РЕЗЮМЕТА НА НАУЧНИТЕ ТРУДОВЕ
ЗА УЧАСТИЕ В КОНКУРСА

(Анотации на материалите по чл. 65. от ПРАСПУ за участие в конкурса, включително самооценка на приносите)
за заемане на академичната длъжност „доцент"
на гл. ас. д-р Александър Пламенов Пенев

За участие в настоящия конкурс (виж Списък на научните трудове за участие в конкурса) са избрани:

- 11 научни труда, в това число 6 статии в научни списания;
- 1 учебно помагало;
- 1 автореферат;

I. ПУБЛИКАЦИИ В СПИСАНИЯ

1. D. Dimov, D. Kralchev, **A. Penev**, *Cliques, packings of segments and binary matrices*, Plovdiv University "Paisii Hilendarski" Scientific Works, Plovdiv, 2007, vol.35 Book 3 – Mathematics, pp. 33-45 [ISSN 0204-5249]

Задачата за пакетиране на отсечки/сегменти има голямо разнообразие от приложения. Чрез интерпретация в термините на теорията на графите (търсене на клики в граф) задачата се свежда до търсене на подматрица от определен вид. В статията е описан „самонаблюдаващ се" алгоритъм, който постига добра производителност при търсенето. Тази идея е удобна за задачи, възникващи в практиката.

Класическите техники за търсене, комбинирани със самонаблюдение, ни позволиха да изградим сравнително бърз и надежден алгоритъм за намиране на пакетирания на отсечки/ сегменти. Направената практическа реализация показа средно време за търсене за големи матрици е около 5 сек., а надеждността му е над 97%, което е достатъчно за повечето случаи, които се срещат в практиката.

2. D. Kralchev, D. Dimov, **A. Penev**, *A Reflexive Algorithm for the Rook Problem*, Plovdiv University "Paisii Hilendarski" Scientific Works, Plovdiv, 2007, vol.35 Book 3 – Mathematics, pp. 67-72 [ISSN 0204-5249]

Нека имаме шахматна дъска NxN, някои от нейните полета са забранени, а останалите са разрешени. Може ли да се поставят N топа на позволените полета на шахматната дъска, така че да не се атакуват един на друг? Така може да формулираме т.нар. Задачата за топовете. Съществуват множество еквивалентни на нея задачи.

В статията е предложен нов евристичен алгоритъм за решаване на задачата за топовете. Алгоритъмът е рефлексивен: наблюдава своето време за изпълнение, което се намира във взаимна връзка с резултата. Рефлексивните алгоритми са подходящи, когато резултата е в корелация с някои лесно разпознаваеми характеристики на алгоритъма. Тук ние използваме времето за изпълнение, но това не е единствената възможност.

В Компютърната графика (например при визуализация на големи и сложни сцени) често се използва подобен подход за прекъсване на изчислението, когато резултатът е близък (визуално) до търсения. Ако евристиката гарантира точен резултат в голям процент от случаите, то визуалният резултат е приемлив и не може да бъде различен от човек.

3. D. Kralchev, D. Dimov, **A. Penev**, *A Reflexive Algorithm for Existence of Null-submatrices*, Plovdiv University "Paisii Hilendarski" Scientific Works, Plovdiv, 2009, vol.36 Book 3 – Mathematics, pp. 59-66 [ISSN 0204-5249]

В статията е предложен евристичен алгоритъм за намиране съществуването на нулеви под-матрици в големи разредени матрици. Алгоритъмът е рефлексивен: изследва собственото си потребление на памет, което е в корелация с неговия резултат.

4. H. Lesev, **A. Penev**, *A Framework for Visual Dynamic Analysis of Ray Tracing Algorithms*, Cybernetics and Information Technologies, vol. 14, No.2, 2014, pp. 38-49, [ISSN 1311-9702, 1314-4081], SJR=0.142

В статията е представен е нов подход за записване на данни с голям обем за състоянието на алгоритми от класа на Ray tracing и последващ динамичен анализ и интерактивна визуализация на изпълнението на алгоритъма. Описаната софтуерна рамка (framework) подпомага извличането на данни за светлинните пътища, проследявани от алгоритмите за визуализация, и предоставя мощна филтрираща подсистема, подпомагаща интерактивната визуализация и изследване на желаното подмножество от записани данни. Предложен е гъвкав формат за записване на данни и ускоряващи структури за по-лесен достъп и филтриране. Реализацията на софтуерната рамка и инструменти имат разширима plugin-базирана архитектура. Рамката осигурява API за регистриране на данни за вграждане в многонишкови разпределени рендери. Инструментите за визуализация и филтриране позволяват по-добро разбиране на алгоритмите за трасиране на лъчи, както за начинаещи, така и за експерти.

5. V. Vassilev, **A. Penev**, M. Vassilev, *SolidOpt – A Multi-Model Software Optimization Framework*, International Journal of Computer Science Issues, vol. 12, No.2, 2015, pp. 32-41, [ISSN 1694-0814, 1694-0784]

Тази статия е естествено продължение и развитие на една предишна наша разработка (К6). Тя представя софтуерната рамка (framework), SolidOpt, която помага за автоматизираната и динамична оптимизация на софтуерните системи през целия им жизнен цикъл. Оптимизациите обикновено са в основата на (оптимизиращите) компилатори, докато в предложения подход акцента се измества в по-късните етапи на жизнения цикъл на системите (тези след разработка). За да се постигнат по-добри резултати, SolidOpt осигурява множество представяния на различни аспекти на софтуерните системи (повечето от които се получават от изпълнимия код на системата чрез декомпилация), такива като Control Flow

Graph, Call Graph, Three Address Code, Abstract Syntax Tree и др. Ние разглеждаме подход на "непрекъсната оптимизация" (continuous optimization), който се съобразява с околната средата на програмата и извършва оптимизации на различни нива на абстракция (върху различните представяния). Тези оптимизационни методи са както известните класически методи (от оптимизиращите компилатори), така и специфични за дадената предметна област методи.

Макар че проектът SolidOpt може да се прилага във различни предмети области, една от основните за които той е създаден е Компютърната графика. Друга свързана област са Визуалните езици за програмиране.

6. A. Golev, **A. Penev**, K. Stefanova, S. Hristova, Using GPU to Speed Up Calculation of Some Approximate Methods for Fractional Differential Equations, International Journal of Pure and Applied Mathematics, Academic Publications, vol. 119, No. 3, 2018, pp. 391-401, [ISSN 1311-8080], SJR=0.139

В статията е описан метод за ефективно изчисление на някои класове частни диференциални уравнения с помощта на съвременен графичен хардуер (GPU).

Ефективното преобразуване на алгоритми предназначени за изпълнение от централния процесор към силно паралелни архитектури, каквито са съвременните графични процесори не е тривиална задача. То изисква методи близки до използваните в компютърната графика.

II. ПУБЛИКАЦИИ НА КОНФЕРЕНЦИИ

7. Д. Кралчев, Д. Димов, **А. Пенев**, *Пакетиране на отсечки*, Научно-приложна конференция „Математика, информатика и компютърни науки“, Велико Търново, 2006, стр. 69-74

Тази статия разглежда проблема за намиране на пакетиране от сегменти/отсечки, които имат много практически приложения като: пространствено разпределяне на визуални елементи, разпределяне на времето на процесора, генериране на разписания и др. Проблемът има два варианта – непрекъснат и дискретен. Целта на статията е да се изследват свойствата на множеството от пакетирания и да се сведе непрекъснатият случай до дискретен, защото е по-лесно (и по-бързо) да се реши последната задача (за практически цели). Статията е продължение на предишни разработки на авторите.

В съвременната Компютърна графика (КГ) все по често се налага оптимално разпределяне на алгоритмите за визуализация (и други алгоритми) в многопроцесорни и хетерогенни системи, както и в мрежа и др. Например в силно паралелните Ray Tracing алгоритми правилното разпределяне (разписание) на изчислителните ресурси (CPU, GPGPU, ...) е от ключово значение за ефективното им натоварване. Друг пример в областта на КГ е хоризонталното (и или вертикално) подреждането на снимки в албуми и последващата им визуализация.

8. **A. Penev**, D. Dimov, D. Kralchev, *Acceleration of structured and heterogeneous configuration of the applications*, Proceedings of the 36th conference of the Union of Bulgarian Mathematicians, Varna, 2007, pp. 327-331 [ISBN 978-954-8880-25-1]

Тази статия третира някои аспекти на конфигурирането на приложения, библиотеки, обектни класове и др. В нея е предложен хетерогенен и структуриран подход за дефиниране на параметрите на системите въз основа на йерархии на двойки атрибут-стойност с допълнителни метаданни. Този подход се базира на множество нива и множество формати на конфигурационните източници (файлове, бази данни и т.н.). За да бъде постигната по-голяма производителност е предложен метод за кеширане на част от конфигурационните източници (тези които са нечувствителни към сигурността).

Подхода описан в статията е залегнал в основата на конфигурационната подсистема на софтуерната рамка (framework) OpenF, която е и предмет на разглеждане на докторската ми дисертация („Отворени хибридни системи за геометрично моделиране“). Този подход, разбира се е много общ и излиза далеч от границите на Компютърната графика. В последствие той беше приложен (и усъвършенстван) успешно и в други области, каквито са Визуалните среди и езици за програмиране.

9. **A. Penev**, V. Vassilev, T. Petrov, *SolidOpt – Innovative Multiple Model Software Optimization Framework*, Second International Conference on Creativity and Innovation in Software Engineering (CISE-2009), IEEE UKRI Section, STRL, Ravda (Nessebar), Bulgaria, 2009, 10-12 June 2009

Тази статия обсъжда създаването на софтуерна рамка (framerowk) наречена SolidOpt, която помага за автоматизирана и динамична оптимизация на софтуер (алгоритми и данни). Определени са основните идеи и принципи в разработването на SolidOpt. Представени са предимствата и необходимостта от използване на множество модели (и йерархия от модели) на софтуера. Показано е значението на обкръжаващата среда (софтуер, хардуер, данни, потребители и др.) за оптималното изпълнение на компютърните програми. Разглеждан е еволюционен подход за оптимизиране на софтуерната система в зависимост от поведението ѝ в околната среда. Представени са някои предварителни резултати от разработването на експерименталната рамка.

В областта на Компютърната графика има голямо приложение възможността системите да са оптимизирани максимално добре за конкретния (понякога и специализиран) хардуер върху които те работят. Много са факторите обуславящи тази необходимост: наличието на алгоритми изискващи много изчисления; големия обем от данни върху които тези алгоритми работят; разнообразния и понякога хетерогенен хардуер и др. Съвременните компилатори по много причини не се справят винаги добре с задачата за оптимизация. Те прилагат общи техники на оптимизация на алгоритмите. Ако имаме система, която освен основните класически оптимизации над софтуер да може да прилага: проблемно ориентирани (домейн-специфични) оптимизации на алгоритъма; понякога дори не еквивалентни оптимизации (а понякога и оптимизации над данните – апроксимация и др.); да се съобразява с данните; да се съобразява с това как потребителите използват тези данни; да работи на различни нива на абстракция; то такава система би била много по-приложима в областта на КГ. Опит за създаване на такава система е създаването на SolidOpt.

10. **A. Penev**, *F-Rep Designer*, Proceedings of the International Conference “From DeLC to Velspace“, Plovdiv, 26-28 March 2014, pp. 239-247 [ISBN 0-9545660-2-5]

В статията е представено компютърно приложение за 3D геометрично моделиране, базирано на функционална представяща (F-Rep). Това приложение, наречено F-Rep Designer, използва платформено независима хибридна CPU/GPGPU интерактивна реализация на алгоритъма за визуализация Ray tracing. Една от основните характеристики на реализацията на системата е, че тя не използва в нито един етап от работата си т.нар. Гранично представяне (Boundary Representation, B-Rep). Малкото известни подобни системи (базирани на F-Rep представяне) почти винаги минават през т.нар. етап на полигонизация на модела, независимо дали става въпрос за визуализация, за взаимодействие с потребителя или и двете.

11. V. Vassilev, M. Vassilev, **A. Penev**, L. Moneta, V. Ilieva, *Clad – Automatic Differentiation Using Clang and LLVM*, Journal of Physics: Conference Series, IOP Publishing, vol. 608, article 01205512, 2015, 10 pp., [ISSN 1742-6588, 1742-6596], doi:10.1088/1742-6596/608/1/012055, SJR=0.24

Диференцирането е широко използвано в много приложения на математиката. Статията е посветена на т.нар. Автоматично диференциране (AD), чрез което може да се избегнат известните проблеми със закръгленията и скоростта, от които страдат численото и символното диференциране. Това се постига чрез трансформиране на изходния код на функциите (такъв какъвто е запрограмиран в съответната приложна програмата). И трите подхода (числен, символен и автоматичен /AD/) имат своите предимства и недостатъци, но AD има много добро приложение в Компютърната графика, при изчисление на нормалните вектори в дадена точка от повърхността на тяло описано функционално (както е например в представящата схема F-Rep и други схеми базирани на неявно представяне). Важно е да се отбележи, че функцията описваща тялото може да не бъде от напред известна т.е. да се въвежда или определя от потребителя моделиращ дадена сцена. Това прави най-удачно използването на AD, защото при другите подходи има проблеми: при числения се губи няколко пъти повече време за изчисляване на функцията, както и понякога намирания по този начин е числено нестабилен; при символния се изисква непрекъсната интерпретация на съответния израз, което също много пъти по-бавно. При нашия подход за AD тези проблеми се избягват, като в допълнение получаваме предимствата на съвременните оптимизиращи компилатори, които допълнително подобряват крайния резултат.

Тази статия показва и друга концепция – как LLVM, Clang и Cling опростяват създаването на подобен инструмент. Идеята е да разширим семантиката на езика (в случая на C++) посредством допълване на компилатора на езика с допълнителен модул за трансформация на вътрешното представяне на компилираната програма. Този подход е различен от повечето известни класически реализации на AD. Ние ще демонстрираме прост прототип на доказателство за концепция, наречен Clad, който е в състояние да генерира производни от n-ти ред на произволни C++ функции (включително темплейти, функтори и виртуални методи) и други езикови конструкции.

В статията се демонстрира също така как Clad може да разтовари (offload) изчисления от CPU на графичните процесори GPGPU, използвайки OpenCL.

III. УЧЕБНО ПОМАГАЛО

12. Д. Димов, А. Пенев, *Ръководство за упражнения по Компютърна графика*, Пловдивско университетско издателство, 2002, 142 стр., ISBN: 954-423-263-X.

В ръководството за упражнения по компютърна графика се предлага подход за реализация на диалогови системи за графично/геометрично моделиране. Подходът е съобразен с ключовите проблеми при реализирането на такъв род системи – архитектура, схема за реализиране, преносимост и техники за реализация на основни диалогови операции. Крайната цел е да се създаде система-ядро, която лесно може да бъде надстройвана за нуждите на различни предметни области, в които данните имат геометрична интерпретация. Изборът на OpenGL като графична платформа е продиктуван единствено от факта, че този стандарт е широко разпространен. Същия подход може да се следва и при избор на други графични платформи.

Ръководството се използва успешно над 15 години като помагало при провеждането на упражнения по дисциплините: „Компютърна графика“ в специалността Информатика (бакалаври) на ФМИ на ПУ „П. Хилендарски“; „Компютърна графика и презентации“ на специалността Математика и информатика (бакалаври); „Компютърна графика и графичен потребителски интерфейс“ на специалността Софтуерно инженерство (бакалаври). То е и в помощ на студентите при провеждане на избираемата дисциплина „Графични стандарти: OpenGL“.
Ръководството следва проектно-базирания подход в практическото обучение на студентите. То може да бъде използвано и за самостоятелни занимания от всички, владеещи основите на компютърната графика.

Изготвил:

ГЛ. АС. Д-Р АЛЕКСАНДЪР ПЛАМЕНОВ ПЕНЕВ

12.07.2018 г.
гр. Пловдив