С Т А Т Ь И

Пыльский Александр
pylski@yandex.ru
Украина, Киев, 2004

Треппинг и оверпринт

Эта статья предназначена для подготовленного читателя. Для тех, кто сталкивался с совмещением красок при цветной печати, и видел, к чему приводят ошибки несовмещения, т.н. **misregistration [1]**. И для тех, кто интересуется способами решения этой проблемы **[2]**.

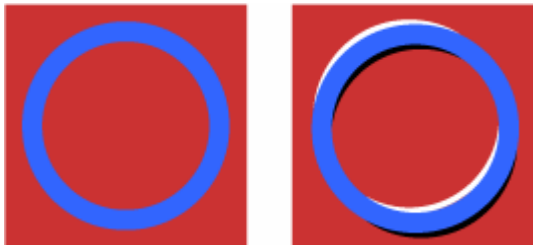

Рис 1. Несовмещение при печати.

О понятиях и терминах

Есть два способа решения проблемы misregistration, простой – **overprint** и сложный – **trapping** (будем применять эти термины в оригинальном, английском написании). Первый способ заключается в управлении режимом наложения красок друг на друга. Второй способ основан на применении специальных объектов – ловушек для misregistration. Оба способа тесно связаны друг с другом (их порой даже путают) и обычно применяются совместно.

Начнем с простого примера. Мы хотим напечатать пару объектов разного цвета. Один объект поверх другого. Что при этом происходит? Верхний объект может вырубить в нижнем «дырку» – т.е. сделать **knockout**. Наша вырубка – это специально созданный третий элемент, проекция верхнего объекта на нижний. Как только в нижнем объекте появилась вырубка, немедленно появилась и проблема – при печати нужно верхним объектом точно попасть в дырку нижнего. Совместить их. А всегда ли эта вырубка необходима?

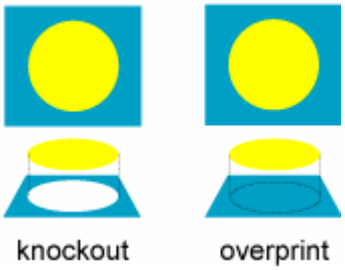

Рис 2. Два варианта печати.

Часть 1. Overprint.

В некоторых случаях мы можем запретить объекту создание вырубки под собой. Для этого и служит специальный атрибут – **overprint**. Объект, имеющий атрибут **overprint**, не делает под собой вырубки в тех цветах, которых он не содержит. Вследствие чего совмещать объект и вырубку не придется (за отсутствием таковой). Нет «дырки» – нет проблем с совмещением.

Когда можно и когда нельзя применять **overprint**? Давайте подумаем – а зачем вообще нам была нужна вырубка? Избежать наложения красок. А зачем? Избежать цветовых сдвигов. Обычные краски ведь прозрачны. Если желтый объект лежит поверх голубого фона, и без вырубки в голубом, то в наложении будет сумма двух цветов, объект станет зеленым (рис 3). Еще хуже, когда фоном идет пестрая картинка (растровая или векторная – не важно). Картинка будет откровенно просвечивать под объектом. А если сделать **overprint** белому объекту? Он просто исчезнет. Поэтому обычно объекты в издательских программах идут с атрибутом **knockout** (делают вырубки).

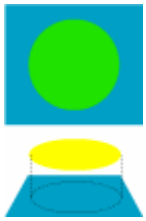

Рис 3. Цветовые сдвиги при наложении.

Overprint Black

В каких же случаях **overprint** возможен? Когда цветовые сдвиги не опасны. Когда они незаметны. К примеру, если объект черный и мелкий (рис 4). Если черный текст восьмого кегля идет поверх голубого фона с атрибутом **overprint**, то цветовой сдвиг в наложении K+C не существенен. Сдвиг есть (текст немного потемнеет), но он практически незаметен. Но если мы забудем про **overprint**, любой глаз заметит малейшее несовмещение между текстом и вырубкой под него в голубом (левый рисунок). Если говорить в общем, то суть выбора проста: нужно решить – что заметнее, дефекты несовмещения или цветовые сдвиги и «просвечивание». Для черного заметность **misregistration** на порядок выше, чем сдвиги. Поэтому обычной практикой в препрессе стало пускать черный в наложение – **all black overprint**.

Рис 4. Черный - **knockout** и **overprint**.

Но здесь кроется опасность.

Только в мелких объектах черный позволяет пренебречь цветовыми сдвигами и «просвечиванием». Подчеркиваю – только в мелких объектах. Офсетный черный – краска хоть и темная, но не абсолютно черная. И при этом достаточно прозрачная. Если вы сделаете **overprint** черной плашке размером 5x5 см и идущей поверх картинки, то картинка будет проглядывать под черным. Если ваша линия имеет толщину 5 мм, не стоит ей делать **overprint**, т.к. объекты, лежащие под ней, будут легко читаться глазом. Аналогично и с текстом большого кегля (рис 5). Итак, обычная практика препресс **«всегда overprint черного»** может и подвести. При расстановке атрибута **overprint** для черного требуется интеллект:
«текст кегля не более ...», «stroke толщины не более ...», «fill размером не более ...».

Рис 5. При большом кегле **overprint** станет заметен.

Атрибут **overprint** может быть расставлен непосредственно в макете, а может и в растровом процессоре репроцентра. Где удобнее? Ответ очевиден – там, где сидит **специалист**. Человек (или «искусственный интеллект»), знающий вышеупомянутые простейшие правила. С обеих сторон есть опасность нажать «ленивую» кнопку **all black overprint**. С другой стороны следует помнить – **overprint**, даже качественный, не заменяет собой **trapping**. Если вы не в состоянии сделать (заказать) **trapping** – старайтесь использовать более грубые настройки для **overprint**, даже в ущерб просвечиванию. В полноценной (**overprint+trapping**) цепочке я рекомендую следующие установки для **black overprint**:

- **текст** не более 12 кегля;

- **stroke** и **fill** толщиной не более 1 мм (≈3 pt).

Если же продукция невысокого класса и **trapping** выполняться не будет, то стоит увеличить эти числа:

- **текст** не более 24 кегля;
- **stroke** и **fill** толщиной не более 2 мм (≈6 pt).

Возможны и более высокие значения, но это уже зависит от изображения, решать нужно творчески. Следует только понимать, что заметность **misregistration** значительно опаснее, чем просвечивание картинки под черной плашкой. Иногда лучше оставить **all black overprint** – будет «третий сорт не брак».

Кроющие краски

В каких еще случаях цветовые сдвиги в наложении не опасны? Рассмотрим вариант применения непрозрачных красок – т.н. кроющих (**opaque**). В отличие от офсетных (прозрачных), кроющие краски не просвечивают. Если в нашем примере (рис 2) при печати применялась специальная кроющая желтая краска (и она печаталась после голубой), то при их наложении суммы цветов не возникнет. Желтый объект останется желтым. Здесь, в отличие от обычных красок, крайне важен порядок – желтый должен печататься после голубого.

Кроющими могут быть не только краски. Жесть – прекрасный пример кроющего материала. Если всерьез, то кроющими в полиграфии считаются фольги, используемые при тиснении, многие трафаретные краски. Бывают и кроющие офсетные краски, к примеру серебро, бронза и другие (так называемые «металлики»). Кроющие, но с замечанием. Плоский офсет, увы, не в состоянии нанести достаточно толстый слой краски. А при толщине краскослоя 1-2 мкм офсетное серебро чуть просвечивает, поэтому такие краски лучше назвать полукроющие (**semi-opaque**)**[3]**. Нередко кроющие краски смешивают с другими, прозрачными. Или осветляют («разбеливают»), применяя прозрачные лаки или же пуская краски в растр. В этих случаях кроющая способность такой краски падает, и применение **overprint** может привести к ошибкам.

Применение атрибута **overprint** для кроющих красок в основном определяется порядком их следования (**print order**)**[4]**, но также может зависеть и от других технологических нюансов и ограничений. Все их раскрыть невозможно. Только один пример – при тиснении вышеупомянутой фольгой ошибки совмещения весьма большие. Фольга - хороший кроющий элемент, поэтому рисунок клише в макете обычно делают **overprint**. Но нужно не забывать про изображение под фольгой (точнее, про толщину краскослоя). Если штамп крупный, и ложится поверх темного рисунка, то краски под штампом окажется много. При этом краска может оказаться недостаточно просушена. В таком случае фольга при тиснении или отказывается переноситься вовсе, или пузырится. Как видим, ситуация схожа с **overprint black**, хотя причины совершенно разные. Здесь также требуется принять решение – когда можно обойтись **overprint**, а когда применить **trapping** (см. ниже).

Практика применения.

Нередко в макетах присутствуют не только сами изображения, но и контура высечек/биговок. Такому контуру назначаем **overprint** однозначно. Аналогично и с формами для выборочной лакировки, и с изображением клише для «слепого» тиснения. Еще один пример – кроющая белая подложка при печати на цветных или прозрачных материалах (рис 6). Такая подложка должна быть окрашена специально назначенным цветом (spot color или «пантон»), встроенный во все издательские программы white здесь не подойдет.

Рис 6. Кроющий белый на прозрачной подложке.

Несколько замечаний для препрессора, занимающегося выводом. К сожалению, средствами растрового процессора полностью автоматизировать процесс управления атрибутом **overprint** пока невозможно. Некоторые растровые процессоры имеют возможности анализа объектов при расстановке **overprint** для черного (рис 7 и 8). К сожалению, возможности далеко не полноценные. Кроме того, **overprint** бывает нужно задавать не только черному. Сделать это автоматически для цвета типа **«visechka»** (или подобным) могут только растровые процессоры фирмы **Creo-Scitex**. В остальных случаях атрибут **overprint** придется расставлять вручную в приложении или в pdf.

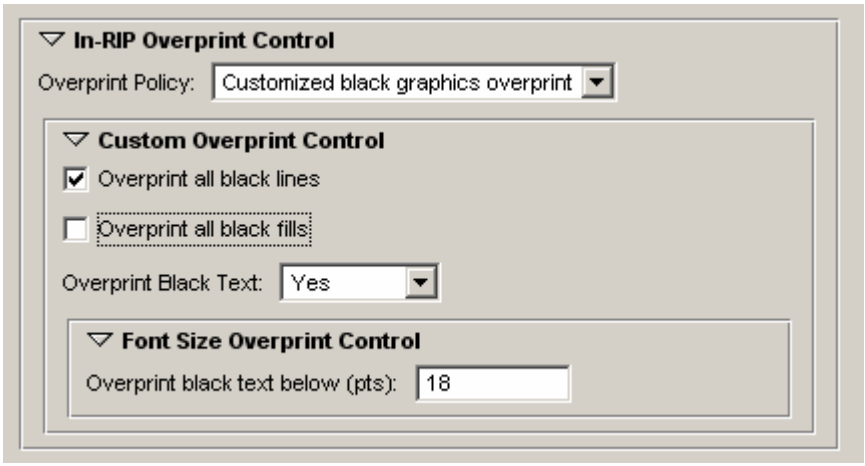

Рис 7. **Black overprint** в процессоре **AGFA Apogee**.

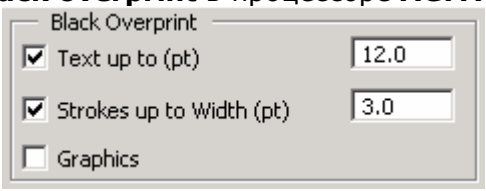

Рис 8. **Heidelberg Supertrap 4.0**.

Помните про контроль за **overprint** белого. Многие (наши любимые) издательские программы могут сделать **overprint** такому цвету, как **black 0% tint**. Т.е. белому. Любой **preflight** обязан проверять – **always white knockout**. Естественно, исключая ситуацию с кроющим белым.

Overprint текста в **QuarkXpress** требует особого внимания. Эта программа норовит обязательно поставить атрибут **overprint** любому (!) объекту, который лежит поверх белого (**white underneath color**). Иногда этот рискованный метод дает сбой. В случае «текст по тексту» (рис 9) красный текст, лежащий поверх черного, ошибочно получил атрибут **overprint**.

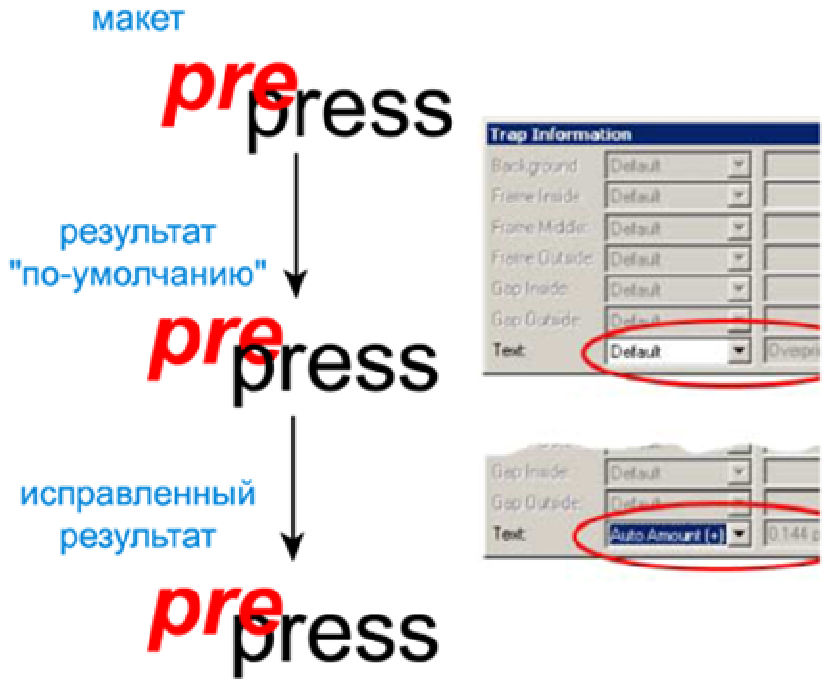

Рис 9. Ошибка «текст по тексту» в **QuarkXPress**.

Особо подчеркну – не забывайте следить за **overprint** при переделке макета, особенно при перекрашивании.

Далі буде.

---

[1] Повышенная заметность ложных контуров **misregistration** связана с тем, что человеческий глаз (точнее мозг) обучен с детства в любой картинке в первую очередь изыскивать информацию. Даже там, где ее не должно быть. Дефекты несовмещения образуют заметные паразитные элементы (артефакты), ухудшающие общее восприятие изображения. Особую чувствительность глаз имеет к яркостным перепадам, цветовые же дефекты значительно менее заметны. На подмене яркостных артефактов на цветовые и основаны наши приемы маскирования **overprint** и **trapping**.

[2] Даже идеальная машина не может совместить «в ноль». Причиной тому деформации бумаги, возникающие при прохождении ее сквозь печатную машину. На практике уточнить ожидаемое несовмещение лучше у технолога типографии. Обычно нормальным несовмещением считается половина линии растра. К примеру, для линиатуры 175 lpi это 70 мкм. Требовать от печатника идеального совмещения бессмысленно, только бумагу испортите в попытках приводки «в ноль».

[3] Любые краски, вне зависимости от того, офсетные они, трафаретные или другого типа, имеют разную прозрачность (точнее, кроющую способность). Ее уровень описывает специальный параметр – **opacity**. Часто, дабы не усложнять тему, краски разделяют на четыре типа – прозрачные, полупрозрачные, полукроющие, кроющие. Обычные триадные краски (т.н. **process**) – типичный **semi-transparent** (полупрозрачные).

[4] В связи с тем, что обычные (**process**, или как это принято в терминологии треппинга – **normal**) краски неидеально прозрачны, порядок их нанесения также стоило бы учитывать. Но для наших задач маскирования это маловажно. Более актуально это для цветоделения, где хорошо известно, что M+Y и Y+M – далеко не одно и тоже. Оба красные, но один «пионерский», а второй малиновый. Поэтому в офсете (в денситометрии) параметр наложения двух красок (бинар) специально отслеживают. Этот параметр также называют **«trapping»**, но к процессу построения ловушек он имеет лишь косвенное отношение.

Пыльский Александр
pylski@yandex.ru
Украина, Киев, 2004
Треппинг и оверпринт

Часть 2. Основы trapping.

В тех случаях, когда нельзя использовать **overprint**, для маскирования дефектов несовмещения применяется второй способ – **trapping**.

Слово **trap** переводится, как **«ловушка»**. В препресс-контексте – цветовая ловушка для дефектов **misregistration**. На рис 10 показано, как специальная ловушка прячет несовмещение между синим кольцом и вырубкой под него в красном фоне.

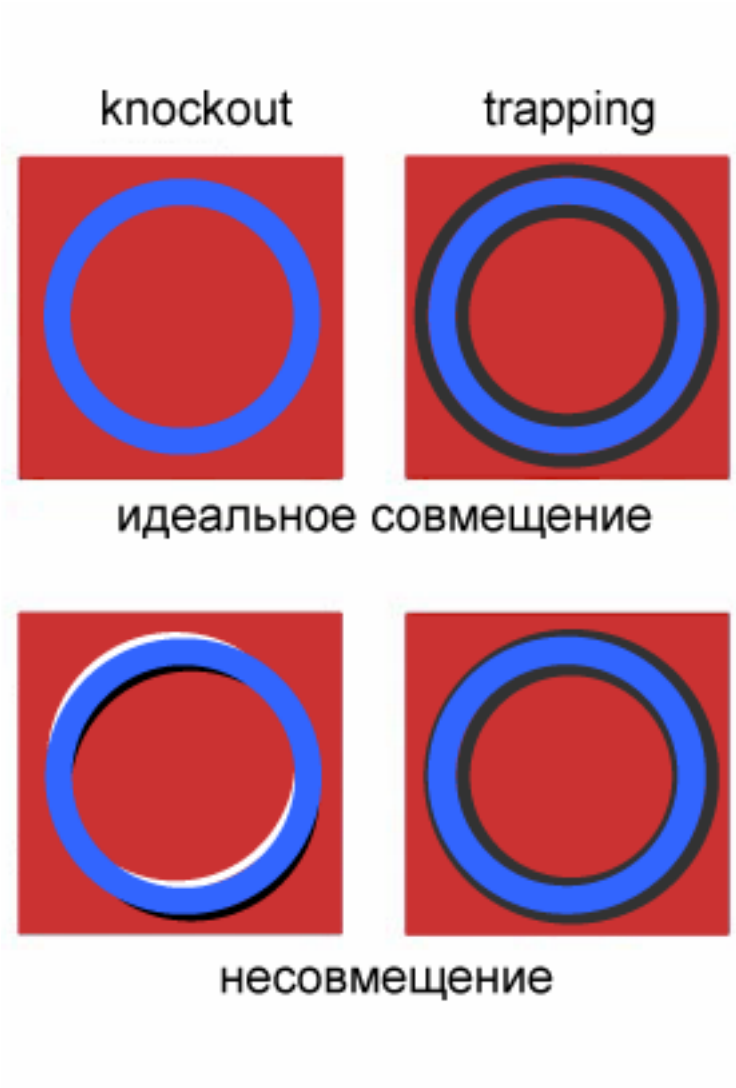

Рис 10. Цветовая ловушка.

По какому принципу работает ловушка? В режиме **knockout** объект и его вырубка строго равны между собой. Малейшее рассогласование – и мы получаем заметные белые огрехи. А если мы несколько увеличим толщину синего кольца, но изменять толщину вырубки не будем? В результате (в наложении цветов) у голубого кольца появится темно-фиолетовый контур, т.н. **keyline** (верхний правый рисунок). Казалось бы – это плохо. Но посмотрите на два нижних рисунка, где имитируется несовмещение при печати. В режиме **knockout** (левый нижний рисунок) у кольца возникли и белые, и темно-фиолетовые паразитные элементы. Большие яркостные перепады. Справа же особых проблем не видно. А возникший контур только подчеркнул синее кольцо, повысил резкость изображения. Создав эффект, подобный **unsharp mask**. Итак, ценой некоторых цветовых смещений (синий с фиолетовым контуром) мы замаскировали резкие яркостные перепады. В таком приеме и заключается суть **trapping**.

Заметность дефектов несовмещения удобно оценивать в ч/б-режиме (рис 11).

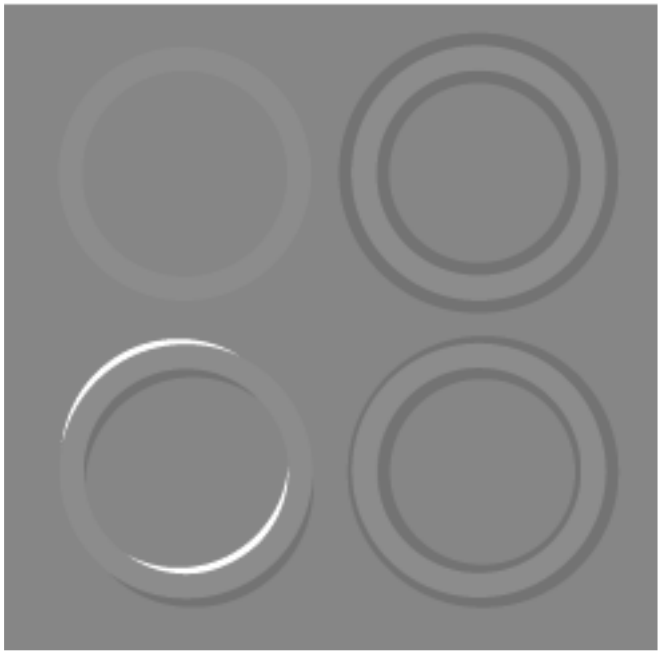

Рис 11. Оценка по серой компоненте.

Немного терминологии

Кроме простого **knockout**, возможны еще два варианта построения вырубки (рис 12).

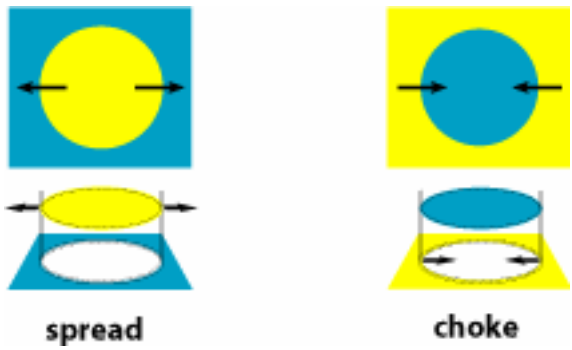

Рис 12. Два направления **trapping**.

Объект больше, чем его проекция, или наоборот, объект меньше, чем его проекция. Оба варианта применяются при **trapping**. Какой из вариантов применять зависит от того, какой из них наименее заметен. Самый лучший **trapping** – незаметный.

- **spread** – расширение объекта относительно его вырубки;
- **choke** – сдавливание, уменьшение вырубки у объекта;
- **keyline** – артефакт, возникающий вследствие появления элемента **trapping**;
- **ND (Neutral Density)** – яркость цвета, точнее, его ахроматичная (серая или нейтральная) составляющая **[5]**.

С последним термином разберемся детальней на примере применения типичных полупрозрачных красок.

Как мы видим, **trapping** по определению вносит в макет некие искажения. На границах объектов, подвергающихся **trapping**, возникают посторонние контуры. Это **keyline**. Наша задача – сделать их незаметным. Как? Дизайнеры хорошо знают принцип **contour-defining**, говорящий о том, что форма объекта задается его контуром. Для темного объекта на светлом фоне визуальный размер задает («рисует») темный контур. Для светлого объекта на темном фоне – наоборот. На основе этих правил уточняем задачу – для минимизации искажений, вносимых **trapping**, нам нужно у любого объекта сохранять размеры соответствующего контура. Наш **keyline** всегда темный - значит он всегда должен находиться на стороне более темного объекта. Так принимается решение, в каком направлении производить **trapping**, когда применять **spread**, а когда **choke** (рис 12).

Итак, первое правило **trapping**:
Светлый цвет всегда тянется под темный.

А какой из наших цветов светлее-темнее? И насколько? Многим известен такой параметр цвета, как плотность (**D**). Чем выше плотность, тем интенсивнее, «мощнее» цвет. Но увы, через плотность нельзя сравнивать цвета между собой. К примеру, очевидно, что желтый намного светлее, чем черный. Но решить, что темнее – magenta или cyan, уже значительно сложнее. Поэтому был введен особый параметр – нейтральная плотность (**Neutral Density** или **ND**). Этот параметр специально выделяет ахроматичную компоненту цвета, что позволяет сравнивать «мощность», светлоту объектов разного цвета. Значение **ND** и определяет, какой из объектов светлее, а какой темнее.

Хорошо известно, что в полиграфии цвет задают, используя цветоделение, т.е. синтез нескольких красок. Поэтому анализ светлоты объектов нужно производить отдельно для каждой краски, используемой при печати. Наиболее часто используют систему **CMYK**, содержащую четыре базовых краски. Однако нередко встречается применение и других красок (т.н. **spot color**, к примеру, хорошо известная система **Pantone**). Все краски, использованные при синтезе цвета, в дальнейшем будем называть компонентами этого цвета, а цвета, содержащие более одной компоненты – составными. Самая темная краска (компонента) в **CMYK** – это черная. Она ахроматична, поэтому у черного **ND(Black) = D(Black)**. С другими красками сложнее. В **trapping**-системах для хранения значений **ND** красок применяются специальные таблицы, которые обычно содержат не только типичные значения для популярных триад (таблица 1), но и подробные таблицы цветов (**color tables**) библиотек **Pantone**. Возможно и вычисление ND из **CMYK** или **Lab**-координат цвета, но точность такого способа несколько ниже. На практике некоторое понижение точности, связанное с применением усредненных данных из упомянутых таблиц (или вычисление **ND** из **CMYK**), вполне допустимо. В особых случаях необходимо использовать денситометры, с помощью которых можно точно замерить **ND** (режим **V** – visual).

ND	**Cyan**	**Magenta**	**Yellow**	**Black**
Euro	0.51	0.62	0.04	1.67
SWOP	0.60	0.76	0.16	1.73

Таблица 1. **Neutral Density** в двух популярных триадах.

Яркость краски в растре можно рассчитать по формуле:

$$ND=-ND(black)*\lg\left(1-R\left(1-10^{-\frac{ND(color)}{ND(black)}}\right)\right)$$

где **ND(color)** – **Neutral Density** 100% плашки соответствующего цвета и **R** – значение растра.

Параметр **Neutral Density** – это десятичный логарифм от уровня нейтральной компоненты. Он удобен тем, что **ND** составного цвета – просто арифметическая сумма **ND компонент**[6].

Два режима trapping

Существуют две различные ситуации, в которых нужно строить **trapping**. Рассмотренные выше примеры, в которых принимало участие два цветных объекта и **trapping** происходил на их границе – это так называемый «мокрый треппинг» (**wet trap**) [7]. Но **trapping** может понадобится и в других случаях. К примеру, двух- или трехкомпонентный объект на белом фоне, или наоборот, применяется выворотка на составной фон. В таком случае нужно говорить о **trapping** индивидуального объекта (точнее, **trapping** пары краска-бумага). Такой режим называют «сухим треппингом» (**dry trap**). Оба режима имеют много общего в своем подходе, но есть и различия. Поэтому рассматривать их будем отдельно.

Trapping пары цветов

Перед созданием элемента **trapping** сперва мы должны ответить на вопрос, а нужен ли он. Кроме тех ситуаций, где вполне справляется **overprint** (см. **часть 1**), есть и такие, где применим обычный **knockout**. **Trapping** не нужен, если:

1. Один из цветов в паре является простым производным от второго

Очевидно, что между объектами с цветами **C100** и **C50** проблем несовмещения не возникнет. В такой паре нет вырубки, поэтому нет и риска появления артефактов. Аналогично и в сложных цветах. К примеру, цвет **C50M50Y50** является производным от **C70M80Y100** (таблица 2) – в такой паре делать **trapping** бессмысленно. Второй цвет полностью состоит из компонент первого, поэтому вырубка в такой паре также не возникает. Принято говорить, что в этом случае первый цвет является производным от второго. Отметим, что цвета при этом могут значительно различаться между собой, но с точки зрения **trapping** они будут «родственными».

	Color 1	**Color 2**	**Разность**	**Trapping**
C	70%	50%	20%	>
M	80%	30%	50%	>
Y	100%	60%	40%	>
K	0%	0%	0%	не нужен
Итак: все компоненты в одном направлении				**не нужен**

Таблица 2. Анализ по компонентам.

*2. Разность цветов в паре не превышает **color step limit***

Как определить, являются ли сложные многокомпонентные цвета в паре «родственными»? Уточним первое правило. Будем рассматривать каждую компоненту (краску) в паре отдельно. Если все компоненты второго объекта меньше, чем у первого, то это явные «родственники». А если нет, но разница весьма незначительна? Порог, при котором разница будет считаться несущественной, принято называть **color step limit**. Он регулируется – чем он ниже, тем больше **trap**-объектов (**keyline**) мы создаем, тем больше вносим искажений в первоначальный макет. Для обычной коммерческой продукции типичное значение **step limit – 25%**. С одним важным замечанием.

В таблице 3 мы видим, что у разности компонент противоположные направления, т.е. они не «родственники». У всех цветных составляющих разность на первый взгляд невысока – 10%, что меньше типичного **color step limit**. Но если у желтого перепад 50-60% практически незаметен, и **trapping** не требует, то перепад в голубом 5-15% уже нуждается в маскировании. Наш параметр **step limit** неточен. Поэтому в современных **trapping**-системах применяют относительный метод вычисления разности цвета, т.н. **relative color step limit**. Он определяется, как разность, деленная на меньшее из значений цвета. Но в том случае, если абсолютная разность менее 5%, относительное значение не вычисляется, и эта компонента не рассматривается. Типичное значение для метода **relative – 200%**.

	Color 1	**Color 2**	**Разность**	**Относ. разность**	**Trapping**
C	15%	5%	10%	10/5=200%	<
M	25%	35%	10%	10/25=40%	>
Y	50%	60%	10%	10/50=20%	не нужен
K	15%	10%	5%	не вычисляется	не нужен
Итак: есть две компоненты, относ. разность которых больше **relative step limit**, и они в противоположном направлении					**нужен**

Таблица 3. Абсолютная и относительная разность цветов.

Подсуммируем оба правила: будем считать две компоненты равными друг другу, если значение одной превосходит значения другой на величину **relative step limit**. Тогда:

Trapping между двумя цветами не нужен, если все компоненты одного цвета больше или равны (с учетом relative step limit) соответствующих компонент второго цвета.

*3. Парой достигнут **common density limit***

При анализе компонент в паре объектов необходимо также учитывать их общую (**common**) составляющую. Она определяет цвет вырубки, возникающей при наложении объектов. На рис 13 показан пример – пара цветов **C100M100Y40K20** и **C20M0Y100K100**. У них есть общая составляющая – **C20M0Y40K20**. Эта составляющая и есть самый светлый цвет, который может возникнуть при несовмещении в такой паре. Если общая составляющая достаточно темная, то **trapping** в такой паре не нужен. Типичный порог (**common density limit**) для высококачественных работ рекомендуется на уровне 0.5D, для простых работ (офсетки-газетки) его можно понизить до 0.35-0.4D.

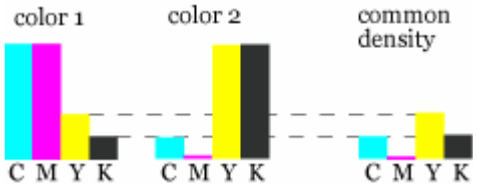

Рис 13. Общая составляющая пары цветов.

4. Хотя бы один из цветов в паре очень светлый

Если один объект из нашей пары по своей яркости (нейтральной компоненте) незначительно отличается от чистой бумаги, то **trapping** здесь точно не нужен. Максимальная разница в яркостях артефактов, возникающих при несовмещении, не может превышать **ND** более светлого объекта. Поэтому, если яркость одного из цветов в паре не превышает **0.04 ND** (за точку отсчета взят желтый), вполне подойдет обычный **knockout** **[8]**. Нередко в таких ситуациях могут применять т.н. **inverse trapping** **[9]** – не маскировать несовмещение с помощью **choke-spread**, а наоборот, специально раздвигать краски между собой. К примеру, в паре **C100-Y100** голубой и желтый стоило бы отделить друг от друга, создавая на границе объектов белый контур (**white outline**). Заметность белого контура в такой паре ниже, чем заметность зеленого, возникающего при несовмещении.

Trapping одного цвета

Маскирование несовмещений в этом режиме более сложная задача. На первый план выходят цветовые огрехи **misregistration**. Порой невозможно решить, как наилучшим образом спрятать артефакты несовмещения в цветном трехкомпонентном тексте, лежащем на белой бумаге. На рис 14 показан пример того, насколько становятся заметны цветовые артефакты на фоне белой бумаги. Мы видим, что хотя произведенный **choke** желтого и позволил несколько снизить заметность несовмещения, голубые артефакты остались весьма заметны. Основная проблема в том, что здесь перестает работать основной прием маскирования – подмена яркостных артефактов на цветовые. Белый фон подчеркивает малейшие ошибки несовмещения. Поэтому желательно еще на стадии дизайна избегать применения таких проблемных элементов, как двух- и трехкомпонентные шрифты малого размера, тонкие линии, выворотки на сложном цветном фоне и т.п. Если же использования сложных (например «фирменных») цветов не избежать, следует заранее обдумать приемы маскирования. Самый простой и радикальный способ в этом случае – использовать дополнительные краски при печати таких сложных цветов. Если же бюджет не позволяет увеличивать расходы на печать, мы должны быть готовы к заметной модификации макета, к нарушению нашего «принципа не-вмешательства».

Рис 14. Цветовые артефакты.

Приемы **trapping** в таком режиме весьма разнообразны и требуют творческого подхода. К примеру, светло-фиолетовому тексту **C60M40** на белом фоне желательно сделать голубой контур. А для тонкой салатовой линии **C40Y95** лучшим вариантом маскирования будет желтый контур. В целом – для двухкомпонентных цветов я рекомендую втягивать меньшую компоненту под большую. И ориентироваться при этом не только на светлоту компоненты, но и на минимизацию цветовых сдвигов. Большинство трех- и четырехкомпонентных цветов могут быть оптимизированы с точки зрения **trapping**. Коричневый цвет можно задать через **C55M65Y65**, а можно через **M30Y30K60**. Второй рецепт значительно проще маскировать. Порой элемент маскирования можно превратить в элемент дизайна. Повторюсь, универсальных подходов к **dry trapping** не бывает, на автоматику в такой ситуации рассчитывать нельзя, такой **trapping** практически всегда зависит от изображения и производится вручную.

Но есть один распространенный случай, когда **dry trapping** можно поручить автомату. Это сложный черный цвет (**rich black**). Очень часто в полиграфии для повышения плотности черного цвета к базовому цвету **process black** добавляют триадные составляющие. При этом используют двух-, трех- или даже четырехкомпонентные рецепты черного **[10]**. Такой радикальный черный часто называют **superblack**, **fat black**, **rich black** («суперчерный», «толстый», «богатый»). Огрехи несовмещения в таком черном крайне опасны, ведь здесь мы наблюдаем наихудший с точки зрения заметности артефактов случай – наибольший диапазон контраста, от белого до радикального черного. К примеру, у белого текста, идущего выворотkой по радикальному черному фону, возникнут разноцветные паразитные контуры, которые резко ухудшат вид. Но именно здесь мы можем превратить недостатки в достоинства. Используя уже известное нам свойство черного скрадывать цветовые огрехи, просто сделаем втяжку цветовых составляющих под черный (рис 15). Такой прием называется **choke rich black**.

Рис 15. Втяжка под черный.

Втяжка под черный давно практикуется опытными дизайнерами. В уже упомянутом примере с выворотkой стоило бы просто назначить белому тексту контур **K100**, и все дефекты несовмещения были бы замаскированы. Аналогично можно поступать и в других ситуациях, требующих втяжки под черный. Естественно, при этом такой контур должен иметь атрибут **knockout** – вот еще один повод опасаться "ленивой" кнопки **all black overprint**. Кроме того, у такого способа есть недостаток, ведь типичный контур во многих издательских программах строится по центру объекта (**center outline**). Что приведет к геометрическим искажениям, белый объект немного уменьшится, наш текст из примера станет тоньше **[11]**. Поэтому я рекомендую всегда пользоваться возможностями специализированных **trapping**-систем, в которых втяжка под черный производится автоматически и не имеет проблем с геометрией или "ленивой" кнопкой. Эта функция настолько важна, что зачастую она работает в принудительном порядке и во многих системах отключить ее невозможно.

Итак, **trapping**-система должна уметь распознавать ситуацию с присутствием **rich black** в макете:
Для объектов с цветом rich black производится втяжка (choke) всех компонент под черный.

Суперчерным считается любой цвет, в составе которого есть черная краска и еще как минимум одна компонента. Осталось уточнить, что именно мы считаем «черным». Это не только **K100**. Более правильным было бы учитывать то, что маскирующие возможности черного проявляются даже тогда, когда он в растре. Я рекомендую делать втяжку под черный с уровня 85% (точнее, **BIL*0.85**) **[12]**. Кроме того, некоторые смесевые (**spot**) краски также подобны черному, к примеру **Pantone Black**. Поэтому в категорию «черный» краску относят по двум различным характеристикам, которые обычно называют **black trapping rules** (в **Indesign** это **black trap thresholds**). Или процентное содержание **K** в объекте больше величины **black color limit**, или же **ND** некоего **spot color** больше **black density limit**.

Кроющие краски, лаки и другие особые случаи

Рассмотренный выше метод анализа «силы» компоненты по **ND** не подходит для кроющих красок. Как уже рассматривалось (см. **часть 1**), для кроющих красок определяющим является порядок печати. Светлая краска, наносимая последней, будет «сильнее», чем

темная, идущая перед ней. Например, «серебро» имеет низкое **ND**, однако если оно печатается последним, то все остальные краски должны втягиваться (**choke**) под него. Если при печати применяется несколько кроющих красок, то для выполнения корректного **trapping** нужно правильно указать порядок их следования – **trap sequence** (или **print order**). К сожалению, в таких ситуациях есть одно жесткое ограничение – далеко не все системы **trapping** могут правильно учитывать взаимодействие нормальных и кроющих красок между собой. Принято считать, что кроющие краски всегда «сильнее» обычных. В том случае, когда кроющая краска печатается первой, такой подход может привести к досадным ошибкам. В некоторых системах (к примеру, **Delta Trapper**) для таких ситуаций предусмотрены специальные таблицы исключений **trap pair**. В остальных случаях **trapping** нужно делать или вручную, или же «подстроить» **ND** так, чтобы обмануть автомат.

Кроме специфики работы с кроющими красками встречается немало вариантов, в которых требуется особое управление при построении **trapping**. К примеру, **overprint** может применяться как к кроющим объектам, так и к бесцветным лакам (формы для местной лакировки). В обоих случаях подход к **trapping** должен быть разным. Для этих целей введем специальный атрибут краски – **opacity**. У него четыре значения:

- **normal** – обычная офсетная полупрозрачная краска, **trapping** происходит по привычным правилам;
- **transparent** – прозрачный лак;
- **opaque** – кроющая краска;
- **opaque & ignore** – особый случай применения кроющей краски (см. ниже).

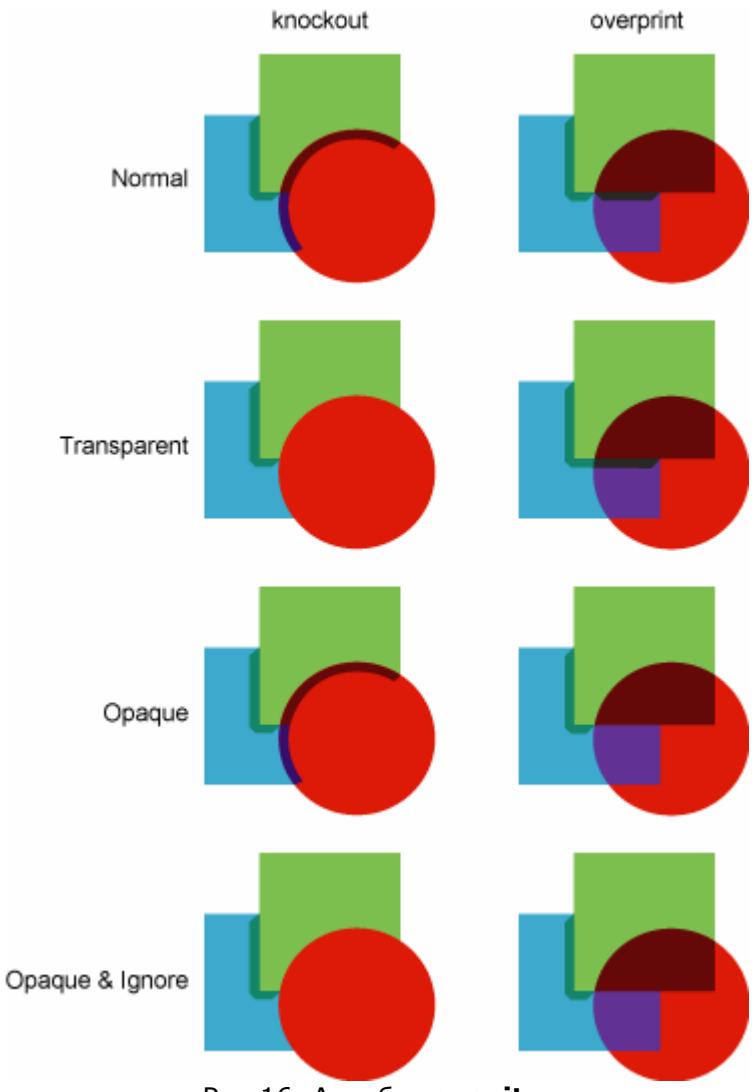

Рис 16. Атрибут **opacity**.

На рис 16 показан пример взаимодействия обычных атрибутов **knockout** и **overprint** с нашим специфическим атрибутом краски. Возьмем два квадрата, голубой и светло-зеленый. Они окрашены обычными красками, при этом голубой квадрат темнее, поэтому светло-зеленый при **trapping** тянется под голубой. Затем к двум квадратам добавляем

специфический объект - красный круг. Для наглядности на рисунке он сделан прозрачным, но это может быть металлизированная краска, клише для тиснения, маска для выборочного лакирования. Третий объект может иметь атрибут **overprint**. Но самое важное – этот объект может влиять на **trapping** первых двух. Для управления **trapping** третьему объекту присвоим специальный атрибут **opacity**. Получаем восемь (строго говоря – шесть) вариантов:

1. *Красный круг имеет атрибуты **normal** и **knockout**.* Происходит обычный **trapping** по правилу **ND**. Как мы видим, красный круг наиболее темный (имеет самое высокое **ND**), поэтому оба квадрата потянуло под него.
2. *Атрибуты **normal** и **overprint**.* В этом случае **trapping** с кругом не происходит (он просто невозможен – вырубки ведь не образовалось). Чуть изменился **trapping** пары квадратов, в точке, где встретились все три объекта, образовался разрыв. Смысл этого я объясню позже, в разделе **trap geometry**.
3. *Атрибуты **transparent** и **knockout**.* Особый случай, и на первый взгляд странный. Ведь абсолютно прозрачный, бесцветный лак по идее требует **overprint**. Но в полиграфии и не такие странности бывают. Этот вариант используется, когда нужно запретить **trapping** некоторым объектам.
4. *Атрибуты **transparent** и **overprint**.* Это вариант для выборочного лакирования. Никакой **trapping** для круга не производится, круг не влияет на **trapping** квадратов.
5. *Атрибуты **opaque** и **knockout**.* В этом варианте **trapping** со специфическим объектом происходит по правилу «кроющая краска всегда сильнее обычной».
6. *Атрибуты **opaque** и **overprint**.* Похоже на вариант 4, но здесь наш специфический объект влияет на соседей. Под красным кругом исчез **trapping** пары квадратов. Логика здесь проста - если объект кроющий, то проблемы несовмещения под ним неважны, ведь сквозь него ничего проглядывать не может. Так зачем там делать **trapping**?

Иногда возникает ситуация, когда объекты, крашенные нашим специфическим цветом, должны полностью запретить **trapping** с собой. Такой случай характерен для объектов, **trapping** которым сделан ручным способом. Для этого и используется специальный атрибут **opaque & ignore**.

Далі буде.

---

[5] Параметр **ND** схож по своему смыслу с общепризнанным определением яркости (**luminosity**) цвета. Поэтому позволим себе в статье иногда использовать термины «яркость»и «светлота».

[6] Строго говоря, это не совсем так. Но на практике в 90% случаев точность такого приближения вполне достаточна.

[7] Практически все автоматизированные **trapping**-системы работают в режиме **wet trap**, для второго режима применяя только втяжку под суперчерный (**choke rich black**).

[8] Строго говоря, в такой ситуации начинают действовать другие правила, ведь такой режим становится схож с **trapping** пары краска-бумага (**dry trap**).

[9] К примеру, в программе **Heidelberg Supertrap** такой режим называется **keepaway mode**.

[10] Во многих типографиях существуют технологические стандарты на рецептуру **rich black**. Они учитывают и ограничение на общее количество краски (**total ink limit**) при печати, и особенности красконаложения в сложных балансах черного.

[11] В некоторых программах эта проблема решаема, контуру можно назначить **outside (inside) outline**.

[12] Не стоит забывать о том, что максимальный достижимый уровень черного в печати определяется технологическим порогом **black ink limit (BIL)** и зависит от печати. К примеру в газетке он обычно находится на уровне 85%.

Пыльский Александр
pylski@yandex.ru
Украина, Киев, 2004
Треппинг и оверпринт

Часть 3. Построение trapping

Мы научились принимать решение о том, когда **trapping** нужен. Теперь выясним вопрос – какой. Напомню, я описываю в основном работу в режиме пары цветов (**wet trap**).

*1. Ширина элемента **trap width***

Эта ширина определяется нашим ожидаемым несовмещением. Я уже отмечал, что нормальным несовмещением в печати считается половина линии растра. У некоторых типографий подход может быть иной, поэтому напрямую связывать **trap width** с линиатурой печати было бы некорректно. В целом можно сказать, что для продукции высокого класса типично несовмещение не более 70-90 мкм. Или приблизительно 0.25 pt – толщина креста в метке совмещения (реза). Конечно, такой подход несколько упрощен. К примеру, на двухкрасочной планетарной печатной машине можно ожидать очень неплохого совмещения между парами цветов первого и второго прогонов. И основное внимание уделять совмещению между прогонами. Такой подход называют **«лестничным» trapping**. При флексографской печати большие проблемы при совмещении вызывают геометрические искажения (**distortion**) пластичных форм, поэтому там ожидаемые несовмещения по длине и ширине могут быть различны.

Оценка ожидаемого несовмещения зависит от многих факторов. На несовмещение при цветной печати влияют как всевозможные геометрические искажения печатного процесса, которые могут различаться у каждой секции (краски), так и проблемы совмещения (регистрации) секций и прогонов друг с другом. Давать универсальные рекомендации здесь очень сложно. Можно только отметить, что **trapping** при производстве крупнотиражной продукции (к примеру этикетки/упаковки) обычно проводится более тщательно и порой учитывает все множество проблем, связанных с совмещением. При подготовке журнальной и рекламной продукции чаще применяются несколько упрощенные подходы, в т.ч. «типичные», несколько заниженные значения **trap width** **[13]**.

Особо хотелось бы отметить, что такой подход применим только в тех случаях, когда **trapping** производится адекватным алгоритмом. Когда **trap**-элемент возникает только на границе пары цветов. Далеко не все программы способны производить такой **trapping**. К примеру, не стоит эти рекомендации использовать для таких приложений, как **QuarkXpress**. Дело в том, что при **trapping QuarkXpress** (как и некоторые другие программы), изменит размер всего объекта(!) на величину **trap width** **[14]**. Что может значительно изменить вид макета. К примеру, текстовым элементом при **trapping** в **QuarkXpress** является строка. Не много, не мало – именно строка. Вспомним рис 9 (см. **часть 1**). Применив утолщение (**auto amount+**) размером 0.25pt к нормальному тексту 10 кегля, мы получим жирное начертание (**bold**).

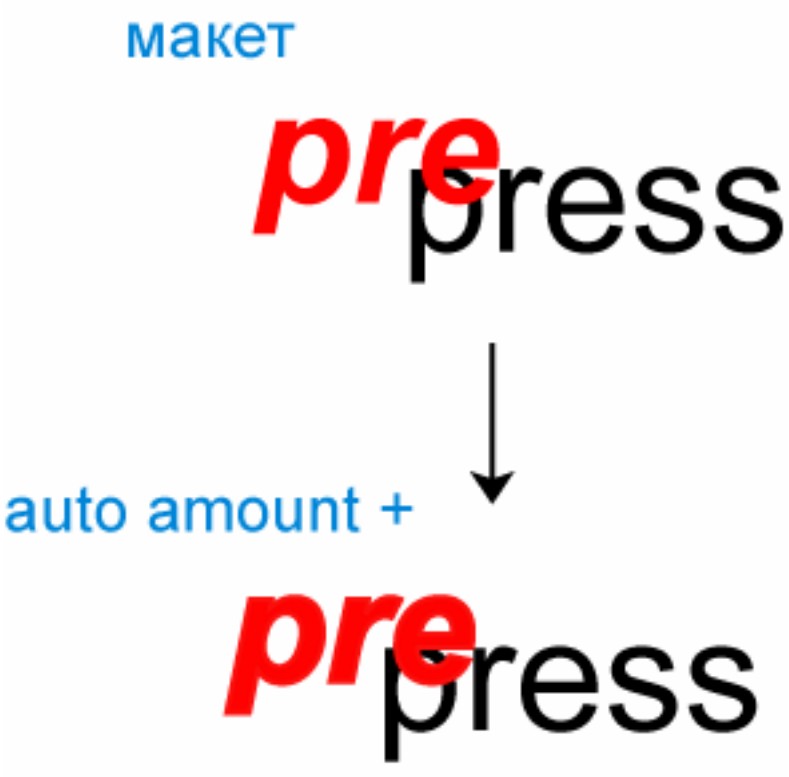

Рис 17. **Trapping** в **QuarkXPress**.

Мы должны помнить, что несмотря на все наши ухищрения, сделать **keyline** абсолютно незаметным невозможно. Увеличивать ширину "с запасом" настойчиво не рекомендуется. Но с одним исключением.

*2. Отдельное правило для черного **black trap width***

Для черного запас не повредит. Ведь мы помним, что искажения цвета в черном не так заметны, как в остальных цветах. Поэтому мы можем без особого риска увеличить ширину **trap**-объектов (**keyline**) при втяжке под черный в полтора-два раза. Параметр **black width scaling** позволяет управлять дополнительным запасом по ширине для черного. На рис 18 показано, как синее кольцо расширяется при **trapping** на красном и на черном фоне.

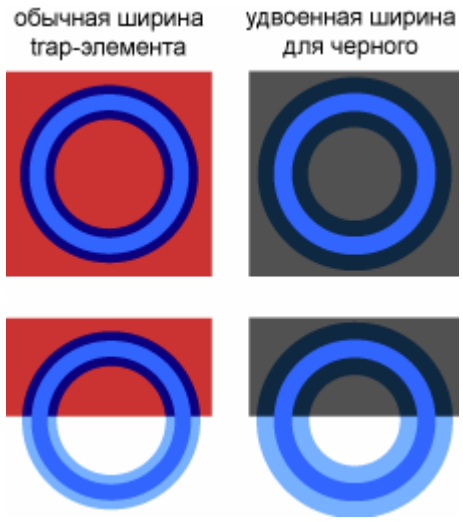

Рис 18. Правило ширины для черного.

*3. Контроль за мелкими элементами **small edge control***

Возможны ситуации, когда размеры объектов становятся сравнимы с **trap width**. Чтобы избежать ошибок, возникающих при несовмещении (рис 19), в таких местах ширина **trapping** должна быть пропорционально уменьшена.

Рис 19. Контроль мелких элементов.

В системах, построенных на базе **Adobe InRip Trapping**, все мелкие элементы проверяют по правилу:
Trap width не может превышать половины ширины мелкого объекта.

Иногда в мелких объектах предпочтительнее использовать **overprint**. Дело в том, что **keyline**, возникающие при **trapping**, не только вносят некоторые цветовые сдвиги в объект, но также могут влиять на общий вид изображения. Пока объект имеет крупные (по сравнению с **keyline**) размеры, таким влиянием можно пренебречь. Но когда размеры его становятся сравнимы с **keyline**, это может привести к заметным ошибкам (рис 20).

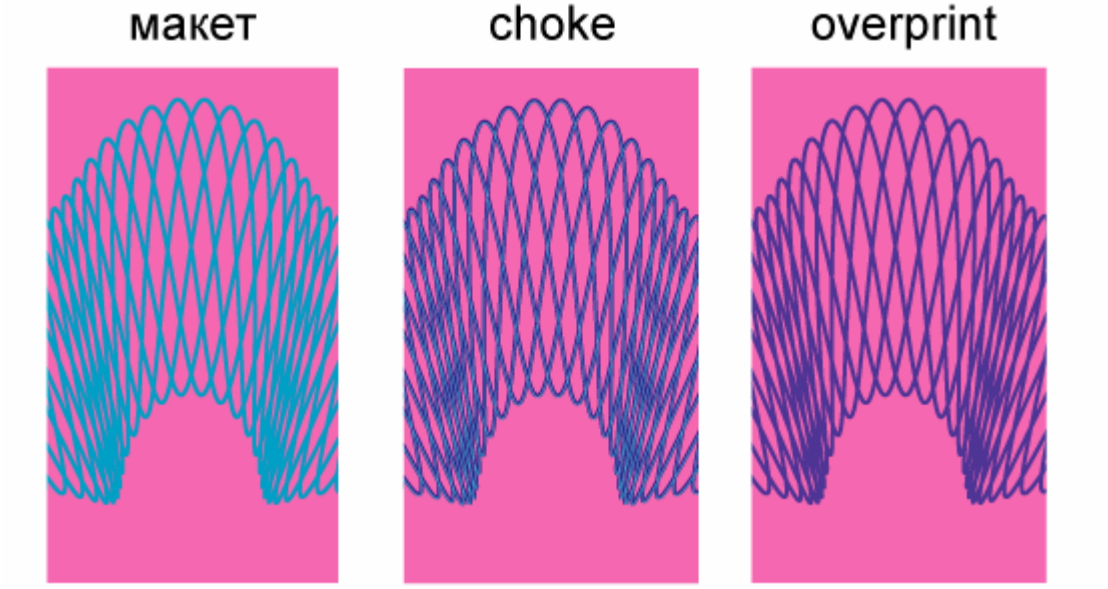

Рис 20. Искажения в тонких линиях.

В этих случаях мы вынуждены чем либо пожертвовать – или допускать цветовые искажения при **overprint** мелких объектов, или мириться с визуальными искажениями рисунка. Учитывая то, что искажения в этих случаях неизбежны, такие решения желательно принимать еще во время дизайна иллюстрации. В программе **Heidelberg Supertrap** так контролируют **choke** для тонких линий. По правилу, схожему с **overprint black** (см. **часть 1**) – для **stroke**, имеющих толщину не более заданной (через параметр **line split**), **trapping** заменяется на **overprint**. В большинстве других систем такой контроль нужно выполнять вручную.

*4. Направление **trapping***

Напомню правило **contour-defining** – зрительно форму объекта задает его контур. Этот контур формируется при наложении двух элементов – собственно объекта и его вырубки. В простейшем случае объект и вырубка строго равны между собой по размеру – это **knockout**. Если вырубка меньше, чем сам объект – это **choke**. Если объект увеличивается, а вырубка

неизменна – это **spread**. В двух последних вариантах возникает наложение красок – **keyline**. Именно **keyline** и задает наш контур, т.е. определяет визуально размеры объекта. Поэтому важно, чтобы **keyline** сохранял размеры объектов, заданные при дизайне. Если в паре объектов верхний темнее нижнего, то верхний объект задает размеры, а нижний будет втянут (**choke**) под него. Если же наоборот, в паре объектов верхний светлее нижнего, то задавать размеры будет нижний, а верхний будет расширен (**spread**). Кто из объектов в паре светлее-темнее, определяет их **ND** (для обычных красок).

А если объекты в паре равны или очень близки по **ND**? Тогда **trapping** лучше производить симметрично, в обе стороны от границы объектов – это режим **centerline**. Параметр **centerline trap limit** определяет, насколько точно объекты в паре должны быть равны по **ND**. При 100% (идеально равны) **centerline** будет возникать очень редко. При 0% все **trap**-объекты (кроме **rich black**) будут построены **centerline**. На мой взгляд, наиболее удачное значение – 90%. При таком значении **centerline trap limit** снижается вероятность появления артефакта **«прыгающий choke-spread»**, который зачастую возникает при **trapping** градиента (рис 21). В развитых **trapping**-системах контролем за **«прыгающим choke-spread»** занимается специальный алгоритм, чувствительность которого управляется параметром **sliding traps color limit** (**shift limit** в **Trapwise**).

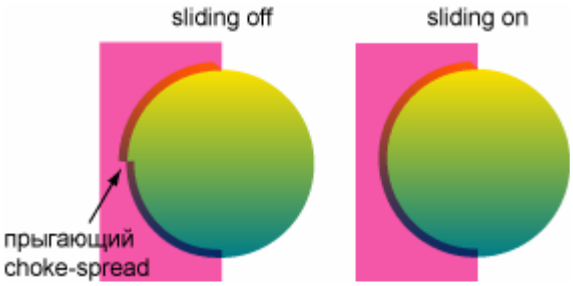

Рис 21. **Trapping** градиентов.

*5. Какого цвета должен быть **keyline***

Простейший вариант – цвет **keyline** определяется максимальными значениями компонент пары. Но в этом случае цвет **keyline** будет чрезмерно темным (рис 22). Что нарушает наш «священный» принцип невмешательства.

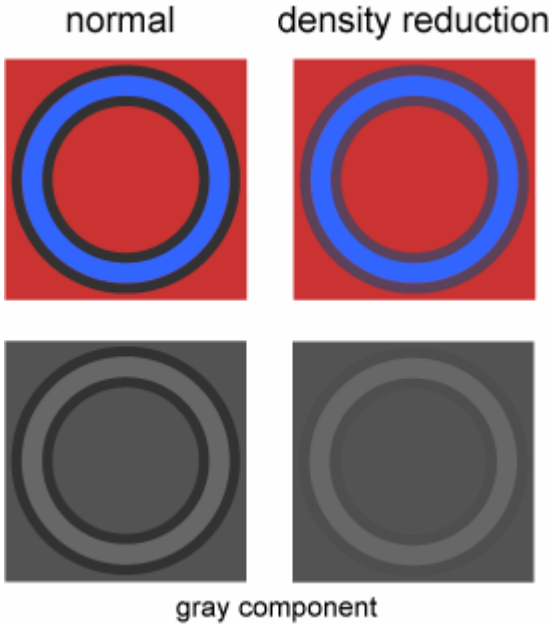

Рис 22. Понижение заметности **keyline**.

Поэтому часто используются более продвинутые методы вычисления цвета **keyline**, снижающие его заметность. В разных системах они имеют различные названия: **tint reduction**, **trap color reduction**, **trap color scaling**.

	Color 1	**Color 2**	**Normal**	**ND Reduction 100%**

C	0%	100%	100%	77%
M	100%	60%	100%	78%
Y	100%	0%	100%	70%
K	0%	0%	0%	0%
ND	0.66	0.82	1.17	0.82

Степень снижения количества краски регулируется. Самый современный метод – **trap color density reduction**. Он понижает **ND keyline** до уровня более темного объекта пары. Полностью исключая таким образом скачки яркости в области **trapping**.

*6. **Trap geometry** или какой формы должен быть **keyline***

Этот достаточно сложный вопрос разобьем на три более простых: какой формы у **keyline** должны быть окончания, какой формы углы, и как вести себя на стыке двух элементов **trapping**. В простейшем случае наиболее оптимальной формой окончания с точки зрения незаметности является угол 45 градусов, т.н. **miter**. В более сложных случаях (рис 23) порой предпочтительно использовать ту форму, которую задает объект, т.н. **clip choke**.

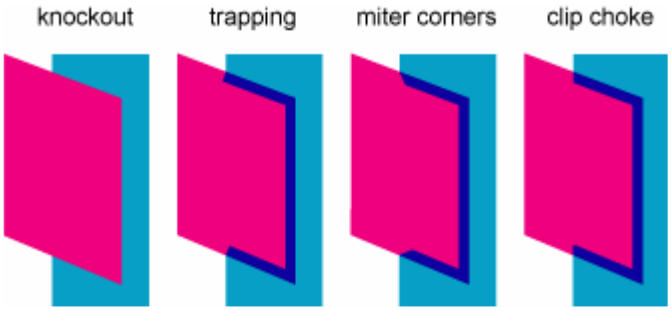

Рис 23. Форма окончания keyline.

Однако применять **clip choke** нужно с осторожностью, этот способ порой приводит к ошибкам (рис 24).

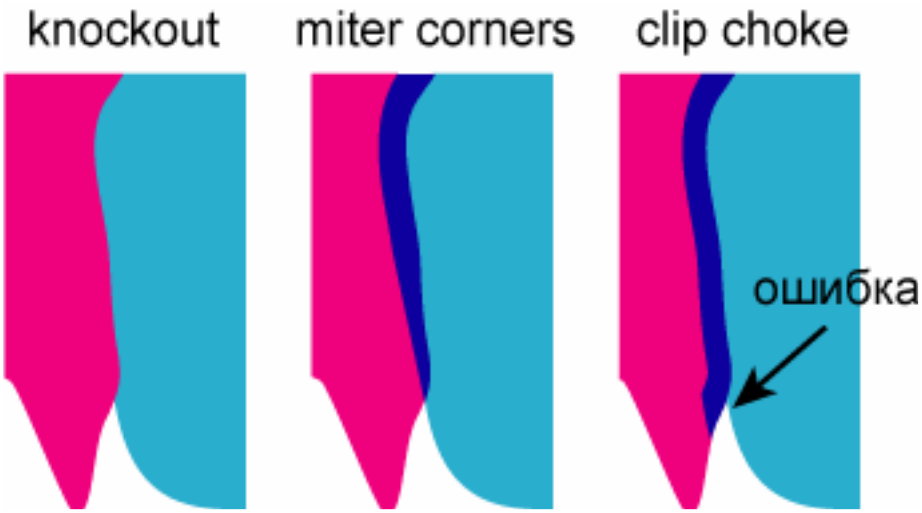

Рис 24. Ошибка при **clip choke**.

В системах, построенных на базе **Adobe InRip Trapping**, используют элегантный алгоритм управления формой окончания, сочетающий преимущества **miter** и **clip choke**, и не приводящий к таким ошибкам.

Формой углов (рис 25) в сложных **keyline** управляет **trap corner shape** (**line join** в **Supertrap**, **trap appearance join style** в **Indesign**). Какую именно форму выбрать? Естественно – минимально заметную. Простейший метод **miter** может давать весьма заметные огрехи, особенно при углах меньше 30?. Срезать угол методом **bevel** – также не лучшее решение, он плохо маскирует несовмещения. Поэтому здесь популярны более развитые методы сглаживания острых углов, такие как **square**, **triangle** и **round**.

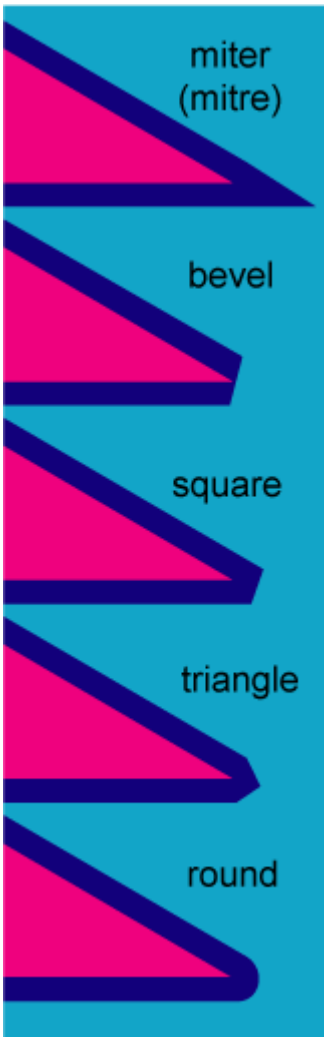

Рис 25. Форма угла.

До сих пор мы рассматривали простейший случай – **trapping** пары. А что происходит в случае, когда объектов становится больше двух? К примеру – три (рис 26).

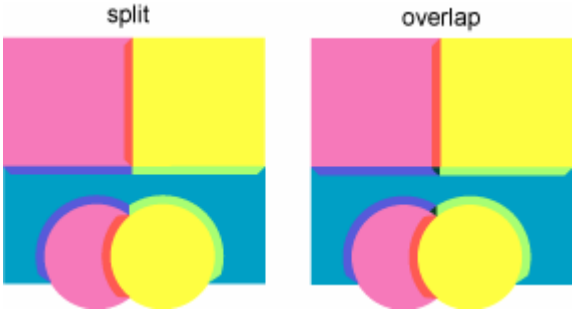

Рис 26. Управление наложением.

Легко представить, что в таком случае возможны ситуации, когда элементы **trapping** будут накладываться друг на друга. При этом наложении (**overlap**) возможно появление нежелательных темных точек. С другой стороны, если мы будем избегать такого наложения (**split**), рискуем получить белую точку. Поэтому в некоторых системах **trapping** такой ситуацией специально управляют. В **Indesign** это **trap appearance end style**. Если хотя бы один из трех объектов достаточно темный, предпочтительно использовать **overlap**. Темная точка примкнет визуально к темному объекту и дефект будет практически незаметен. Если же все цвета яркие или светлые – лучше избегать наложений.

*6. **Trapping** растровых изображений*

До сих пор мы рассматривали работу с векторными и текстовыми объектами. **Trapping** растровых изображений основан на тех же принципах, но имеет свои особенности. Высококачественный **trapping** при использовании растровых изображений сделать очень сложно. При разработке всевозможной этикетки или упаковки, прочей продукции с

высокими требованиями к **trapping** я рекомендую максимально использовать векторные форматы для изображений. И в обычной коммерческой продукции желательно все отсканированные логотипы, клише, и пр. преобразовать в вектор. Но бывают ситуации, когда **trapping** с использованием растрового изображения необходим. Существуют три варианта **trapping** с участием растровых изображений:

*6.1 **trapping** пары объект (вектор) – картинка (растр);*

Вспомним пример из **части 1**. Наш векторный объект - это большое клише, **overprint** которому давать опасно. Тиснение идет поверх картинки. Приладка на тиснении имеет невысокую точность, поэтому мы должны компенсировать это затягиванием картинки под клише на пару миллиметров. Второй пример – в качестве фона используется металлизированная краска. Для полноцветных изображений на таком фоне также требуется **trapping**. В подобных ситуациях он включается опцией **trap object to images**.

*6.2 **trapping** пары картинка (растр) – картинка (растр);*

А если в нашем примере клише не было векторизовано и поставлено в макет растром (**contone**). Или в спешке забыли векторизовать логотип, который лежит поверх растрового изображения. В общем случае – если крупный, типично "векторный" элемент дизайна по каким-либо причинам сделан растром и пересекается с другим растровым изображением – включаем опцию **trap images to images**.

*6.3 **trapping** внутри изображения.*

Крайне нежелательно использовать этот метод. Для нормальной растровой картинки результат будет ужасен – потеря резкости, «мыльность», прочие артефакты. Зачем же тогда многие программы имеют опцию **trap within image**? Для особых случаев. К примеру, когда все макетирование сделали в растровом редакторе (**PhotoShop**) и сохранили этот чудо-макет в формате tiff.

Направление **trapping** при работе с растрами нередко нужно задавать вручную. В таких изображениях светлота в зоне **trapping** может резко изменяться даже на уровне от пикселя к пикселю (рис 27). И обычный метод выбора направления по **ND** вызовет неприемлемые скачки **spreads-choke**. Я рекомендую использовать метод **center**. Как в случае, когда **ND** обеих объектов на границе близки, так и в случае, когда они постоянно и резко изменяются. Методы **choke** и **spreads** можно использовать только тогда, когда светлота одного объекта заведомо выше (ниже) второго по всей границе, или же один из объектов - кроющий.

Рис 27. Направление **trapping** при работе с растрами.

В некоторых случаях (например, при использовании **copydot**) растровое изображение может быть представлено в черно-белом (**one bit**) виде. Обработка такого формата включается отдельно опцией **trap one bit images**.

---

[13] В расчетах ожидаемого несовмещения применяются статистические методы. Среднее (или типичное) значение величины несовмещения допускает, что в тираже могут встречаться т.н. «броски» - листы, в которых несовмещение оказалось больше ожидаемого.Не всякая продукция допускает такие «броски» внутри тиража, поэтому при производстве этикетки/упаковки (к примеру) обычно используют более высокие значения **trap width**.

[14] В **QuarkXpress** по умолчанию используется ширина **trap width** 0.144 pt (50 мкм). Несмотря на то, что это явно заниженное значение, я не советую его изменять.