УДК 004:519.876, 004.93

А.М. Касім

Формалізація процесу обробки даних для растрового маскування прозорості зображень рухомих об'єктів

Предложен метод обеспечения растровой прозрачности фоновых пикселов разноформатных символов подвижных объектов при их выводе на картографическом фоне. Суть метода сводится к выполнению в промежуточном буфере двух последовательных формализованных растровых операций с масками прозрачности *AND* и *OR*, что гарантирует приоритетный относительно картографического фона вывод прозрачных спрайтов без необходимости соблюдения требования непопадания значения фонового прозрачного цвета в состав цветовых кодов в контуре изображения отдельного символа.

Ключевые слова: динамическая сцена, маска прозрачности, растровые логические операции, спрайт, картографический фон.

Запропоновано метод забезпечення растрової прозорості фонових пікселів різноформатних символів рухомих об'єктів при їх виведенні на картографічному фоні. Суть методу зводиться до виконання в проміжному буфері двох послідовних формалізованих растрових операцій з масками прозорості *AND* і *OR*, що гарантує пріоритетне відносно картографічного фону виведення прозорих спрайтів без необхідності дотримання вимоги непотрапляння значення фонового прозорого кольору до складу колірних кодів у контурі зображення окремого символу.

Ключові слова: динамічна сцена, маска прозорості, растрові логічні операції, спрайт, картографічний фон.

Вступ. Комп'ютерна графіка була і залишається однією з найпривабливіших галузей прикладного програмування. Її значення в сучасному світі, як й інформаційних технологій, складно переоцінити, адже вона впевнено інтегрувалася в різноманітні сфери людської діяльності: наукові дослідження, дослідно-конструкторські розробки, медицина, мистецтво, комп'ютерні ігри тощо.

Особливий інтерес при створенні графічних програмних засобів викликають анімаційні додатки, пов'язані з генерацією динамічних зображень у реальному часі. Ареал їх застосування охоплює розв'язання важливих науково-прикладних задач, як-то побудова систем візуалізації диспетчерських тренажерів, графічне моделювання динамічних явищ і процесів, розробка систем автоматизованого диспетчерського управління рухомими об'єктами, проектування моніторингових модулів охоронних систем, створення програмних прототипів та імітаційних моделей динаміки руху об'єктів, які переміщуються по поверхні Землі та в навколоземному просторі.

Стрижнем, що пронизує функціонування згаданих систем, є візуалізація динамічних образів рухомих об'єктів, представлених у векторному та растровому форматах, на фоні електронної карти заданої ділянки місцевості. При цьому задача синтезу динамічної сцени у реальному часі полягає у формуванні послідовності кадрів зображень сцени зі швидкістю, достатньою для забезпечення ефекту плавності руху – 50 кадрів за секунду. Науковою базою для розв'язання означеної задачі є роботи вітчизняних та зарубіжних вчених [1–7]. Проте в жодному з наведених літературних джерел не міститься відомостей щодо формалізації процесу обробки даних растрового маскування прозорості різноформатних символів.

Постановка задачі

Зважаючи на те, що створення логіко-математичних моделей та програмних засобів для розв'язання вказаних прикладних задач потребує формалізації графікоємних даних та розробки методів роботи з ними, *метою даної статті* є висвітлення питання формалізованого представлення процесу растрового маскування прозорості спрайтів.

Основна частина

У межах синтезу кожної динамічної сцени однією з перших алгоритмічних процедур є

нанесення у місцевизначені області картографічного фону сконструйованих за певними правилами стилізованих символів, кутовий ракурс яких детермінує поточну орієнтацію відповідних рухомих об'єктів. Для виконання зазначеної процедури використовуються методи, що ґрунтуються на принципах точкового та растрового маскування. Розвитку цих методів, які дають значний приріст обчислювальної продуктивності, або з урахуванням специфічної структури зображення маски прозорості, або шляхом створення додаткового списку прозорих зображень, присвячено роботи [8–12].

Растровий символ рухомого об'єкта – спрайт – можна зберігати у файлі як набір бітів тільки у прямокутному вигляді, а на екран необхідно виводити тіло, обмежене контуром символу, яке має довільну форму. При індикації спрайту з пріоритетом до картографічного фону останній має зникати лише в тих місцях, де виводяться точки власне зображення символу, а не прямокутника, куди це зображення вписано.

Отже, в комп'ютерну програму потрібно закласти функцію ідентифікації, яка б визначала, котрі точки спрайту є «істотними», активними, тобто складовими образу символу, а які не є такими, тобто становлять фон прямокутного зображення. Для цього використовують концепцію прозорого кольору, яка підтримується більшістю графічних бібліотек (*OpenGL*, *DirectX*), а також бібліотеками сучасних середовищ програмування, у тому числі й *MFC* та *VCL* [4–9].

Зокрема, у середовищі *Borland Delphi*, якщо властивість *Transparent* бітового образу класу *TBitmap* має значення *true*, то метод *Draw* генерує зображення растрового символу без його фону. Залежно від значення властивості *TransparentMode* прозорий колір визначається або нижнім лівим елементом бітового образу, або властивістю *TransparentColor*. При цьому фіксований колірний код не повинен зустрічатися в межах контуру символу, тому якщо не виключити його звідти, забарвлення результуючого кадру виявиться непередбачуваним [9–12].

На практиці процедура відображення прозорих спрайтів на картографічному фоні спирається на принцип точкового маскування [9, 11, 12]. Маскування з точковою перевіркою передбачає, що зображення кадру з символом промальовується поелементно: перед виведенням чергового піксела його колір порівнюється зі специфічним колірним кодом, явно прийнятим як прозорий. Якщо результат порівняння негативний, що свідчить про незбіг значень порівнюваних кольорів, даний піксел включається в остаточне зображення, інакше – пропускається. Недолік цього методу полягає в тому, що прозорий колір можна використовувати тільки для прозорих пікселів спрайта. Тоді в цьому випадку процес створення спрайта векторним способом має відбуватися у відповідності до наступних дій:

- як фоновий вибирається який-небудь колір, що не зустрічається в контурі символу, наприклад, яскраво-рожевий, брудно-зелений і подібні кольори;
- тіло спрайта генерується на прямокутнику, який повністю зафарбовано цим кольором;
- після виконання двох попередніх умов обраний колір призначається прозорим.

Щоб уникнути спотворення кінцевого кадру при реалізації ефекту прозорості спрайтів з довільними різнокольоровими контурами на противагу використанню описаної техніки пропонується метод забезпечення прозорості фону растрових символів [12], в основу якого покладено принцип растрового маскування [9–11], згідно з яким кожен кадр символу описується двома бітовими образами – масками растрових логічних операцій *AND* («побітове І») і *OR* («побітове ЧИ») (рис. 1). *OR*-маска фактично є зображенням майбутнього символу в тому вигляді, в якому він має з'явитися на результуючому кадрі. У цьому випадку як колір прозорих пікселів вибрано чорний (індекс нуль для палітри з 256 кольорів – всі біти встановлені в нуль). *AND*-маска – це бінарний чорно-білий силует спрайта. Для такої маски чорна область збігається з контуром символу на *OR*-масці, а білі зони (індекс палітри – 255, всі біти встановлені в одиницю) об'єднують прозорі пікселі.

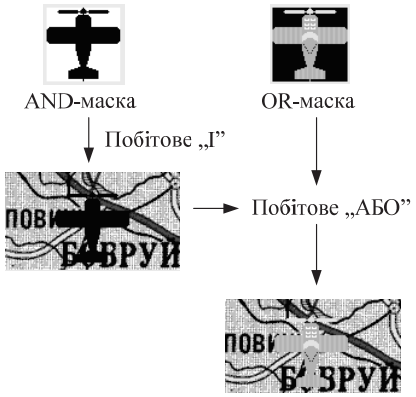

Рис. 1. Процес створення на картографічному фоні прозорого спрайту за принципом растрового маскування

Прозорість фону зображення символу забезпечується таким чином завдяки його *OR*- і *AND*-маскам прозорості. Процес створення другої маски можна спростити, організувавши її формування на основі першої за визначеним правилом – там, де на *OR*-масці – чорний колір, в *AND*-масці має бути білий, а всі інші кольори *OR*-маски потрібно замінити на чорний. Для переведення *OR*-маски в *AND*-маску створюємо процедуру, яка за допомогою вкладених операторів циклу *For* сканує зображення символу *OR*-маски по рядках і стовпчиках та здійснює для кожної точки наступне перетворення: якщо колір точки – чорний, то замість неї необхідно вивести білу точку, інакше – вивести чорну.

Основна ідея методу полягає в тому, щоб скопіювати картографічний фон у буфер і перемалювати усі маски поверх цього фону. Спочатку до канви фонового картографічного зображення застосовуємо растрову операцію *AND* з одноіменною маскою. Ця операція ніби вирізає чорний отвір в тому місці канви, де з'явиться спрайт з прозорим фоном. Потім отриманий бітовий образ логічно складається з *OR*-маскою. Чорні області *OR*-маски зберігають канву результуючого зображення точно так, як і білі на першому кроці.

Для формалізації даного процесу введемо точкову матрицю растрового символу розміром $[w \times h]$ пікселів: $\left(a_{xy}\right)_{x=0,y=0}^{w-1,h-1}$. Позначимо через *Spr* корисну частину спрайта, тобто ту область растрового символу, яка є безпосередньо зображенням спрайта з відповідними кольорами пікселів *spr_cl_code* у координатах (x, y), а через *SprBackgr* – фонову область спрайта, яка не підлягає виведенню разом з *Spr*. Тоді перша маска *MaskOR* визначається бітовою матрицею, в якій фон спрайта має чорний колір:

$$a_{xy} = \begin{cases} spr\_cl\_code, & (x,y) \in Spr \\ 0, & (x,y) \notin Spr, \quad (x,y) \in SprBackgr \end{cases},$$

а для другої *MaskAND* задамо:

$$b_{xy} = \begin{cases} 0, & (x,y) \in Spr \\ 1, & (x,y) \notin Spr, \quad (x,y) \in SprBackgr \end{cases},$$

в якій фон спрайта має білий колір, а його зображення – чорний.

При цьому маска прозорості *MaskAND* повинна мати такі ж розміри, що й основний растр *MaskOR*. Аналогічна вимога також стосується решти породжених матриць.

Спрайт з прозорим фоном виводиться на скопійовану в проміжний буфер ділянку картографічного фону *BufMapBackgr* з відповідними кольорами пікселів *map_cl_code* у координатах (x, y):

$$c_{xy} = \{map\_cl\_code,\ (x,y) \in BufMapBackgr\}.$$

В результаті накладання *MaskAND* на *BufMapBackgr*, використовуючи операцію «І», останній модифікується:

$$d_{xy} = b_{xy} \text{AND } c_{xy} =$$

$$= \begin{cases} 0, & (x,y) \in Spr \\ map\_cl\_code, (x,y) \in SprBackgr \end{cases},$$

тобто при виведенні білої частини *MaskAND* на *BufMapBackgr* в режимі «І» відповідна область *BufMapBackgr* залишається незмінною, а при виведенні її чорної частини – відповідна область *BufMapBackgr* зафарбовується чорним кольором.

В результаті виконання операції «ЧИ» над *MaskOR* і модифікованим *BufMapBackgr* отримуємо бітову матрицю:

$$e_{xy} = a_{xy} \text{OR } d_{xy} =$$

$$= \begin{cases} spr\_cl\_code, & (x,y) \in Spr \\ map\_cl\_code, & (x,y) \in SprBackgr \end{cases},$$

тобто при виведенні чорної частини *MaskOR* на модифікований *BufMapBackgr* в режимі «ЧИ» відповідна область *BufMapBackgr* залишається незмінною, а при виведенні кольорової частини – відповідна область *BufMapBackgr* набуває вигляду корисного зображення спрайта.

Головна перевага запропонованого методу – забезпечення прозорості різнокольорових растрових символів.

Висновки. Отже, при синтезі динамічних сцен для реалізації растрової прозорості символів рухомих об'єктів зручніше користуватися запропонованим формалізованим підходом забезпечення прозорості фонових ділянок спрайтів, не чутливим до набору кольорів у складі контуру кожного конкретного символу. Це в свою чергу означає незалежність від вимоги неперетинання фонового та належних тілу спрайта кольорів. Більше того, завдяки формалізації задачі пріоритетного виведення прозорих зображень символів на картографічний фон вдалося виділити класи операцій, загальні для багатьох існуючих методів формування динамічних сценаріїв в частині досягнення ефекту прозорості, а саме – побудову маски прозорості, її використання з урахуванням взаємодії пікселів зображень джерела та приймача, перенесення бітів зображення з джерела в місце призначення (біт-бліттінг). Попереднє виконання порозрядних логічних операцій типу *AND* і *OR* з початковими та результуючими пікселами у спеціально відведеному відеобуфері згідно запропонованої техніки «прозорих» кольорів дозволяє також уникнути можливого миготіння при імплементації процесу бліттінга.

1. *Метод* организации динамической сцены, выводимой на экран геоинформационных комплексов реального времени / И.М. Васюхина, А.М. Касим, А.Е. Кулик и др. // Вестн. Херсон. нац. техн. ун-та. – 2005. – № 1(**21**). – С. 207–210.
2. *Касим А.М.*, *Васюхин М.И.*, *Капштык О.И.* Алгоритмы построения зрительных сцен в аэронавигационных геоинформационных системах реального времени // УСиМ. – 2008. – № 3. – С. 79–84.
3. *Методи* створення динамічних графічних образів при вирішенні задач відображення поточної обстановки на території аеропорту та прилеглих до нього зонах / А.М. Касім, М.І. Васюхін, В.Д. Гулевець та ін. // АСУ и приборы автоматики: всеукр. межвед. науч.-техн. сб. – Харьков: Изд-во ХНУРЭ, 2010. – **151**. – С. 112–118.
4. *Глушаков С.В.*, *Клевцов А.Л. Delphi* 2007: Самоучитель. – М.: АСТ Москва, 2008. – 448 с.
5. *Культин Н.Б.* Основы программирования в *Turbo Delphi*. – СПб.: БХВ-Петербург, 2007. – 384 с.
6. *Тюкачев Н.А.*, *Илларионов И.В.*, *Хлебостроев В.Г.* Программирование графики в *Delphi*. – Там же, 2008. – 784 с.
7. *Жарков В.А.* Самоучитель по анимации и мультипликации в *Visual C++ .NET* 2003. – М.: Жарков Пресс, 2003. – 448 с.
8. *Мозговой М.В.* Занимательное программирование: Самоучитель. – СПб.: Питер, 2005. – 208 с.
9. *Шамис В.А. C++ Builder Borland Developer Studio* 2006. Для профессионалов. – Там же, 2007. – 781 с.
10. *Набайоти Баркакати*. Программирование игр для *Windows* на *Borland C++*. – М.: БИНОМ, 1995. – 512 с.
11. *Секреты* программирования игр: / Ла Мот А., Д. Ратклифф, М. Семинаторе и др. – СПб.: Питер, 1995. – 720 с. (Советы профессионала).
12. *Касім А.М.* Методи та засоби формування динамічних сценаріїв в навігаційно-керувальних комплексах: Дис. … канд. техн. наук. – К., 2013. – 200 с.

Поступила 17.12.2015

Тел. для справок: +38 044 526-3518 (Киев)
E-mail: *aneesa.qasem@gmail.com*

А.М. Касим

Формализация процесса обработки данных для растрового маскирования прозрачности изображений движущихся объектов

Введение. Компьютерная графика была и остается одной из наиболее привлекательных отраслей прикладного программирования. Ее значение в современном мире, так же как и информационных технологий, сложно переоценить, поскольку она уверенно интегрировалась в различные сферы человеческой деятельности: научные исследования, опытно-конструкторские разработки, медицину, искусство, компьютерные игры и др.

Особый интерес при создании графических программных средств вызывают анимационные приложения, связанные с генерацией динамических изображений в реальном времени. Ареал их применения охваты-

вает решение важных научно-прикладных задач, таких как построение систем визуализации диспетчерских тренажеров, графическое моделирование динамических явлений и процессов, разработка систем автоматизированного диспетчерского управления подвижными объектами, проектирование мониторинговых модулей охранных систем, создание программных прототипов и имитационных моделей динамики движения объектов, перемещающихся по поверхности Земли и в околоземном пространстве.

Стержнем, пронизывающим функционирование упомянутых систем, является визуализация динамических образов движущихся объектов, представленных в векторном и растровом форматах, на фоне электронной карты заданного участка местности. При этом задача синтеза динамической сцены в реальном времени заключается в формировании последовательности кадров изображений сцены со скоростью, достаточной для обеспечения эффекта плавности движения – 50 кадров в секунду. Научной базой для решения обозначенной задачи служат работы отечественных и зарубежных ученых [1–7]. Однако ни в одном из приведенных литературных источников не содержится сведений о формализации процесса обработки данных растрового маскирования прозрачности разноформатных символов.

Постановка задачи

Учитывая, что создание логико-математических моделей и программных средств для решения указанных прикладных задач требует формализации графикоемких данных и разработки методов работы с ними, *цель данной статьи* – освещение вопроса формализованного представления процесса растрового маскирования прозрачности спрайтов.

Основная часть

В пределах синтеза каждой динамической сцены одна из первых алгоритмических процедур – это нанесение в местоопределенные области картографического фона сконструированных по определенным правилам стилизованных символов, угловой ракурс которых детерминирует текущую ориентацию соответствующих подвижных объектов. Для выполнения отмеченной процедуры используются методы, базируемые на принципах точечного и растрового маскирования. Развитию этих методов, дающих значительный прирост вычислительной производительности, либо с учетом специфической структуры изображения маски прозрачности, либо путем создания дополнительного списка прозрачных изображений, посвящены работы [8–12].

Растровый символ движущегося объекта – спрайт – можно хранить в файле как набор битов только в прямоугольном виде, а на экран необходимо выводить тело, ограниченное контуром символа произвольной формы. При индикации спрайта с приоритетом к картографическому фону последний должен исчезать лишь в тех местах, где выводятся точки собственно изображения символа, а не прямоугольника, в который это изображение вписано.

Итак, в компьютерную программу следует заложить функцию идентификации, которая бы определяла, какие точки спрайта «существенны», активны, т.е. составляющие образ символа, а какие не являются таковыми, т.е. представляют фон прямоугольного изображения. Для этого используют концепцию прозрачного цвета, которая поддерживается большинством графических библиотек (*OpenGL*, *DirectX*), а также библиотеками современных сред программирования, в том числе *MFC* и *VCL* [4–9].

В частности, в среде *Borland Delphi*, если свойство *Transparent* битового образа класса *TBitmap* имеет значение *true*, то метод *Draw* генерирует изображение растрового символа без его фона. В зависимости от значения свойства *TransparentMode* прозрачный цвет определяется либо нижним левым элементом битового образа, либо свойством *TransparentColor*. При этом фиксированный цветовой код не должен встречаться в рамках контура символа, поэтому если не исключить его оттуда, окраска результирующего кадра окажется непредсказуемой [9–12].

На практике процедура отображения прозрачных спрайтов на картографическом фоне опирается на принцип точечного маскирования [9, 11, 12]. Маскировка с точечной проверкой предусматривает, что изображение кадра с символом прорисовывается поэлементно: перед выводом очередного пиксела его цвет сравнивается со специфическим цветовым кодом, явно принятым как прозрачный. Если результат сравнения отрицательный, что свидетельствует о несовпадении значений сравниваемых цветов, данный пиксел включается в окончательное изображение, иначе – пропускается. Недостаток этого метода заключается в том, что прозрачный цвет можно использовать только для прозрачных пикселов спрайта. Тогда в этом случае процесс создания спрайта векторным способом должен происходить в соответствии со следующими действиями:

- в качестве фонового выбирается какой-нибудь цвет, не встречающийся в контуре символа, например, яркорозовый, грязно-зеленый и подобные цвета;
- тело спрайта генерируется на прямоугольнике, полностью закрашенном этим цветом;
- после выполнения двух предыдущих условий выбранный цвет назначается прозрачным.

Во избежание искажения конечного кадра при реализации эффекта прозрачности спрайтов с произвольными разноцветными контурами в противовес использованию описанной техники предлагается метод обеспечения прозрачности фона растровых символов [12], в основу которого положен принцип растрового маскирования [9–11], согласно которому каждый кадр символа описывается двумя битовыми образами – масками растровых логических операций *AND* («побитовое И») и *OR* («побитовое ИЛИ») (рис. 1). *OR*-маска – это фактически изображение будущего символа в том виде, в котором он

должен появиться на результирующем кадре. В этом случае в качестве цвета прозрачных пикселов выбран черный (индекс ноль для палитры из 256 цветов – все биты установлены в ноль). *AND*-маска – это бинарный черно-белый силуэт спрайта. Для такой маски черная область совпадает с контуром символа в *OR*-маске, а белые зоны (индекс палитры – 255, все биты установлены в единицу) объединяют прозрачные пикселы.

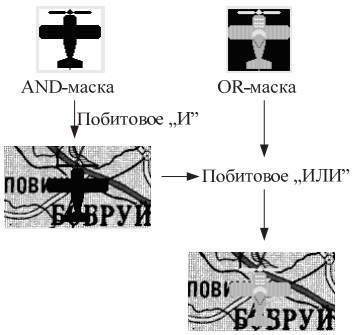

Рис. 1. Процесс создания на картографическом фоне прозрачного спрайта по принципу растрового маскирования

Прозрачность фона изображения символа обеспечивается таким образом благодаря его *OR*- и *AND*-маскам прозрачности:. Процесс создания второй маски можно упростить, если организовать ее формирование на основе первой по определенному правилу – там, где на *OR*-маске обозначен черный цвет, в *AND*-маске должен быть белый, а все остальные цвета *OR*-маски следует заменить на черный. Для перевода *OR*-маски в *AND*-маску создаем процедуру, которая с помощью вложенных операторов цикла *For* сканирует изображение символа *OR*-маски по строкам и столбцам и осуществляет для каждой точки следующее преобразование: если цвет точки черный, то вместо нее необходимо вывести белую точку, иначе – вывести черную.

Основная идея метода заключается в том, чтобы скопировать картографический фон в буфер и перерисовать все маски поверх этого фона. Сначала к канве фонового картографического изображения применяем растровую операцию *AND* с одноименной маской. Эта операция будто вырезает черное отверстие в том месте канвы, где появится спрайт с прозрачным фоном. Затем полученный битовый образ логически складывается с *OR*-маской. Черные области *OR*-маски сохраняют канву результирующего изображения точно так, как и белые на первом шаге.

Для формализации данного процесса введем точечную матрицу растрового символа размером $[w \times h]$ пикселов $\left(a_{xy}\right)_{x=0,y=0}^{w-1,h-1}$. Обозначим через *Spr* полезную часть спрайта, т.е. ту область растрового символа, которая есть непосредственно изображением спрайта с соответствующими цветами пикселов spr_cl_code в координатах (x,y), а через *SprBackgr* – фоновую область спрайта, не подлежащую выводу вместе с *Spr*. Тогда первая маска *MaskOR* определяется битовой матрицей, в которой фон спрайта имеет черный цвет:

$$a_{xy} = \begin{cases} spr\_cl\_code, & (x,y) \in Spr \\ 0, & (x,y) \notin Spr, \quad (x,y) \in SprBackgr \end{cases},$$

а для второй *MaskAND* зададим:

$$b_{xy} = \begin{cases} 0, & (x,y) \in Spr \\ 1, & (x,y) \notin Spr, \quad (x,y) \in SprBackgr \end{cases},$$

в которой фон спрайта имеет белый цвет, а его изображение – черный.

При этом маска прозрачности *MaskAND* должна иметь такие же размеры, что и основной растр *MaskOR*. Аналогичное требование касается также остальных порожденных матриц.

Спрайт с прозрачным фоном выводится на скопированный в промежуточный буфер участок картографического фона *BufMapBackgr* с соответствующими цветами пикселов map_cl_code в координатах (x,y):

$$c_{xy} = \{map\_cl\_code, \quad (x,y) \in BufMapBackgr\}.$$

В результате наложения *MaskAND* на *BufMapBackgr*, с использованием операции «И», последний модифицируется:

$$d_{xy} = b_{xy} \text{ AND } c_{xy} = \begin{cases} 0, & (x,y) \in Spr \\ map\_cl\_code, & (x,y) \in SprBackgr \end{cases},$$

т.е. при выводе белой части *MaskAND* на *BufMapBackgr* в режиме «И» соответствующая область *BufMapBackgr* остается неизменной, а при выводе черной части она закрашивается черным цветом.

В результате выполнения операции «ИЛИ» над *MaskOR* и модифицированным *BufMapBackgr* получаем битовую матрицу:

$$e_{xy} = a_{xy} \text{ OR } d_{xy} = \begin{cases} spr\_cl\_code, & (x,y) \in Spr \\ map\_cl\_code, & (x,y) \in SprBackgr \end{cases},$$

т.е. при выводе черной части *MaskOR* на модифицированный *BufMapBackgr* в режиме «ИЛИ» соответствующая область *BufMapBackgr* остается неизменной, а при выводе цветной части – принимает вид полезного изображения спрайта.

Главное преимущество предложенного метода – обеспечение прозрачности разноцветных растровых символов.

Заключение. Итак, при синтезе динамических сцен для реализации растровой прозрачности символов подвижных объектов удобнее пользоваться предложенным формализованным подходом обеспечения прозрачности фоновых участков таких объектов, не чувствительным к набору цветов в составе контура каждого конкретного символа. Это в свою очередь означает независимость от требования непересечения фонового и принадлежащих телу спрайта цветов.

Окончание на стр. 84

Более того, благодаря формализации задачи приоритетного вывода прозрачных изображений символов на картографический фон удалось выделить классы операций, общие для многих методов формирования динамических сценариев в части достижения эффекта прозрачности, а именно – построение маски прозрачности, ее использование с учетом взаимодействия точек изображений источника и приемника, перенос битов изображения из источника в место назначения (бит-блиттинг). Предварительное выполнение поразрядных логических операций типа *AND* и *OR* с исходными и результирующими пикселами в специально отведенном видеобуфере согласно предлагаемой технике «прозрачных» цветов позволяет также избежать возможного мигания при имплементации процесса блиттинга.

Поступила 09.04.2015
Тел. для справок: +38 044 526-3518 (Киев)
E-mail: *aneesa.qasem@gmail.com*

UDC 004:519.876, 004.93

A.M. Qasem

The Formalization of the Data Handling Process of Raster Masking of Sprites' Transparency

Keywords: dynamic 2D-scene, transparency mask, raster logic operations, sprite, cartographical background

Method is proposed for providing the transparency of the background bitmap pixels of multi-format symbols of moving objects at their outputting to the cartographic background. The essence of the method is in the intermediate buffer to perform the two sequential formalized raster operations with the masks of transparency AND and OR. It ensures the prioritized displaying of transparent sprites in relation to the cartographic background, without the need to comply the requirement which consists in absence the value of transparent background color among the color codes in the image contour of a single symbol.

Due to the formalizing of the task of the display of transparent symbol images on the cartographic background it was succeeded to allocate the classes of operations that are common to many existing methods of forming the dynamic scenarios in terms of achieving the transparency effect. Such operations include the construction of a transparency mask, its use, taking into account the interaction of image pixels of the source and receiver, the transfer of the image bits from the source to the destination (bit-blitting).

Pre-execution of the bitwise logical operations as AND and OR with initial and final pixels into special designated videobuffer according to the proposed technics for transparent colors also allows to avoid the possible flashing during the implementation of the blitting process.

●