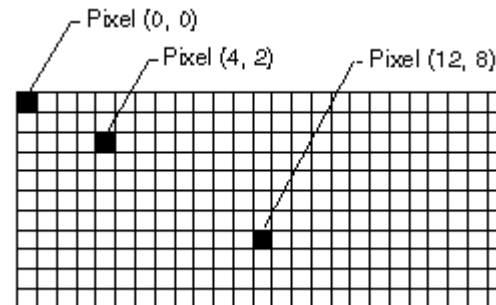
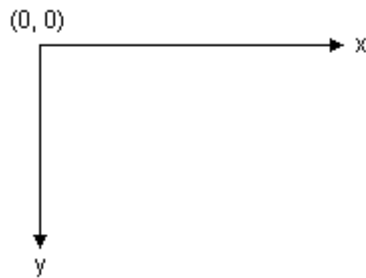


Лекция 13.

Работа с графикой

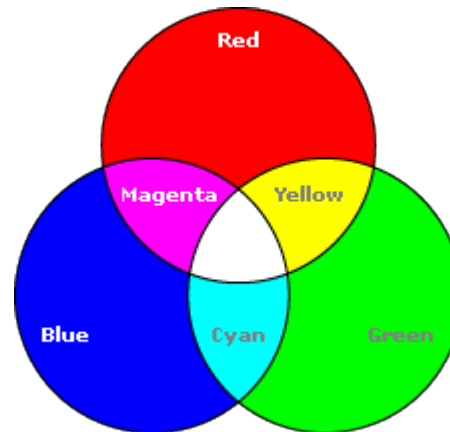
Обработка графики

- По сути, изображение представляет собой двумерный массив пикселей (то есть сетку с двумя координатами — x и y)
- Оси идут от верхнего левого угла направо и вниз

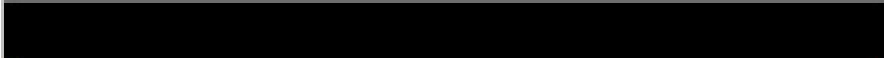


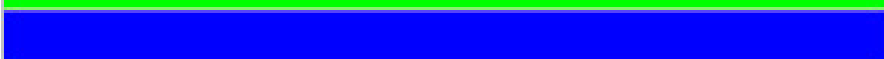







Обработка графики

- Цвет пикселя задается тремя компонентами – **красная (Red)**, **зеленая (Green)** и **синяя (Blue)** – **RGB**
- Интенсивность каждой компоненты обозначается целым числом от 0 до 255
- Используя разные комбинации **RGB** с разной интенсивностью компонент можно получить любой цвет



Некоторые цвета

Color	Color HEX	Color RGB
	#000000	rgb(0,0,0)
	#FF0000	rgb(255,0,0)
	#00FF00	rgb(0,255,0)
	#0000FF	rgb(0,0,255)
	#FFFF00	rgb(255,255,0)
	#00FFFF	rgb(0,255,255)
	#FF00FF	rgb(255,0,255)
	#C0C0C0	rgb(192,192,192)
	#FFFFFF	rgb(255,255,255)

Инвертирование цветов

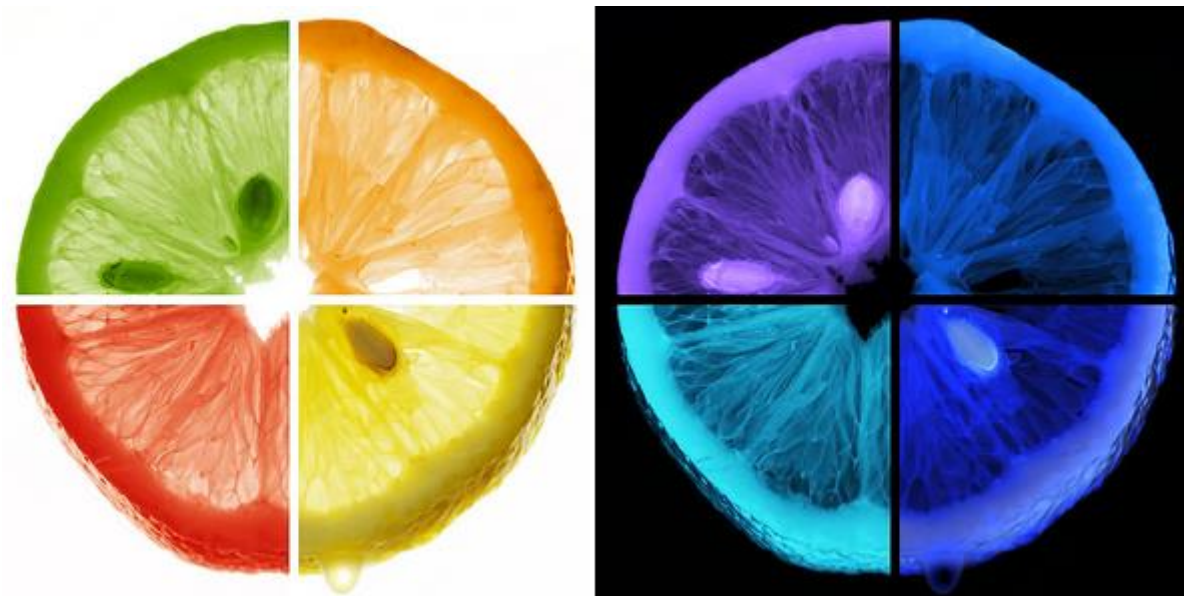
- Нужно просто заменить каждую компоненту пикселя на 255 минус текущая интенсивность
- Обозначим это так:

$$r_{result} = 255 - r$$

$$g_{result} = 255 - g$$

$$b_{result} = 255 - b$$

Инвертирование цветов



Перевод в черно-белое

- Серый цвет получается когда у пикселя все три компоненты равны между собой: $r = g = b$
- Но при этом в картинке у каждого пикселя это число может быть своим
- При переводе в черно-белое, должно соблюдаться
$$r_{result} = g_{result} = b_{result} = 0.3r + 0.59g + 0.11b$$
- Коэффициенты перед компонентами не равны, т.к. человеческий глаз воспринимает интенсивность этих цветов по-разному

Задача на дом «Перевод в черно-белое»

- Возьмите за основу программу **ImageTest** (выложена в документах группы)
- Реализовать перевод в черно-белое

Попиксельные операции

- **Функция насыщения**

(обработка выхода за диапазон от 0 до 255):

$$sat(x) = \begin{cases} 0, & x < 0 \\ x, & x \in [0, 255] \\ 255, & x > 255 \end{cases}$$

- **Гамма-коррекция**

$$r_{result} = sat(r^\gamma); \quad g \text{ и } b \text{ — аналогично}$$

γ — некоторое вещественное число

- Результат округляем при помощи **Math.Round**

Изменение яркости

- $$sat(x) = \begin{cases} 0, x < 0 \\ x, x \in [0, 255] \\ 255, x > 255 \end{cases}$$
- **Изменение яркости**
- $r_{result} = sat(r + x)$, x – некоторое число
- Аналогично для g и b компонент

- $$sat(x) = \begin{cases} 0, & x < 0 \\ x, & x \in [0, 255] \\ 255, & x > 255 \end{cases}$$
- **Контраст**
- $r_{result} = sat(x(r - 127.5) + 127.5),$
 x – некоторое число
- Аналогично для g и b компонент

Сглаживание (размытие)

- $$H = \begin{pmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{pmatrix}$$
- Представим, что мы смотрим на некоторый пиксель, сопоставим ему центральный элемент этой таблицы
- Вокруг него есть 8 смежных с ним пикселей
- В итоге образуется область из пикселей размера 3x3
- Верхнего левого соседа сопоставим с верхним левым числом этой таблицы, верхнего центрального соседа – с верхним центральным числом и т.д.

Сглаживание (размытие)

- $$H = \begin{pmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{pmatrix}$$
- Тогда итоговая красная компонента для центрального пикселя вычисляется как сумма следующих слагаемых:
 - Красная компонента верхнего левого пикселя умноженная на верхнее левое число таблицы
 - Красная компонента верхнего центрального пикселя умноженная на верхнее центральное число таблицы
 - И т.д., всего 9 слагаемых
- Аналогично для синей и зеленой компонент

Граничные пиксели

- Понятно, что у пикселей, которые являются крайними в изображении, нет некоторых соседей
- Такие пиксели мы рассматривать не будем, будем обрабатывать, отступив на 1 пиксель от каждой границы

Исходная и результирующая картинка

- Если мы поменяем цвет одного пикселя, то при подсчете цветов соседних пикселей будет использоваться уже новое значение
- Это неверно, и должно использоваться старое значение
- Поэтому нам понадобится 2 картинки – исходная и результирующая
- Читать всегда надо из исходной, а записывать – в результирующую

Другие операции

- На том же принципе, что и сглаживание, реализовано большое количество эффектов
- Только там другие коэффициенты в таблице
- Например, увеличение резкости:
- $$H = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$
- Сам этот алгоритм (когда проходимся по области, перемножаем на коэффициенты и суммируем) называется **сверткой**

Задача на курс «Размытие»

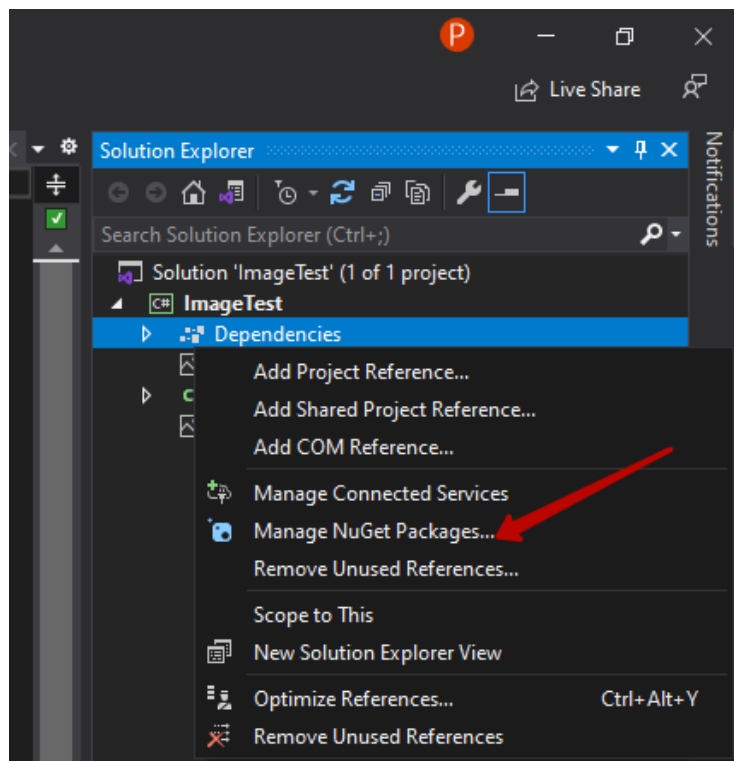
- Реализовать размытие
- По задаче требуется использовать матрицу коэффициентов, чтобы поменяв числа, например, можно было получить эффект увеличения резкости
- Если вариант с матрицей сразу сделать сложно, сначала сделайте размытие без матрицы

Программа ImageTest

- В этой лекции в качестве программы–заготовки будем использовать программу **ImageTest**
- В ней открывается файл с картинкой, из него грузятся данные, затем мы меняем значения пикселей, и потом сохраняем результат в новый файл
- Для работы с картинками используем класс **Bitmap**, который находится в библиотеке **System.Drawing.Common**
- Эта библиотека не подключена по умолчанию к проекту

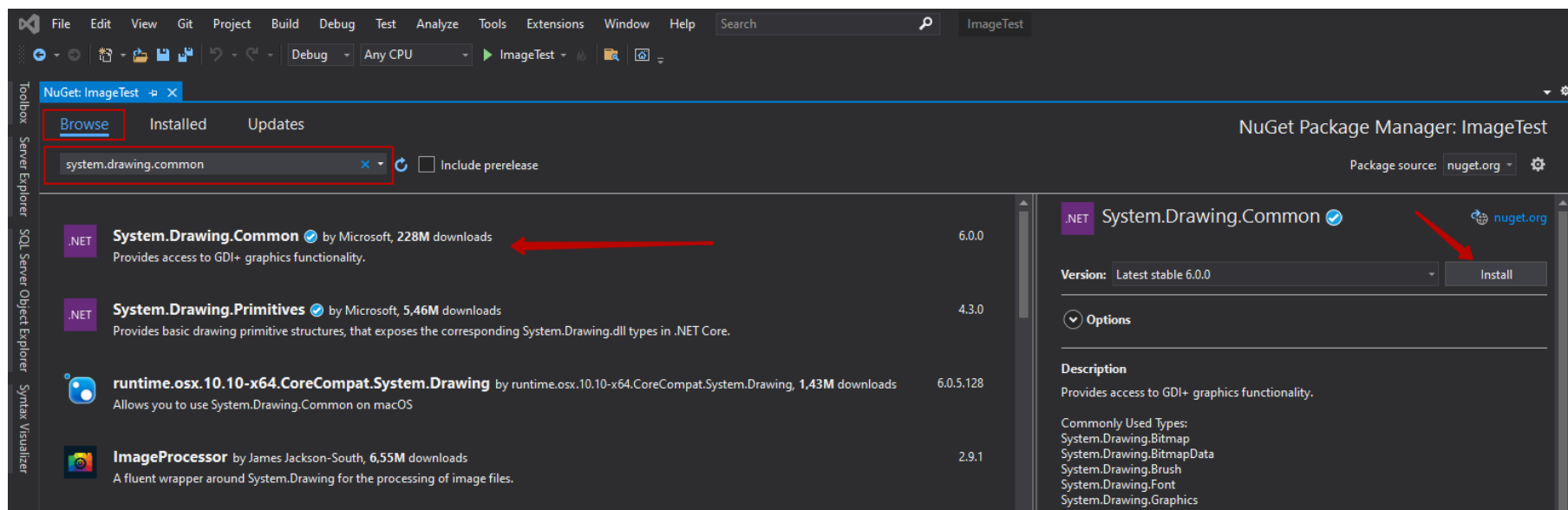
Добавление ссылки на библиотеку

- По умолчанию к проекту не подключена библиотека **System.Drawing.Common**
- Подключить её можно через пункт **Manage NuGet Packages...** контекстного меню пункта **Dependencies**



Добавление ссылки на библиотеку

- Во вкладке **Browse** ищем **System.Drawing.Common**
- Выбираем эту библиотеку и нажимаем **Install**



Программа ImageTest

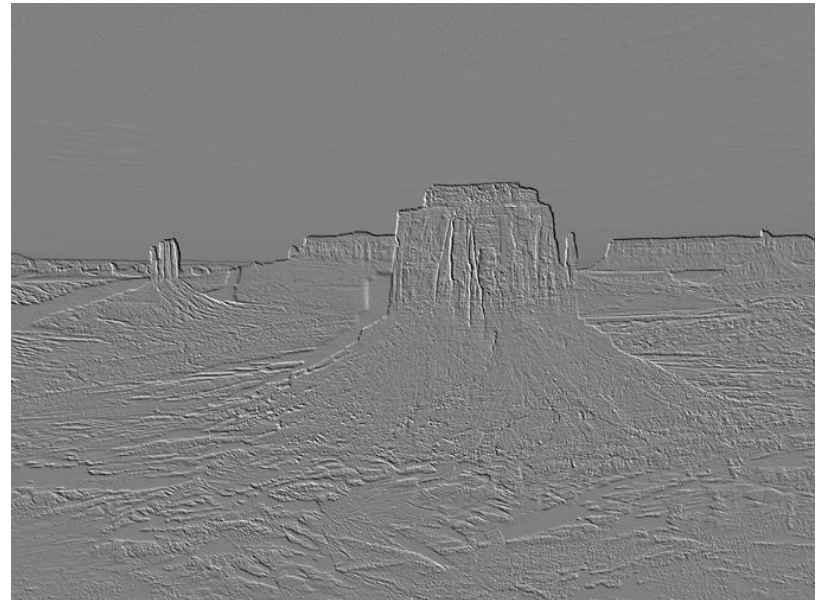
- Один из конструкторов класса `Bitmap` принимает путь к файлу с картинкой
- При запуске из среды разработки нужно указывать пути относительно папки, откуда запускается скомпилированный код – относительно папки **bin\Debug\net5.0**
 - Последняя папка зависит от версии .NET
- Поэтому путь к входному файлу получится такой:
 - `“..\..\..\image.jpg”`

Тиснение

- Тиснение тоже реализуется через свертку
- Сначала нужно перевести картинку в черно-белые цвета
- Затем применить свертку:

- $$H = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}$$

- К каждому результирующему цвету нужно добавить 128
- Тогда фон получится серым, а границы будут более светлыми
- Числа в матрице можно менять, но сумма коэффициентов должна быть 0



Акварелизация

- Картинка после обработки выглядит так, будто она была нарисована акварелью



Акварелизация

- В алгоритме 2 этапа:
 1. Для каждого пикселя сделать следующее:
 - Взять красную компоненту пикселя и красные компоненты соседей из области 5x5 вокруг пикселя, положить все эти числа в массив
 - Отсортировать массив
 - В качестве результата взять элемент по среднему индексу (12)
 - Аналогично сделать для зеленой и синей компонент
 2. Применить эффект повышения резкости

Тиснение и акварелизация

- В курс эти задачи не входят, можете реализовать их для себя в свободное время