$$

где $p'(e)$ положение звена в глобальной системе координат для состояния системы M

На основе данной математической модели построим информационную модель руки, состоящую из математической модели, матрицы точек в пространстве для модели в начальном положении, текстуры поверхности руки, матрицы текстурных координат, матрицы нормалей.

Для получения матриц параметров модели была использована технология Motion Capture (захват движения). Данная технология предусматривает использование оцифрованных данных о движении, записанных с человека-носителя языка жестов. Для этого существует специальная перчатка, имеющая яркие зоны в областях, близких к соединению костей. На несколько видеокамер записывается «проговаривание» всей азбуки дактильного жестового языка. С помощью специального программного обеспечения, Motion Builder, получается динамика изменения положений в пространстве выделенных зон, а на основе этих данных - информация об изменении углов.

Таким образом, за счет использования данных такой природы, нет необходимости во введении ограничений на множество допустимых углов в математической модели. Данные, полученные с помощью

технологии Motion Capture, более реалистично воспроизводят оригинальную динамику «проговаривания» дактилем, чем данные, полученные с помощью дизайнерского программного обеспечения.

Алгоритмы расчета точек поверхности руки (скининг)

Поверхность модели руки представляет собой множество точек, положение которых зависит от положения звеньев. Каждой точке сопоставлено множество $\{(e_i, w_i)\}$, где e_i звено, влияющее на точку, $w_i \in [0,1]$ - коэффициент зависимости, $\Sigma_i w_i = 1$. Соответственно, положение точки в пространстве определяется формулой $\Sigma_i p'(e_i) \cdot w_i$. Алгоритм расчета данной формулы для всех точек информационной модели и подготовка соответствующих структур данных называются *скинингом* [4].

Для эффективной визуализаций модели на практике было предложено несколько алгоритмов скининга с целью дальнейшей апробации. Все алгоритмы используют массив входных данных точек в исходном положении, записывают результат в массив точек для конечного положения. Типично, что массив результата инициализируется точками с координатами (0,0,0), в процессе работы алгоритма в соответствующие элементы добавляются составные части; таким образом, лишь в конце работы алгоритма результирующий массив содержит правильные положения точек.

Последовательный алгоритм скининга использует представления зависимостей в виде списков (звено; массив коэффициентов зависимостей, каждый элемент которого соответствует элементу из массива координат точек в исходном положении). Он описывается следующим псевдо-кодом:

```
для каждого звена i:
    для каждой вершины j:
        получить координату вершины j в исходном положении,
        умножить на матрицу перехода в систему координат,
        связанную со звеном i,
        умножить на коэффициент зависимости,
        добавить координату к элементу массива результатов
```

Алгоритм рассчитан на то, что локальные части массивов координат начального положения, коэффициентов зависимостей, конечных координат на практике находятся в кэш-памяти процессора, что позволит быструю обработку данных с относительно малым количеством операций обмена данными с оперативной памятью.

Последовательный алгоритм скининга с *минимаксной оптимизацией* является улучшенным вариантом последовательного алгоритма скининга, так как на этапе предварительной обработки для каждого звена вычисляются минимальный и максимальный индексы вершин, на которые это звено влияет. Это ориентировано на то, что зависимые точки занимают в массиве координат примерно соседние индексы, что позволит уменьшить количество операций с вершинами, коэффициент зависимостей которых 0.

Сегментационный алгоритм скининга является улучшенным вариантом последовательного алгоритма, поскольку он использует представление в виде списков (звено; список пар (начальный индекс, массив коэффициентов зависимостей)). Представление выбирается таким образом, что длина массива коэффициентов зависимостей не превышает заданную величину, например, 256. Каждый элемент с индексом *i* из массива коэффициентов зависимостей соответствует элементу *i* + *начальный индекс*. Алгоритм рассчитан на оптимальное использование кэш-памяти процессора в практическом применении. Он описывается следующим псевдо-кодом:

```
для каждого звена i:
    для каждой пары из списка
    (начальный индекс, массив коэффициентов зависимостей),
    что соответствует звену i:
        для каждого коэффициента зависимостей из массива
        с индексом k:
          получить координату вершины в начальном положении
```

с индексом (k + начальный индекс),
умножить на матрицу перехода в систему координат, связанную со звеном i,
умножить на коэффициент зависимости,
добавить координату к элементу массива результатов
с индексом (k + начальный индекс),

Прямой алгоритм использует представления зависимостей в виде списков пар (звено; массив пар (индекс вершины; коэффициент зависимости)). Он использует рекурсивную процедуру обхода вершины, что описывается следующим псевдо-кодом. Сам алгоритм заключается в выполнении процедуры обхода старшей вершины.

```
процедура Обход вершины (звено i)
    для каждой пары из массива, соответствующего звену i:
        получить координату вершины
        в начальном положении по индексу,
        умножить на матрицу перехода в систему координат,
        связанную звеном i,
        умножить на коэффициент зависимости,
        добавить координату к элементу массива результатов

    для каждого звена j, которое является дочерним к звену i:
        выполнить обход вершины j
```

Среди приведенных алгоритмов, на практике прямой алгоритм выполняет минимальное количество операций. В случае, когда сумма объемов памяти, которую занимают точки модели, массив координат начальных и конечных положений, дополнительные структуры данных, меньше, чем объем кэш-памяти процессора, данный алгоритм показывает наивысшую эффективность. В противном случае, рекомендуется использовать сегментационный алгоритм.

Данные алгоритмы были имплементированы, для них были проведены сравнительные тесты на эффективность. Под эффективностью алгоритма подразумевается скорость расчета отдельного кадра, время на подготовку структур данных игнорируется. Тестирование проводилось на скелетной модели, состоящей из 84 460 точек на компьютере с процессором Intel Core2 Duo E8200 (2,66 ГГц), 2 Гб оперативной памяти. Расчет проводился в один поток. Результаты тестирования приведены на рисунке 1.

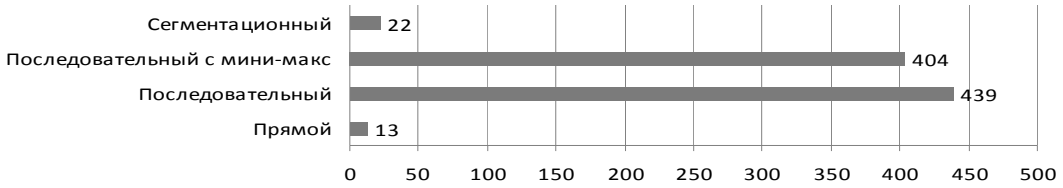

Рис. 1. Результаты тестирования алгоритмов скининга

Алгоритм дактилирования слова как конкатенация отдельных дактилем

Важным преимуществом предложенной системы является возможность дактилировать произвольные слова, состоящие из отдельных дактилем («букв»). При наличии информации об отдельных дактилемах важно иметь способ их визуальной конкатенации. С использованием модели руки как основы представления графической информации, это становится алгоритмически возможным, в отличие от данных в формате видео.

Была создана база данных, содержащая последовательности состояний модели для каждой буквы. Каждая такая последовательность начинается и заканчивается в некотором «нейтральном» состоянии. Для правильной конкатенации необходимым является подход для определения предельных

состояний каждой дактилемы, где заканчивается информация, которая важна для передачи содержания дактилемы, а также алгоритм создания промежуточных состояний для перехода между дактилемами.

Учитывая природу полученных данных, предельные состояния были установлены экспериментально. Генерирование промежуточных состояний может быть либо алгоритмическим (на основе состояний, которые конкатенируются) или предопределенным. Алгоритмический подход является более удобным, поскольку уменьшает количество необходимой информации, которую нужно подготовить дизайнеру. Для большинства пар дактилем можно алгоритмически сгенерировать промежуточные слайды, например, с помощью линейной аппроксимации на основе предельных состояний смежных букв.

Есть последовательности, к которым применить алгоритмический подход невозможно, например, при дактилировании «Н» - «Г» [1] нужно сначала согнуть пальцы, а потом вращать ладонь. Для этого случая предполагается использование отдельной последовательности состояний перехода.

Распараллеливание вычислений в задаче подготовки кадра

Большинство выпускаемых современных процессоров оснащено двумя или более ядрами. Дальнейшее развитие ориентировано именно на увеличение количества ядер, чем на наращивание мощности отдельного ядра [5].

Для подзадачи подготовки кадра в задаче отображения трехмерной анимации дактильной азбуки жестов был рассмотрен алгоритм распараллеливания вычислений, поскольку по своей сути подготовка одного кадра является функцией от модели руки и параметров модели, т.е. не зависит от результата вычислений других кадров и от момента вычисления. Такой подход имеет место, когда вычисление нового кадра является ресурсоемкой задачей, например, проведение скининга на процессоре. Для этого был разработан алгоритм, состоящий из $2 + n$ потоков, где n количество ядер процессора: основной поток, поток анимации, n потоков вычисления кадров. Используется тот факт, что вычисления различных кадров является равносильной задачей в терминах процессорного времени независимо от параметров модели. Поэтому вычисление k кадров можно разбить на n потоков максимально равномерно таким способом: каждый i-ый поток обрабатывает кадры с тем номером, остаток от деления которого на n равен i.

Основной поток (поток интерфейса) инициирует создание n потоков вычислений и потока анимации. Поток анимации показывает с заданным промежутком подготовленные кадры, освобождает память после их показа. Поскольку на момент начала анимации должна быть уверенность, что каждый кадр будет готов перед его показом, то поток информации запускается приостановленным, а его запуск делегируется на один из потоков вычисления кадров, в частности, на поток с индексом 0. В дальнейшем, этот поток будем называть *ведущим*.

Каждый поток вычисления кадров в начале получает остаток от деления и начинает цикл вычислений. Также проводится обновление счетчика вычисленных кадров.

Ведущий поток также фиксирует момент времени начала вычислений. После того как было вычислено 5 кадров, он аппроксимирует момент времени окончания вычислений всех кадров. Поскольку длительность анимации известна изначально, то возможно определить, будет ли подготовка кадров выполнена вовремя для непрерывного отображения анимации. Когда аппроксимированного времени до конца завершения вычислений становится меньше продолжительности анимации, ведущий поток запускает поток анимации и прекращает прогнозирование завершения.

Задача распределения потоков на каждое ядро выполняется операционной системой. Но часто бывают ситуации, когда ядра заняты посторонними процессами. Это привело бы к тому, что потоки предобработки получили бы различные кванты процессорного времени, а как следствие, кадры вычислялись неравномерно.

Например, возможна была бы такая ситуация (в примере используется двухъядерный процессор):

№ кадра	0	1	2	3	4	5	6	7	...
Состояние	+	+	+	-	+	-	+	-	...

Где состояние «+» означает, что кадр подготовлен, «-» - не подготовлен.

Для решения данной проблемы был применен подход с использованием семафоров [6,7]: каждый поток проассоциирован одним семафором, который инициализирован единицей. В начале вычисления кадра поток уменьшает свой семафор на единицу, после завершения кадра он увеличивает семафор следующего потока на единицу (последний поток увеличивает семафор нулевого потока). Таким образом, вычисление кадров одного потока не опережает остальные более чем на единицу.

По особенностям своего назначения, семафор не может принимать значения меньше нуля. По своему назначению, значение семафора не может изменяться меньше нуля. Семафор является объектом ядра операционной системы, который отвечает за совместный доступ нескольких потоков к общим ресурсам. У этого объекта две операции – увеличить (на единицу), уменьшить (на единицу). Если значение семафора – ноль, а поток выполняет операцию "уменьшить", выполнение его приостанавливается операционной системой, а значит, становится доступным дополнительное процессорное время. Операционная система ведет учет всех потоков, которые были приостановлены вследствие попытки уменьшить семафор. Когда некоторый другой поток делает операцию "увеличить", то другой поток из списка приостановленных продолжает выполнение.

На практике, такой подход означает следующее: каждый поток анимации в начале работы алгоритма "может" вычислить 1 кадр, за что отвечает значение его семафора. После вычисления своего кадра, он позволит вычисления другому потоку и будет приостановленным, пока поток не увеличит соответствующий семафор. Таким образом, ядра процессора максимально загружены, а кадры вычисляются постепенно.

Использование технологии внутрипроцессного обмена асинхронными сообщениями

Программа дактилирования украинского жестового языка, как и всякая другая, использует API операционной системы обращения к жесткому диску и ресурсам сети Интернет. Данные функции являютсяблокирующие, иными словами, вызывающий поток приостанавливается до получения результата.

Стандартный подход к созданию приложений для современных операционных систем (Windows, Linux, Mac OS) предполагает модель событий, как основную концепцию. Различные действия пользователя, действия других программ, либо изменения в системе передаются в программу посредством событий. Для их обработки каждая программа имеет поток, который извлекает события из очереди и вызывает соответствующие методы (обработчики) в программе. Поскольку большинство событий являются извещениями, которые относятся к пользовательскому интерфейсу, данный поток принято именовать потоком пользовательского интерфейса.

В большинстве случаев удается предвидеть среднюю продолжительность обработки API обращения к жесткому диску и выделить отдельные потоки для длительных операций. Исключения составляют те случаи, когда относительно легкие операции ввода-вывода блокируют поток пользовательского интерфейса в результате, к примеру, неготовности жесткого диска. В таких случаях программа не отвечает на другие действия пользователя и выглядит как «зависшая». К сожалению, на практике трудно использовать отдельные потоки для каждой подзадачи, которая требует ввода-вывода.

Для решения этой проблемы предлагается технология обмена сообщениями между разными потоками внутри единого процесса.

Технология предлагает использование шины обмена данными, на которой регистрируются обработчики сообщений. Каждый обработчик имеет свой идентификатор, один или несколько потоков для обработки. Каждое сообщение содержит идентификатор соответственного обработчика, набор параметров, переменные результата, объект ядра операционной системы, типа «событие», для извещения об окончании завершения обработки данного сообщения.

Предлагается реализация данной технологии в среде .Net. Ее диаграмма классов представлена на рисунках 2, 3.

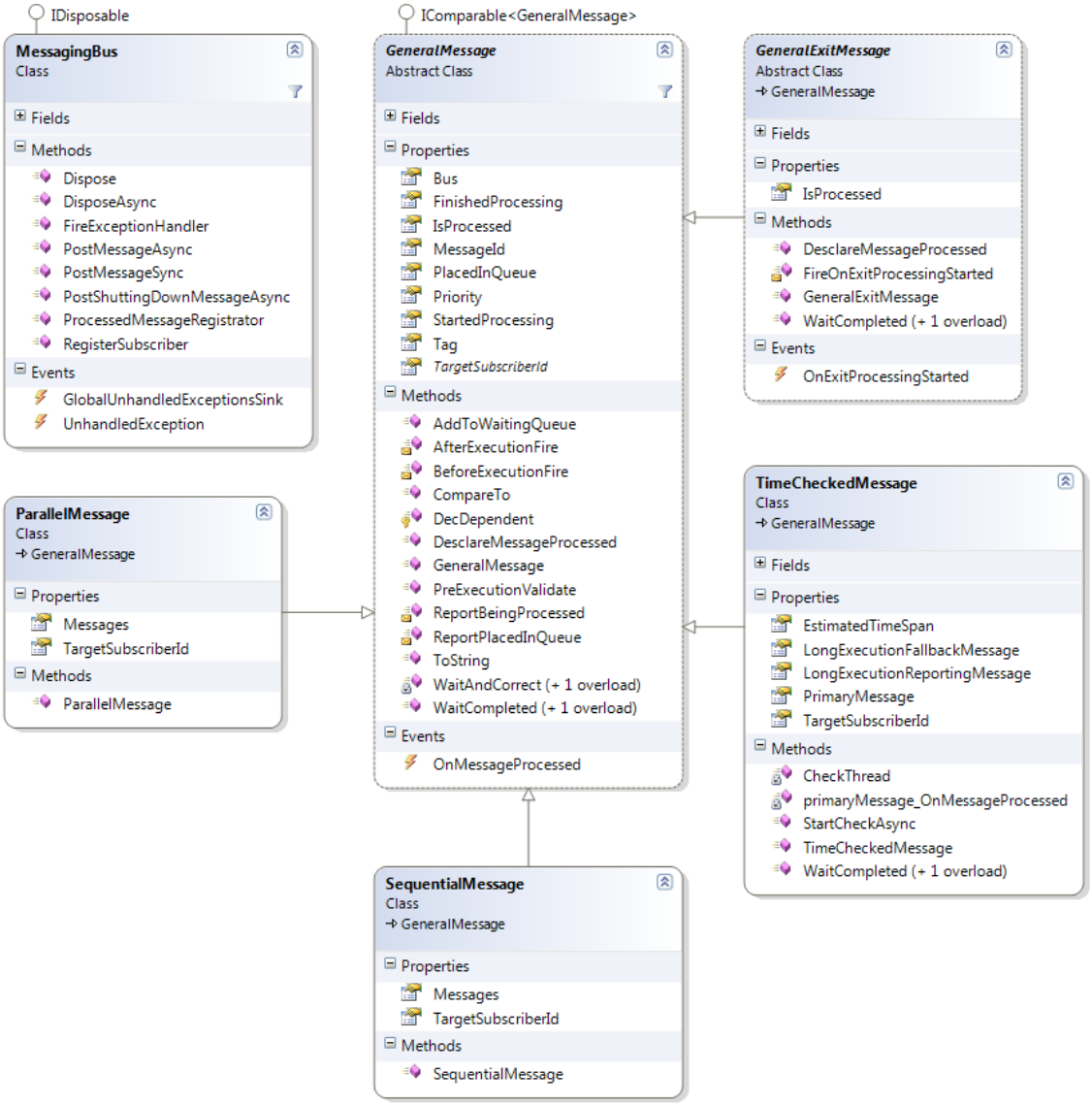

Рис. 2. Диаграммы классов для шины и основных сообщений

Основной класс, MessagingBus, описывает шину. С помощью метода RegisterSubscriber на шине регистрируются обработчики – классы, которые наследуют GeneralSubscriber. Метод PostMessageAsync передает сообщение обработчику и возвращает управление вызывающему потоку. Дополняющий его метод PostMessageSync дополнительно блокирует вызывающий поток до окончания обработки сообщения.

Абстрактный класс GeneralMessage представляет обобщенное сообщение. В частности, он имеет свойства TargetSubscriberId – идентификатор обработчика, Priority – уровень приоритета; метод AddToWaitingQueue регистрирует другие сообщения, которые влияют на результат данного, метод WaitCompleted блокирует вызывающий поток до окончания обработки сообщения; событие OnMessageProcessed уведомляет об окончании обработки сообщения.

Отдельно выделен класс GeneralExitMessage, который предназначен как базовый класс для сообщения, извещающего о завершении работы программы. Он должен быть передан на шину с помощью метода PostShuttingDownMessageAsync, который передаст это сообщение обработчику только тогда, когда на шине больше не будет других сообщений в очереди, либо в состоянии обработки.

Класс SequentialMessage принимает в качестве параметра массив других сообщений для поочередной обработки. Подобный ему класс ParallelMessage также принимает в качестве параметра массив других сообщений, но обработка его заключается в разовой передаче всех сообщений на шину и ожидания окончания их обработки.

Класс TimeCheckedMessage позволяет контролировать время обработки сообщения (PrimaryMessage); если время превышает заданный интервал, то на шину также передается LongExecutionReportingMessage. В случае передачи последнего, также будет передано сообщение LongExecutionFallbackMessage по завершению обработки основного сообщения.

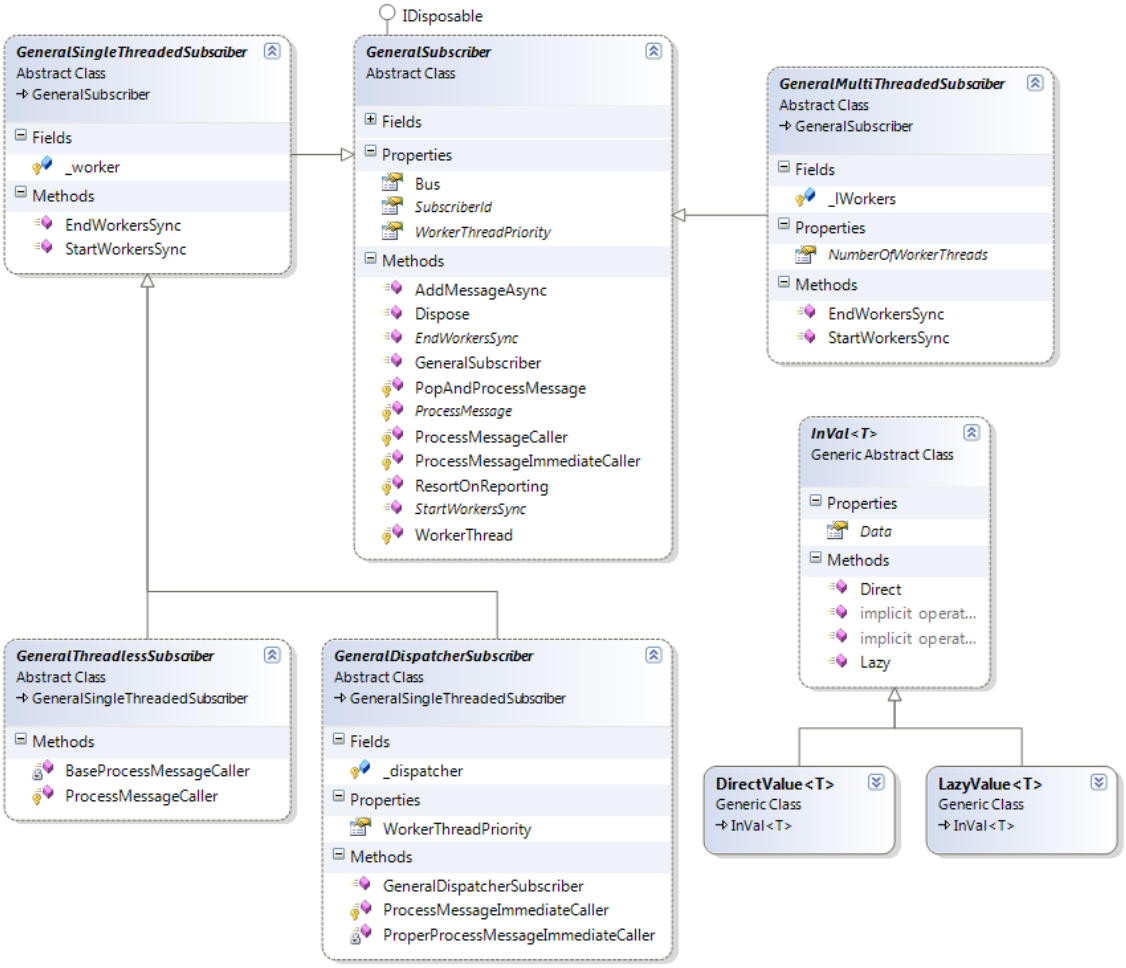

Рис. 3. Диаграммы классов для обработчиков сообщений и входных данных

По своей природе, программирование с использованием сообщений можно назвать отложенным вызовом функций. Поэтому, возможно два сценария: расчет входных параметров происходит 1) в момент организации сообщения, 2) перед началом обработки. Реализация данных сценариев произведена с помощью параметризированных классов InVal<T>, DirectValue<T>, LazyValue<T>. Конструктор DirectValue получает непосредственно значение, конструктор LazyValue получает делегат на функцию, которая вычислит входной параметр непосредственно перед началом обработки.

Класс GeneralSubscriber описывает общие свойства и поведение обработчика, реализует методы для работы с очередью сообщений. Класс GeneralSingleThreadedSubscriber позволяет осуществлять работу в один поток, GeneralMultiThreadedSubscriber – в несколько; число потоков регулируется свойством NumberOfWorkerThreads. Класс GeneralDispatcherSubscriber выполняет работу в потоке пользовательского интерфейса.

Использование данной технологии позволяет удобно описывать рабочий цикл программы. Пример использования приведен в листинге 1.

```
Bus.PostMessageAsync(new SequentialMessage(
    new ParallelMessage(
        new SequentialMessage(
            new ShowStatusMessage() { Status = "Загрузка модели..."
},
            new ComputingLoadModelMessage()
            {
                Path = "FinalData",
                ModelConfiguration = ModelConfiguration
            },
            new ShowStatusMessage() { Status = "Модель загружена." }
        ),
        new ParallelMessage(LoadCharacterMessages.ToArray())
    ),
    new ConfigureModelMessage(),
    new PositionHandMessage(),
    new ShowStatusMessage() { Status = "Инициализация завершена." },
    new EnableUsageMessage()
));
```

Листинг 1. Пример использования технологии передачи асинхронных сообщений

Использование технологии передачи данных через сеть Интернет

Для распространения программного приложения, позволяющего дактилировать отдельные буквы и слова, использовано обобщенный способ оптимальной передачи динамично созданного изображения в сети Интернет с возможностью управления им [4]. Поскольку вычислительные возможности компьютеров широкого круга пользователей значительно отличаются, также как и возможности их каналов связи, данный аспект был учтен при продумывании способа массового обслуживания, посредством передачи данных через сеть Internet. Разработанная технология имеет ряд преимуществ над уже существующими, которые или не имеют интерактивности, или для ее достижения необходимо использовать специальные плагины, которые необходимо установить на компьютер клиента.

Технология передачи данных содержит трехуровневую информационную модель коммуникации между серверной и клиентскими частями: на первом (высшем) логическом уровне видеоинформация передается из серверной стороны на сторону клиента. На следующем втором логическом уровне, сервер кодирует видеоинформацию, фрейм за фреймом, клиент проводит инициализацию (предусматриваемую сервером), а также воспроизводит картинку на основе команд. На этом логическом уровне программист (инженер, разработчик) может учитывать специфику задачи и структуру медиаданных, т.е. предоставить собственную пару компрессора-декомпрессора данных. На этом уровне отдельный поток отвечает за обслуживание именных каналов и передачу команд компрессора. Рекомендуемый уровень компрессии выходит из расчета, что 10 процентов времени канал должен либо простаивать, либо резервируется на непредвиденную ситуацию, например, передачу данных другим программным приложениям на стороне клиента. На следующем третьем логическом уровне команды сериализуются на стороне сервера и десериализуются на стороне клиента. Это последний логический уровень данной технологии. Информация на данном уровне передается с использованием HTTP (Silverlight / Asp.net).

Выводы

В данной статье впервые для воспроизведения дактильных украинских жестов использовано анимацию трехмерной модели руки, рассмотрены алгоритмы расчета поверхности руки с помощью трехмерной модели руки, базирующейся на скелетной модели, выполнена реализация этих алгоритмов и практическая апробация, рассмотрен алгоритм анимации дактилирования слова на основе данных об отдельных дактилемах, усовершенствованы алгоритмы подготовки кадра для воспроизведения процесса скелетной анимации дактильного жестового языка за счет поддержки многоядерных процессоров. Данные алгоритмы и методы совместимы с технологией отображения информации средствами Интернет для воспроизведения процесса анимации дактильного жестового языка, что позволит создать соответствующее программное приложение.

Дальнейшие исследования будут направлены на повышение качества отображения руки, улучшение алгоритмов отображения с целью поддержки использования программы на компьютерах с меньшей мощностью.

Благодарности

Статья частично финансирована из проекта ITHEA XXI Института Информационных теорий и Приложений FOI ITHEA и Консорциума FOI Bulgaria (www.itea.org, www.foibg.com).

Библиография

[Кульбіда С. В.] Кульбіда С. В. Українська дактилологія. К. : Педагогічна думка, 2007. — 255 с. — ISBN 978-966-644-076-4.

[Воскресенский А. Л.] Воскресенский А. Л., Халагин Г. К. От звучащей речи к жестовой // Речевые технологии. — 2009. — № 1. — С. 99—106.

[Бармак О. В.] Бармак О. В., Крак Ю. В., Троценко Б. А. Технологія оптимізованої передачі жестової мови в мережі інтернет // Проблеми програмування. — 2009. — № 3. — С. 73—80.

[Domine S.] Domine. S. Mesh Skinning — Режим доступа: http://developer.nvidia.com/attach/6662. — Название с экрана.

[Merritt R.] Merritt R. CPU designers debate multi-core future — Режим доступа: http://www.eetimes.com/showArticle.jhtml?articleID=206105179. — Название с экрана.

[Agarwal A.] Anant Agarwal, Srinivas Devadas, Henry Hoffmann Partitioning Strategies for Concurrent Programming // Technical Report MIT-CSAIL-TR-2009-026. — 2009. — Режим доступа: http://dspace.mit.edu/bitstream/handle/1721.1/45567/MIT-CSAIL-TR-2009-026.pdf. — Название с экрана.

[Massingill B.] B. Massingill, T. Mattson, B. Sanders. Patterns for parallel programming — Reading, Massachusetts : Addison-Wesley Professional, 2004. — 384 p.

Authors' Information

Iurii Krak *– senior scientist, V.M.Glushkov Cybernetics Institute of NASU, 40 Glushkov ave., Kiev, Ukraine, 03680, krak@unicyb.kiev.ua*

Iurii Kryvonos *- Director Deputy, V.M.Glushkov Cybernetics Institute of NASU, 40 Glushkov ave., Kiev, Ukraine, 03680*

Bohdan Trotsenko *– junior scientist, V.M.Glushkov Cybernetics Institute of NASU, 40 Glushkov ave., Kiev, Ukraine, 03680bohdan@trotsenko.com.ua*