

Средства и приемы обработки видео

Дмитрий Ватолин

*Московский Государственный Университет
CS MSU Graphics&Media Lab*


Содержание:

- ◆ **ММХ технология**
- ◆ Программа VirtualDub
- ◆ Программа AviSynth
- ◆ Программа Mathcad

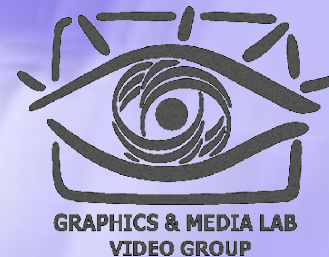


MMX™ Technology

*Потоковая обработка данных.
Средство существенного увеличения
скорости работы видеофильтров.*



Курс по Intel MMX



В слайдах использованы рисунки из курса по MMX компании Intel, который настоятельно рекомендуется пройти.

Курс можно скачать по адресу:

<http://graphics.cs.msu.su/courses/mdc2004/library/mintro.exe>

(размер: 14 МБ)

<http://graphics.cs.msu.su/courses/mdc2004/library/runcbt.exe>

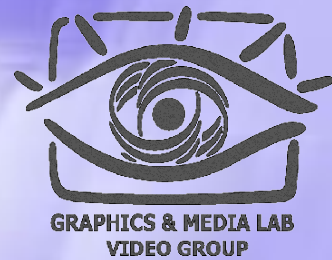
(размер: 2,7 МБ)

История возникновения

Технология MMX была разработана компанией Intel и является своего рода развитием команд процессора.

Технология базируется на архитектуре процессора Pentium® и позволяет **ускорить вычисления за счет параллельной обработки данных.**

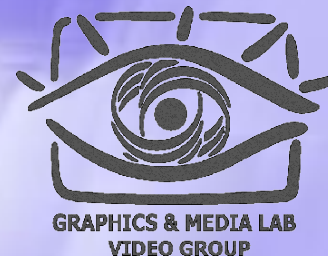
Где применяется технология MMX™ ?



Технология MMX используется во многих мультимедийных приложениях, например **при обработке видео, звука и графики** (ускорение цифровой обработки сигналов и данных).

Забавно, что MMX команды сегодня применяются даже **при заполнении и копировании** буферов операционной системы.

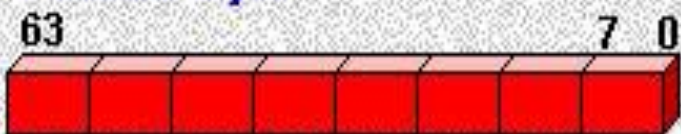
MMX™ технология



1. В технологии MMX применяются инструкции использующие особенности архитектуры нового процессора.
2. MMX инструкции работают как со знаковой так и с беззнаковой арифметикой.
3. Появились 8 новых регистров с соответствующими именами MM0...MM7.

Типы данных в MMX™

Packed Byte



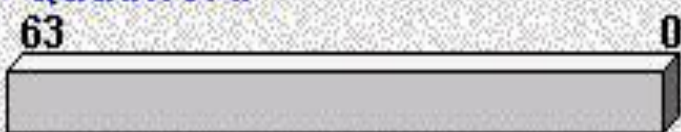
Packed Word



Packed Doubleword

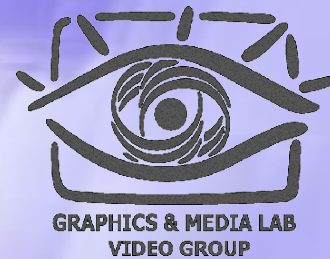


Quadword



В 64 бита можно поместить от 8 «переменных» размером по 8 бит (байтов) и до одной «переменной» размером в 64 бита.

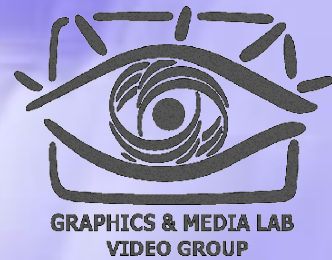
Система команд MMX™



Система команд MMX состоит из 57 команд, сгруппированных в следующие категории:

- Команды передачи данных
- Арифметические команды
- Команды сравнения
- Команды преобразования
- Логические команды
- Команды движка
- Команда освободить MMX™ состояние (EMMS)

Семантика инструкций MMX™

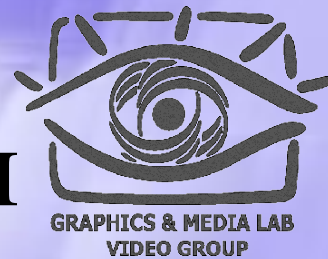


<команда> [dest, src]

<команда> записывается по следующим правилам:

- 1) Команда начинается с “P” (кроме movd, movq)
- 2) “US” работа с без знаковой арифметикой
- 3) “S” или “SS” работа со знаковой арифметикой
- 4) “B”, “W”, “D”, “Q” соответственно обозначают тип с которым работает инструкция

Арифметика с насыщением



MMX технология поддерживает арифметику с насыщением (saturated arithmetics).

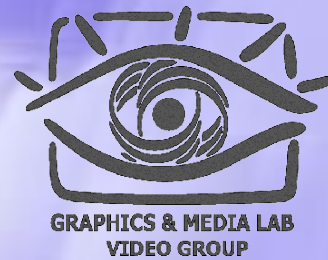
- В режиме с насыщением, результаты операции, которые переполняются сверху или снизу отсекаются к границе *datarange* соответствующего типа данных
- В режиме без насыщения, результаты, которые переполняются как в обычной процессорной арифметике (см. курсы по С и ассемблеру).

Таблица



Тип данных	Нижний предел		Верхний предел	
	Шестн адцат.	Десяти чн.	Шестн адцат.	Десяти чн.
Знаковый байт	80H	-128	7FH	127
Знаковое слово	8000H	-32768	7FFFH	32767
Беззнаковый байт	00H	0	FFH	255
Беззнаковое слово	0000H	0	FFFFH	65535

Команды передачи данных (пример)



MOVD (Переместить 32 Бита) передает 32 бита упакованных данных из памяти в регистры MMX и обратно, или из целочисленных регистров в регистры MMX и обратно.

MOVQ (Переместить 64 Бита) передает 64 бита упакованных данных из памяти в регистры MMX и обратно, или между регистрами MMX.

Пример арифметических инструкций

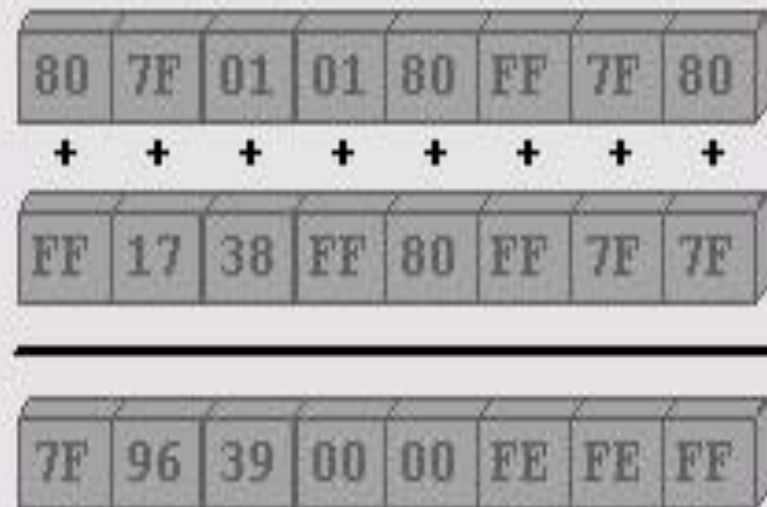


Арифметические	Wraparound	Знаковая	Без знаковая
Сложение	PADD	PADDS	PADDUS
Вычитание	PSUB	PSUBS	PSUBUS
Умножение	PMULL/H		
Умножение и сложение	PMADD		

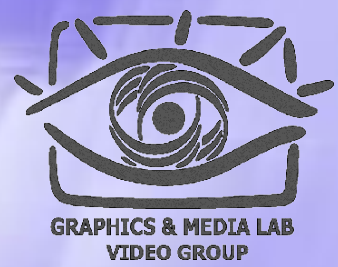
Пример для сложения типа Byte



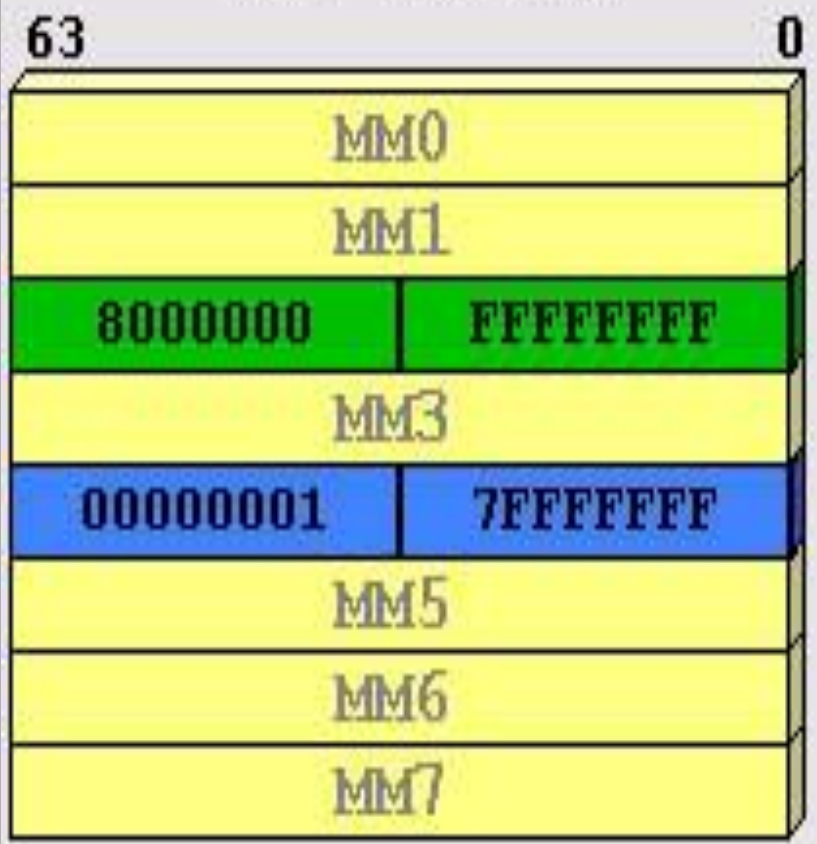
`paddb MM2, MM4`



Пример для сложения типа Word



MMX™ Registers

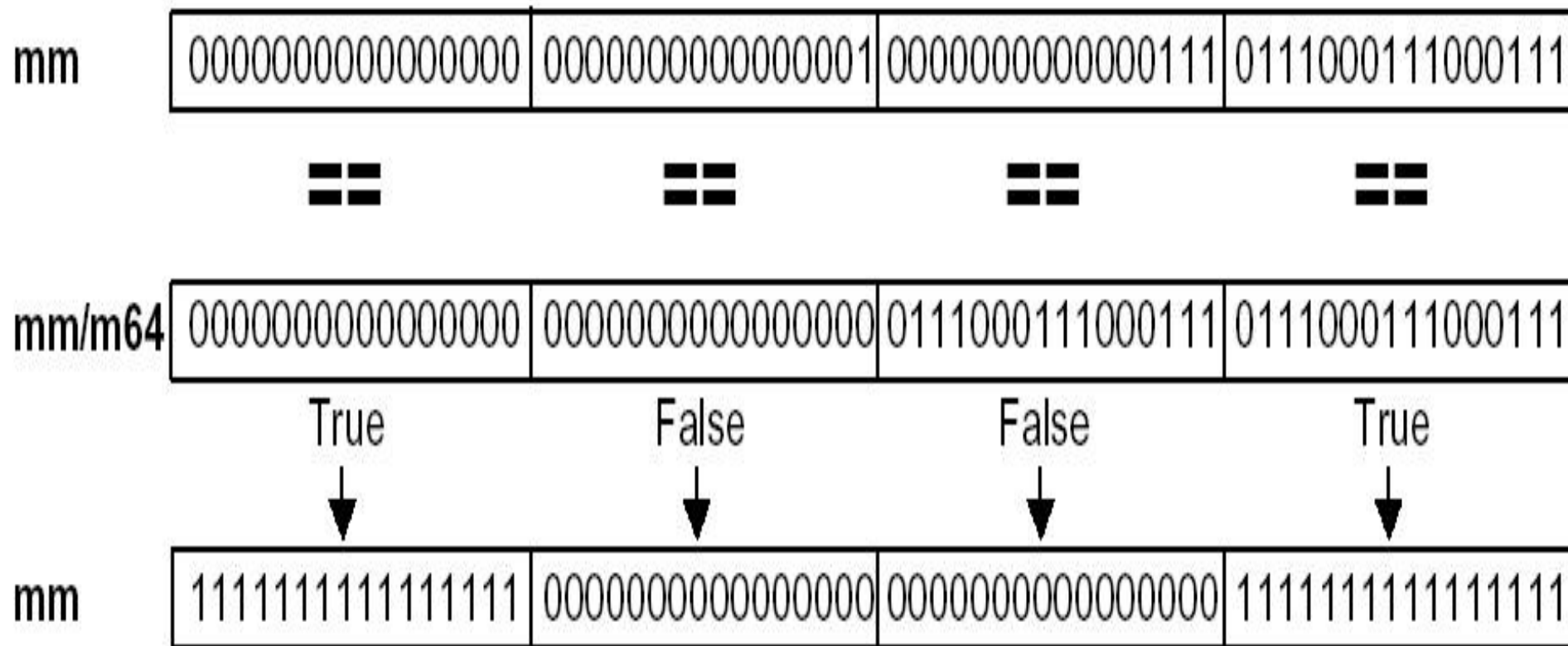


`paddw MM2, MM4`



Пример для сравнения

PCMPEQW mm, mm/m64



Пример кода с MMX™

```
1  movq MM0, [a_vector]
2  movq MM1, [b_vector]
3  pmaddwd MM0, MM1
4  paddd MM7, MM0
5  add [a_vector], 8
6  add [b_vector], 8
7  sub [count], 4
8  jnz loop
9  movq MM0, MM7
10 psrlq MM7, 32
11 paddd MM7, MM0
12 movd mem_vdp, MM7
```

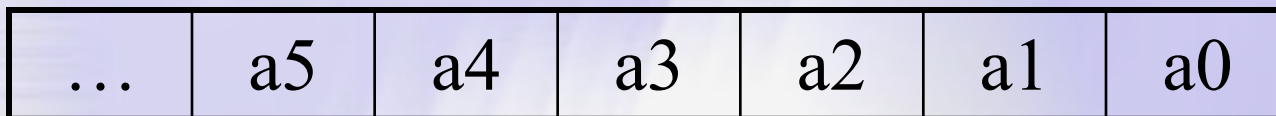
Расчет скалярного произведения:

$$\sum a(i) * b(i)$$

Пример кода с MMX™

```
1 movq MM0, [a_vector]  
2 movq MM1, [b_vector]
```

A_vector



MM0



Пример кода с MMX™

3 pmaddwd MM0, MM1

pmaddwd

MM0

a3	a2	a1	a0
----	----	----	----

MM1

b3	b2	b1	b0
----	----	----	----

MM0

$a3*b3+a2*b2$	$a1*b1+b0*a0$
---------------	---------------

Пример кода с MMX™

4 padd MM7, MM0

MM7

00000000	00000000
----------	----------

padd

MM0

$a_3*b_3+a_2*b_2$	$a_1*b_1+b_0*a_0$
-------------------	-------------------

MM7

$a_3*b_3+a_2*b_2$	$a_1*b_1+b_0*a_0$
-------------------	-------------------

Пример кода с MMX™

```
5 add [a_vector], 8  
6 add [b_vector], 8
```

a_vector



a7	a6	a5	a4	a3
----	----	----	----	----

Пример кода с MMX™



```
7 sub [count], 4
```

Счетчик уменьшаем на 4.
Уже обработано 4 элемента

```
8 jnz loop
```

Продолжается цикл если ещё
осталось что обрабатывать

Пример кода с MMX™

9 movq MM0, MM7

MM0

$a_{11} * b_{11} + a_{10} * b_{10} + a_7 * b_7$ $+ a_6 * b_6 + a_3 * b_3 + a_2 * b_2$	$A_9 * b_9 + a_8 * b_8 + a_5 * b_5 + a_4$ $* b_4 + a_1 * b_1 + a_0 * b_0$
--	--

Пример кода с MMX™

10 psrlq MM7, 32

MM7

$a_{11} * b_{11} + a_{10} * b_{10} + a_7 * b_7$ $+ a_6 * b_6 + a_3 * b_3 + a_2 * b_2$	$a_9 * b_9 + a_8 * b_8 + a_5 * b_5 + a_4 * b_4$ $+ a_1 * b_1 + a_0 * b_0$
--	--

shift

MM7

00000000	$a_{11} * b_{11} + a_{10} * b_{10} + a_7 * b_7$ $+ a_6 * b_6 + a_3 * b_3 + a_2 * b_2$
----------	--

Пример кода с MMX™

11 padd MM7, MM0

MM7

$a_{11} * b_{11} + a_{10} * b_{10} + a_7 * b_7 + a_6 * b_6 + a_3 * b_3 + a_2 * b_2$	$A_9 * b_9 + a_8 * b_8 + a_5 * b_5 + a_4 * b_4 + a_1 * b_1 + a_0 * b_0$
---	---

padd

MM0

00000000	$a_{11} * b_{11} + a_{10} * b_{10} + a_7 * b_7 + a_6 * b_6 + a_3 * b_3 + a_2 * b_2$
----------	---

MM7

$a_{11} * b_{11} + a_{10} * b_{10} + a_7 * b_7 + a_6 * b_6 + a_3 * b_3 + a_2 * b_2$	Наш результат
---	---------------

Содержание:

- ◆ ММХ технология
- ◆ Программа **VirtualDub**
- ◆ Программа AviSynth
- ◆ Программа Mathcad

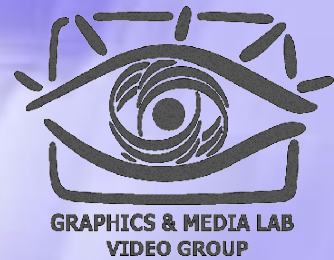


VirtualDub

*Лучшая программа для
работы с потоковым видео*



План



- ✓ О программе VirtualDub
- ✓ Как писать фильтры
- ✓ Пример
- ✓ Итоги

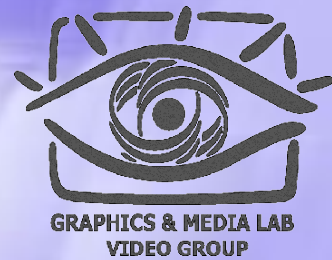
Что это такое?

VirtualDub является бесплатно распространяемой программой.

Это САМАЯ распространенная программа для поточной обработки видео (в т.ч. Подготовки MPEG-4 фильмов с DVD).

У неё открытые исходники, что позволяет модифицировать исходный код программы.

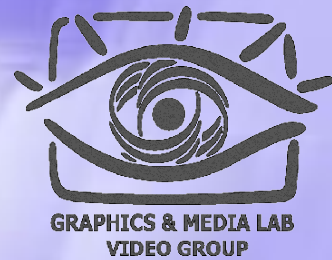
Где можно скачать и узнать о VirtualDub



<http://virtualdub.org/>

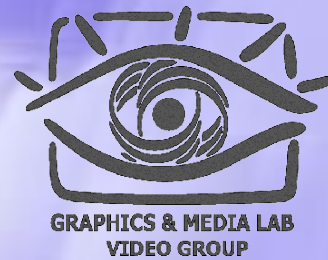
Это официальный сайт VirtualDub. Здесь можно скачать последние версии и документацию по использованию.

Область применения



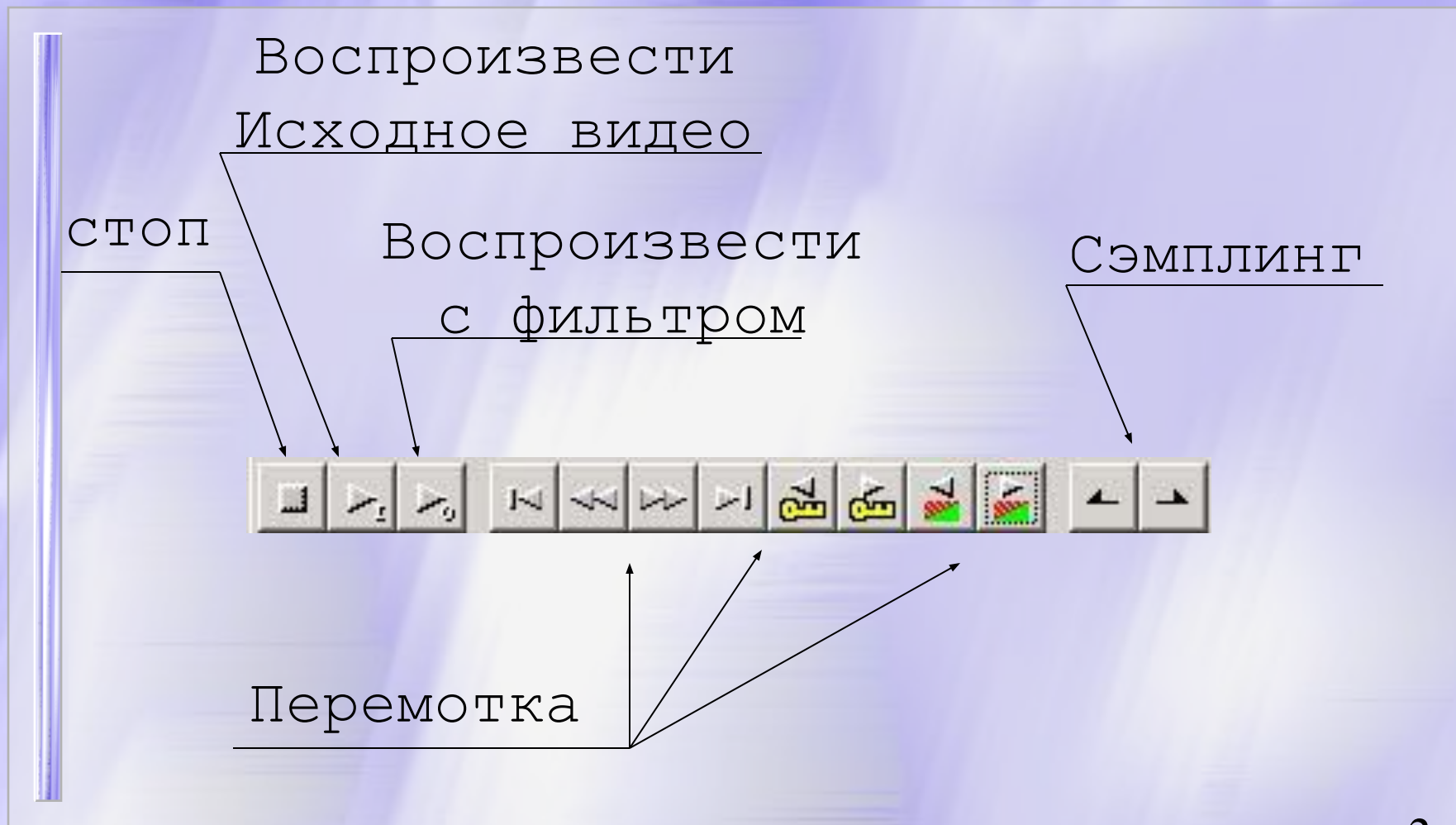
- Осуществляет просмотр и базовое редактирование видео
- Позволяет конвертировать в разные форматы видео и аудио треки
- Обрабатывает видео (и аудио) с использованием фильтров
- Осуществляет восстановление файлов
- Позволяет указывать в скрипте автоматическую обработку фильмов
- Осуществляет качественный захват видео с камеры

Внешний вид VirtualDub

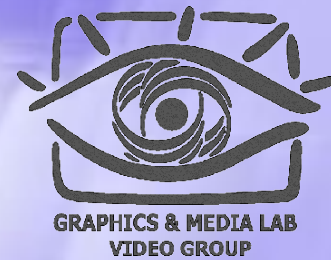


08/20/2023

Панель управления



Меню File

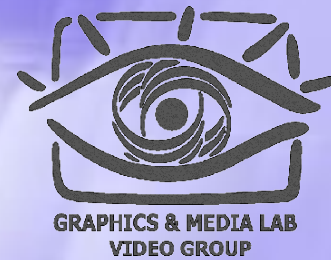


Open video file...	Ctrl+O
Append AVI segment...	
Preview input...	Space
Preview filtered...	Enter
Preview output from start...	F5
Save as AVI...	F7
Save old format AVI...	Shift+F7
Save segmented AVI...	
Close video file	Ctrl+W
File Information...	
Save striped AVI...	
Save stripe master...	
Save image sequence...	
Save WAV...	
Load processing settings...	Ctrl+L
Save processing settings...	Ctrl+S
Start frame server...	
Capture AVI...	
Run script...	
Job control...	F4
1 battle.avi	
2 Alien Ant Farm - Smooth Criminal.avi	
3 1.BMP	
4 015.AVI	
Quit	

Работа с файлами:
сохранение, загрузка,
сохранение скриптов и т.
п.

Работа Job – создание
последовательных
указаний для VirtualDub

Меню Edit



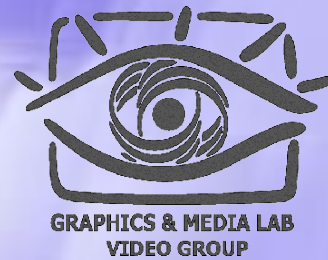
Beginning	Ctrl+Left
End	Ctrl+Right
Previous frame	Left
Next frame	Right
Previous keyframe	Shift+Left
Next keyframe	Shift+Right
Back 50 frames	Alt+Left
Forward 50 frames	Alt+Right
Previous drop frame	{
Next drop frame	}
Previous range	<
Next range	>
Move to selection start	[
Move to selection end]
Go to...	Ctrl-G

Более обширные возможности по перемотки видео вплоть до перехода на указанный номер кадра.

Delete selection	Del
Set selection start	Home
Set selection end	End
Mask selected frames	
Unmask selected frames	
Reset frame subset	

Работа с сэмплингами в расширенном режиме.

Меню Video



Filters...	Ctrl+F
Frame Rate...	Ctrl+R
Color Depth...	Ctrl+D
Compression...	Ctrl+C
Select Range...	

Direct stream copy	
Fast recompress	
Normal recompress	
● Full processing mode	

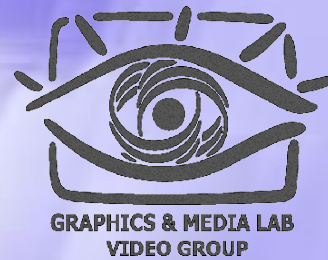
Copy source frame to clipboard	Ctrl+1
Copy output frame to clipboard	Ctrl+2
Scan video stream for errors....	
Error mode...	

Обработка видео с использованием фильтров.

Подключение и указание параметров

Проверка на наличие ошибочных кадров в видео потоке.

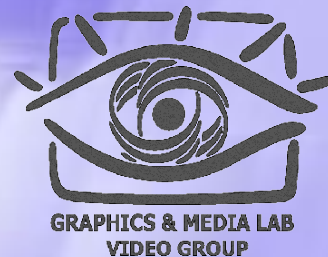
Меню Audio



Установка параметров звука и фильтров которые будут подключены в момент обработки.

- Установка режима:
- Прямое копирование потока
 - Режим полной обработки

Меню Option



Show log...	F8
Show real-time profiler	Shift-F8
Performance...	
Dynamic Compilation...	
Preferences...	
✓ Display input video	F9
✓ Display output video	F10
Display decompressed output	Shift-F10
✓ Show status window	
Swap input/output panes	
Synchronous blit	
Vertical display	
Histograms	
✓ Sync to audio	
Drop frames when behind	
Enable DirectDraw acceleration	
Preview field mode	▶

1. Просмотр Log файлов
2. Установка параметров кодеков
3. Отображение потоков видео
4. Формат отображаемой информации
5. Расположения окон

Как писать фильтры для VirtualDub



Фильтр для VirtualDub представляет собой DLL библиотеку которая имеет вид:

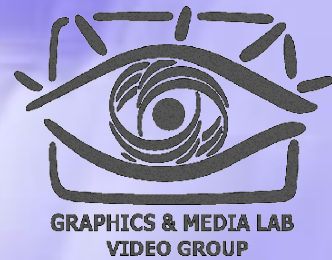
<имя>. vdf

После этого можно скопировать его в папку Plugins и подключить в программе как фильтр.

Структура файла *.vdf

Файл должен содержать минимальный набор функций для работы. Те функции которые используются должны быть описаны в специальной структуре. Если функция не используется, то в поле должно стоять NULL.

Структура



```
typedef struct FilterDefinition{
    . . .
    FilterInitProc      initProc;
    FilterDeinitProc    deinitProc;
    FilterRunProc       runProc;
    FilterParamProc     paramProc;
    FilterConfigProc    configProc;
    FilterStringProc    stringProc;
    FilterStartProc     startProc;
    FilterEndProc       endProc;
    FilterScriptStrProc fssProc;
    . . .
} FilterDefinition;
```

Пример

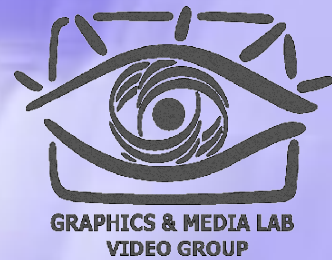
Разберем пример программы которая Blue компоненту уменьшает в два раза, а Green оставляет без изменения.

```
int runProc(const FilterActivation *fa,  
const FilterFunctions *ff);
```

Это аналог процедуры main() в C++, VirtualDub начнет действия с вызова этой функции при обработке кадра (не учитываем вызов интерфейса).

RunProc

Фильтр, уменьшающий Blue в два раза



```
{
    . . .
    src = (Pixel32 *)fa->src.data;
    dst = (Pixel32 *)fa->dst.data;
    h = fa->src.h;
    do {
        w = fa->src.w;
        do {
            old_pixel = *src++;
            new_pixel = (old_pixel & 0xFF0000) +
                ((old_pixel & 0x0000FE)>>1) + 0x008000;
            *dst++ = new_pixel;
        } while(--w);
        src = (Pixel32 *)((char *)src + fa ->
src.modulo);
        dst = (Pixel32 *)((char *)dst +
fa -> dst.modulo);
    }
    while(--h);
    return 0;
}
```


Служебные функции

Для VirtualDub надо включить 2 функции они служебные и не несут большой смысловой нагрузки, но их надо указывать для совместимости с VirtualDub:

```
extern "C" int __cdecl  
    VirtualdubFilterModuleInit2(FilterModule *fm,  
    const FilterFunctions *ff, int& vdfd_ver, int&  
    vdfd_compat);  
extern "C" void __cdecl  
    VirtualdubFilterModuleDeinit(FilterModule *fm,  
    const FilterFunctions *ff);
```

Описание структуры

```
struct FilterDefinition {  
    NULL, NULL, NULL, // next, prev  
    "tutorial", // name  
    "blue color", // desc  
    "anyone", // maker  
    NULL, // private_data  
    0, // inst_data_size  
    . . .  
};
```

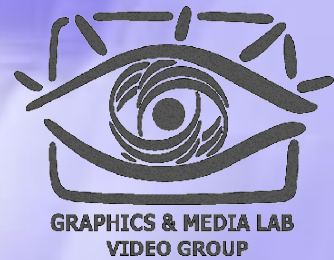
Описание атрибутов относящиеся к интерфейсу фильтра, подсказка для пользователя при подключении фильтра.

Описание структуры

// Описание используемых функций.

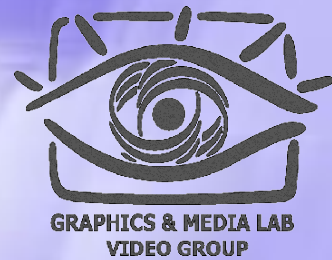
```
. . .  
NULL,          // initProc  
NULL,          // deinitProc  
tutorialRunProc, // runProc  
NULL,          // paramProc  
NULL,          // configProc  
NULL,          // stringProc  
NULL,          // startProc  
NULL,          // endProc  
NULL,          // script_obj  
NULL,          // fssProc };
```

Компиляция



- После компиляции файл будет иметь вид *.dll
- Переименовать файл в *.vdf (мы создавали проект для написания DLL библиотеки)
- Поместить его в папку Plugins
- Подключить в опциях VirtualDub фильтр

Итоги



Плюсы:

- Программа является бесплатной и с открытыми исходниками
- Все проблемы с открытием видео и его сохранением уже решены
- Возможность загрузки фильтров
- Программирование ведется на C++ что позволяет пользоваться весьма гибким аппаратом для реализации своих алгоритмов

Содержание:

- ◆ ММХ технология
- ◆ Программа VirtualDub
- ◆ Программа **AviSynth**
- ◆ Программа Mathcad

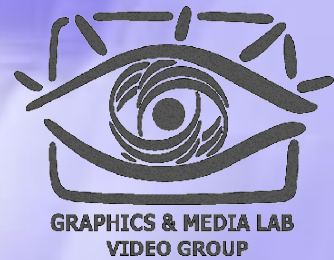


AviSynth

*Скриптовый язык потоковой
обработки видео*



План



- ✓ О программе AviSynth
- ✓ Операторы
- ✓ Семантика и прагматика операций:
 - логические
 - математические
- ✓ Классификаций функций AviSynth
- ✓ Использование Plugins
- ✓ Итоги

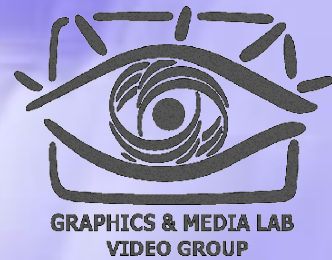
Что это такое?



AviSynth является программой-источником (FrameServer) которую используют различные приложения для обработки видео.

Также AviSynth обладает развитым скриптовым языком и механизмом Plug-In, позволяющим в потоке обрабатывать фильмы.

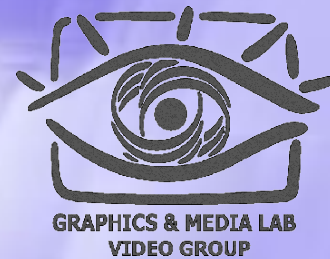
Как работает AviSynth



Использование AviSynth состоит из двух этапов:

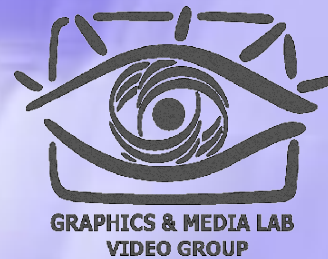
- ◆ создается простой текстовый документ который содержит последовательность команд – скрипт;
- ◆ запускается приложением обрабатывающее видео, например, можно запустить его VirtualDub или Windows Media Player.

Почему удобен AviSynth



AviSynth является открытым и свободно распространяющимся проектом. Исходники можно исправлять и вносить в них те изменения, которые вам нужны. Этот проект только стартовал, и есть уникальная возможность поучаствовать в нём.

Где можно скачать и узнать о AviSynth

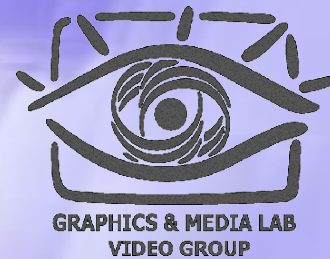


Официальный сайт AviSynth:

<http://www.avisynth.org/>

Здесь можно вносить свои предложения, а также предложена весьма интересная идея –
корректировать сайт вместе с разработчиками.

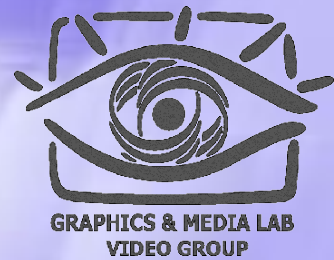
Типы доступные в AviSynth



Семантика ТИПОВ ДАННЫХ	Прагматика
<code>clip</code>	Переменная хранящая параметры видео/аудио клипа
<code>string</code>	Строковая переменная
<code>int</code>	Целочисленная переменная
<code>float</code>	Переменная с плавающей точкой
<code>bool</code>	Булевская переменная

Логические операции

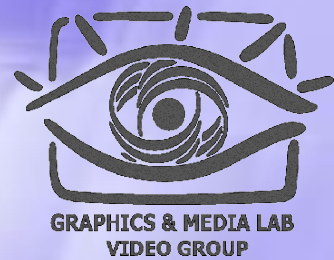
AviSynth



Семантика	Прагматика
==	Равенство
!=	Неравно
	Логическое Или (OR)
&&	Логическое И (AND)

Логические операции

AviSynth



Семантика	Прагматика
+	Сложение
-	Вычитание
*	Умножение
/	Деление
%	Mod (Операция в кольце)
\geq (\leq)	Больше/меньше или равно
$>$	Больше
$<$	Меньше

Пример

`a = 20 - 5 - 5`

Результатом является 10

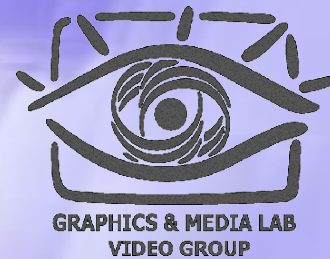
`b = (a==true) ? 1 : 2`

Аналог операции в C++ «:?»

В эквивалентном виде:

```
if (a==true) then b=1 else  
b=2
```


Классификация функций в AviSynth



1. Численные функции
2. Строковые функции
3. Функции перевода
4. Проверочные функции
5. Другие типы функций

Остановимся на некоторых из их подробнее.
Все эти функции а также многие другие
приведены на сайте.

Численные функции

`Floor (float)` Переводит `float` в `int`
до ближайшего снизу

`Floor (1.2) = 1`

`Floor (1.6) = 1`

`Floor (-1.2) = -2`

`Floor (-1.6) = -2`

Численные функции

`Round (float)` Переводит `float` в `int`
округляя результат

`Round (1.2) = 1`

`Round (1.6) = 2`

`Round (-1.2) = -1`

`Round (-1.6) = -2`

Численные функции

Стандартные математические функции:

`Sin (float)`

`Cos (float)`

`Pi ()`

`Log (float)`

`Exp (float)`

`Pow (float base, float power)`

`Sqrt (float)`

Численные функции

```
Spline (float X,  x1,y1,  x2,y2,  . . . . ,  
bool "cubic")
```

Пример:

```
Spline (5,  0,0,  10,10,  20,0,  false) = 5
```

```
Spline (5,  0,0,  10,10,  20,0,  true) = 7
```

Работа со строками

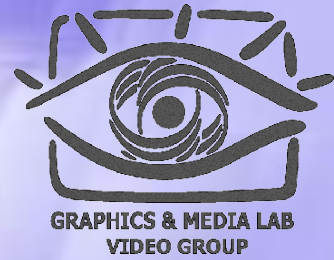
`UCase ("AviSynth")` Буквы в верхний регистр

`LCase ("AviSynth")` Буквы в нижний регистр

`RevStr ("AviSynth")` Инверсия букв

`StrLen ("AviSynth")` Длина строки

Пример



```
UCase ("AviSynth") = "AVISYNTH"
```

```
LCase ("AviSynth") = "avisynth"
```

```
RevStr ("AviSynth") = "htnySivA"
```

```
StrLen ("AviSynth") = 8
```

Функция перевода

`Value (string)` – Переводит строку в `int`

Пример:

`Value ("-2.7") = -2.7`

Функции проверки типа

Функции проверки типа переменных:

`IsBool (var)`

`IsInt (var)`

`IsFloat (var)`

`IsString (var)`

`IsClip (var)`

Пример

```
a = AVISource("d:\capture.00.avi")  
b = AVISource("d:\capture.01.avi")  
c = AVISource("d:\capture.02.avi")  
sound_track=AVSource("d:\audio.wav")  
AudioDub(a+b+c, sound_track)
```

В переменные a,b,c записываются параметры трех видео роликов. Которые потом будут показаны последовательно друг за другом с общей звуковой дорожкой которую мы тоже предварительно загрузили.

Фильтры

В AviSynth можно использовать фильтры написанные уже раньше. Для этого их надо подключить специальной функцией:

```
LoadPlugin ("filename" [, ... ])
```

Фильтры в AviSynth имеют вид:

```
<имя>.avs
```

Фильтры



AviSynth позволяет подключить фильтры написанные в VirtualDub. Единственное ограничение – необходимый формат видео RGB32. Но если будет другой формат можно воспользоваться встроенными в AviSynth функциями для перевода в нужный формат.

Использование фильтров VirtualDub



```
LoadVirtualDubPlugin  
("filename", "filtername", preroll)
```

Подключение фильтра VirtualDub:

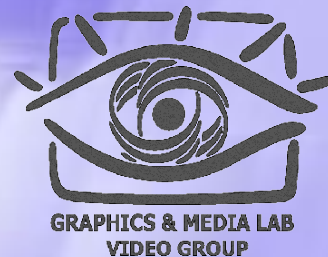
1. Первый параметр <имя>.vdf плагин VirtualDub
2. Второй параметр имя файла с конвертированного по AviSynth (<имя>.avs)
3. Preroll показывает сколько кадров необходимо держать в буфере (например, для деинтерлейсинга)

Пример

```
Import("d:\vdub_filters.avs")  
AviSource("d:\filename.avi")  
ConvertToRGB32() # Там где надо  
VD_SmartBob(1, 0, 10, 1)  
ConvertBackToYUY2() # Там где надо
```

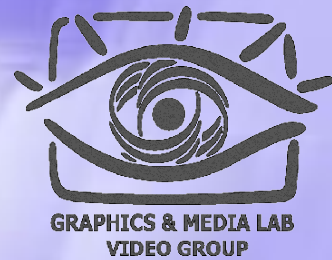
VD_SmartBob(1, 0, 10, 1) – функция осуществляющая подгрузку плагина с соответствующими параметрами для данного фильтра.

Загрузка фильтра деинтерлейсинга



```
function VD_SmartBob(clip 'clip', bool
  'show_motion', int 'threshold', bool
  'motion_map_denoising')
{
  LoadVirtualDubPlugin("d:\bob.vdf", "_VD_Sm
  artBob", 1)
  Return
  clip.SeparateFields._VD_SmartBob(clp.GetP
  arity?1:0,default(show_motion,false)?1:0,
  default(threshold,10),
  default(motion_map_denoising,true)?1:0)
}
```

Итоги



AviSynth является весьма гибким средством применительно к любому приложению работающему с видео. Возможность работы с исходниками позволяет получить при правильном подходе весьма ощутимые результаты.

Основное преимущество – возможность СУЩЕСТВЕННО сэкономить время при массовых операциях с фильмами.

Содержание:


- ◆ ММХ технология
- ◆ Программа VirtualDub
- ◆ Программа AviSynth
- ◆ **Программа Mathcad**



Mathcad

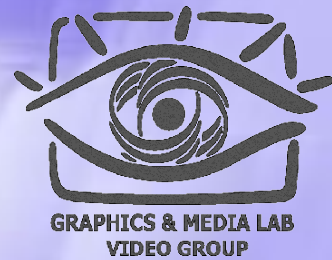
*Удобнейшее средство визуализации
данных.*

*Средство предварительной
проработки фильтров.*



Введение в Mathcad

Достоинства mathcad'a

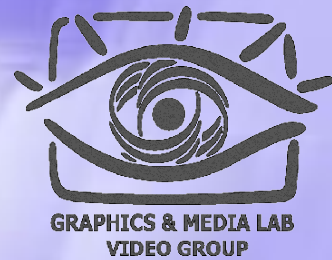


Почему стоит использовать Mathcad:

- Промежуток времени для получения первых результатов работы алгоритма значительно меньше по сравнению с разработкой в какой-либо среде
- Каждое изменение текста программы динамически влияет на результат
- Поиск ошибок осуществляется быстрее, чем в исходном тексте программы на к-л. языке программирования
- Реализовав основную часть алгоритма в mathcad'e, время написания реальной программы уменьшается на порядок
- **Множество** реализованных, готовых к использованию функций

Введение в Mathcad (2)

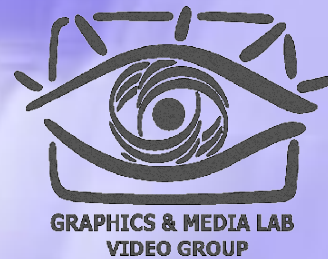
Достоинства mathcad'a



Почему стоит использовать Mathcad:

- Исходный код программы выводится в графическом режиме, и потому выглядит нагляднее, чем в текстовом редакторе
- Как правило, для реализации к-л. задачи в mathcad'e требуется написать меньше исходного теста, чем например в C++
- Реализовав основную часть алгоритма в mathcad'e, время написания реальной программы уменьшается на порядок
- Отличная помощь: все описано кратко и понятно
- Простота использования
- Индексация в массиве начинается с нуля

Пример функции чтения Сбоку - изображение с NEDI



```
LoadC2(A, i, j, M) := | m ← M - 1  
                      | str ← 0  
                      | for d ∈ 0..m  
                      |   i0 ← i + d  
                      |   j0 ← j - m + d  
                      |   for k ∈ 0..m  
                      |     Cstr,0 ← Ai0-k, j0+k-2  
                      |     Cstr,1 ← Ai0-k-2, j0+k  
                      |     Cstr,2 ← Ai0-k+2, j0+k  
                      |     Cstr,3 ← Ai0-k, j0+k+2  
                      |     str ← str + 1  
                      | C
```



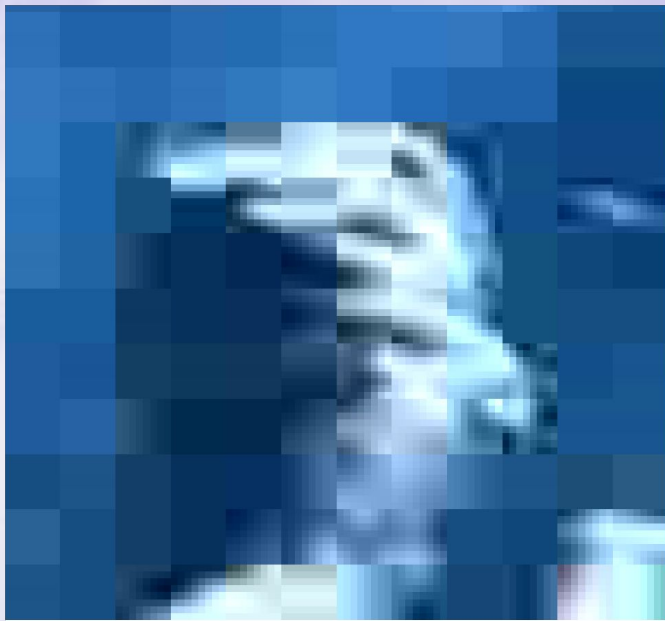
R2, G2, B2

Разработка фильтра

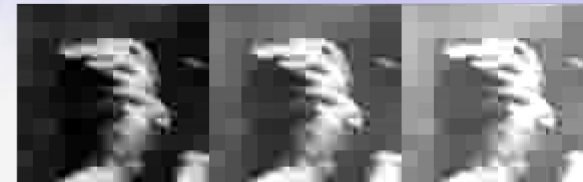
Чтение изображения

Визуализация матрицы – Ctrl + T

```
S := "F:\Doklad\battle_100_59_blocked.bmp " P := READRGB(S)
```



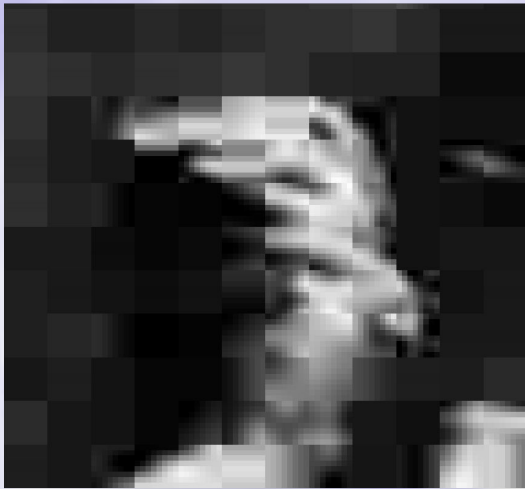
s


$$\text{DeTriplex}(P)^P := \left\{ \begin{array}{l} w \leftarrow \frac{\text{cols}(P)}{3} \\ R_0 \leftarrow \text{submatrix}(P, 0, \text{rows}(P) - 1, 0, w - 1) \\ R_1 \leftarrow \text{submatrix}(P, 0, \text{rows}(P) - 1, w, 2 \cdot w - 1) \\ R_2 \leftarrow \text{submatrix}(P, 0, \text{rows}(P) - 1, 2 \cdot w, 3 \cdot w - 1) \\ R \end{array} \right.$$

Разработка фильтра

Показ компонент изображения

$RGB := \text{DeTriplex}(P)$



RGB_0



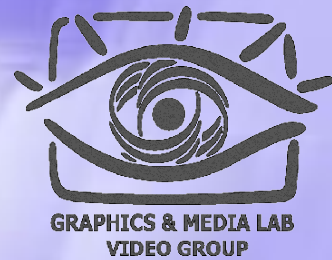
RGB_1



RGB_2

Разработка фильтра

Перевод изображения в YUV

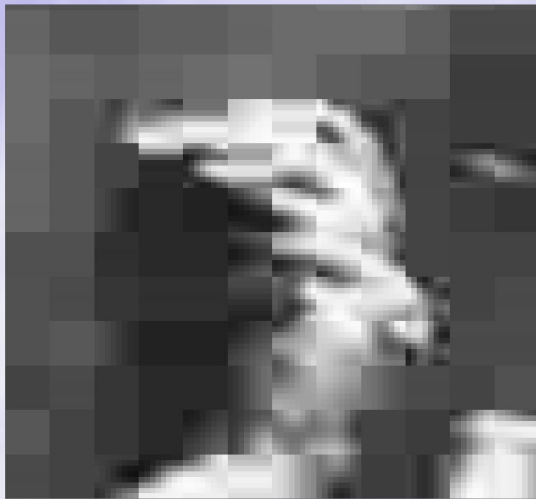


```
RGB_to_YUV(RGB) := | R ← RGB0  
                    | G ← RGB1  
                    | B ← RGB2  
                    | YUV0 ← Clip(0.299 · R + 0.587 · G + 0.114 · B)  
                    | YUV1 ← Clip(-0.147 · R - 0.289 · G + 0.436 · B + 128)  
                    | YUV2 ← Clip(0.615 · R - 0.515 · G - 0.1 · B + 128)  
                    | YUV
```

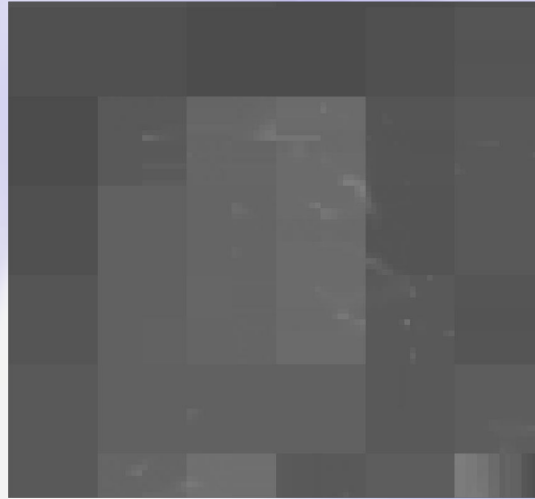

Разработка фильтра

Показ результата

$YUV := RGB_to_YUV(RGB)$



YUV_0



YUV_2



YUV_1