AsyTEX ～ Asymptote in TEX ver0.10
簡易マニュアル

1　ソフト紹介

最近 (でもないか？) あちらこちらで，Asymptote なるプログラムのことが話題になっています。

Asymptote は「METAPOST を発展させたようなプログラム」で,「ベクタグラフィック言語」とよばれるものの一種です。曲線の指定方法など，基本的なアイデアは METAPOST のそれの延長上にあり，METAPOST を使ったことのある人にはどういう機能を持ったものか，想像しやすいと思います。(METAPOST を使ったことのない人は，うちのサイトの METAPOST のコーナーを覗いてみてください。) ただ，変数の宣言など，言語としての作法は C 言語に近いため，METAPOST に慣れ親しんでいる人は，この点においては逆にとまどうかも知れません。あと，METAPOST と比べてもめちゃくちゃ高機能な感じです。

この Asymptote には，LaTeX などと連携して使うことを想定して，Asymptote のコードを LaTeX ソースに埋め込むためのマクロが初めから用意されているのですが，Asymptote の実行は手動のため,

LaTeX の実行 $\Longrightarrow$ 抽出された Asymptote コードに Asymptote を実行 $\Longrightarrow$ 再度 LaTeX の実行

という手間を踏まなければなりません。

で，これは面倒だ，ってことで，拙作の METAPOST in TEX (MEPOTEX) と同じように，1 回 LaTeX を実行すれば，自動で Asymptote を起動して，図版の貼り込みまでやってくれるマクロを作ってしまおう，と思い立ち，作ってみたのがこのパッケージです。

というわけで，作っては見たものの。。。 Asymptote って想像以上に起動が重かったです (機能を考えれば当然かも知れませんが)。1 回の LaTeX の実行で図の貼り込みまで終わらせるためには，図 1 つ 1 つに対して Asymptote を起動する必要があるのですが，どんな簡単な図でも 5 秒は止まる感じで，図が 100 個も 200 個もあったら結構シャレにならない感じです。昔々，パソコンが非力だった時代，TEX が 1 ページをタイプセットするのに 1 分ぐらいかかっていた時代を知っている身からすれば，図 1 個に 5 秒などたいしたこと無いとも思えますが，最近の速いパソコンで「TEX のタイプセットが一瞬で終わる」のに慣れていると，ちょっとストレスが大きいかも知れません。正直なところ,「簡単な図をたくさん」ということなら METAPOST の利用を**強く**お勧めします。

逆に Asymptote の高機能を活かした使い方 — たとえばシェーディングのかかった立体図形を表示させ，画面上で (マウスを使って) 好きな向きに回転させ，気に入った角度からの図を EPS ファイルに出力する，などといった使い方 — をするなら，LaTeX ソース内に Asymptote ソースを埋め込むのではなく，Asymptote ソースを別個に作りそれを直接 Asymptote に渡す方がいいように思います。(LaTeX ソース内に埋め込んだがために使えない，あるいは使いづらくなる機能が多々ありますので。)

というわけで，いろいろ中途半端な立ち位置にあるマクロパッケージとなってしまいましたが,「ちょっと Asymptote を使ってみよう」という軽い気持ちで使うには十分なものだと思いますので，よろしければ使ってみて下さい。

2　インストール方法

うちのパソコンは WindowsXP ですので，それをベースに説明します。それ以外の OS の場合は適宜読み替えて下さい。

まず，Asymptote 本体はインストール済みであるとします。(Asymptote 本体のインストール方法は TeX Wiki などにありますので，そちらでお調べ願います。)

TeX Wiki のインストール方法にも記載されていますが，インストールすればインストール先のディレクトリ (フォルダ) 内に

```
asy.bat
```

というファイルができていますので，これをパスが通った[1]ディレクトリにコピーします。これにより Asymptote が

[1]ディレクトリにパスを通すというのは，[スタート]→[コントロールパネル]→[システム]→[詳細設定]→[環境変数] の「システム環境変数」にある Path に，そのディレクトリの位置を登録することを意味します。— Path を選択し，[編集] ボタンを押し，既存の値の末尾に半角セミコロン (;) を付け，そのうしろに登録したいディレクトリを (ドライブレターからフルパスで) 書き込みます。なお，コントロールパネルを閉じた後，一旦パソコンを再起動して下さい。

asy **ファイル名**

で実行できるようになります。一度「コマンドプロンプト」を起動し，

asy

と入力してエンターキーを押してみて下さい (起動実験ですのでファイル名は不要です)。

Welcome to Asymptote version X.XX (to view the manual, type help)[2]

と表示されれば成功です。今は特にすることもないので，

quit

と入力して終了させて下さい。

次に，これも TeX Wiki のインストール方法に記載されていますが，コンフィグファイルを設定します。一度でも Asymptote を起動していれば

C:¥Documents and Settings¥**ユーザー名**¥.asy[3]

というディレクトリができているはずですので，そこに config.asy というファイルを作ります。中味のお勧めは TeX Wiki を参考にして下さい。で，TeX Wiki 記載の内容に加えて dir を必ず設定して下さい。これは，Asymptote がソースやマクロなどを検索するディレクトリを指定する項目です。ここに，AsyTEX 付属のマクロ (後述) を保存するディレクトリと，ついでに，Asymptote 付属のサンプルプログラムがあるディレクトリを登録しておきます。たとえば，Asymptote を d:¥Asymptote にインストールしたなら

dir="d:/Asymptote/examples;d:/Asymptote/AsyTeX";

の 1 行を config.asy に追加することになります。

次はいよいよ AsyTEX のファイルのインストールです。といっても現在ファイルが 2 個だけですが。

まず，AsyTeX.sty (拡張子が .sty の方) を TEX が読める場所に置きます。最近の TEX の配布セットなら

(TEX のインストール先)¥share¥texmf-local¥tex¥

の下に適当な名前のディレクトリ (たとえば AsyTeX) を作ってそこに AsyTeX.sty をコピーします。ちょっと古い配布セットだと texmf-local が無いので，そのときは texmf に読み替えて下さい。

次に，AsyTeX.asy (拡張子が .asy の方) を Asymptote が読める場所に置きます。これは上で config.asy 内の dir に設定した場所です。dir に設定した通りのディレクトリ (上の例なら AsyTeX) を作ってそこに AsyTeX.asy をコピーします。

最後に，TEX から Asymptote を起動できるようにします。これは最近の TEX とちょっと古い TEX でやり方が違います。

- ちょっと古い TEX (目安は'09 年 2 月以前)

  他のプログラムの実行を「すべて許可」「すべて不許可」の二択ですので,「すべて許可」にします。具体的には TEX の入っているディレクトリの share¥texmf¥web2c 内にある texmf.cnf ファイルの中の

  shell_escape = f

  を

  shell_escape = t

  に変更します (すでに t になっていればそのまま)。

  (注) この設定のときは Asymptote だけでなくあらゆるプログラムが TEX から起動できるようになりますので,

  素性の怪しいソースをタイプセットするのは危険

  です。見ず知らずの人が作ったソースをタイプセットすることが多い人は texmf.cnf ファイルを書き換えるのではなく，LATEX 起動時に個別に許可・不許可を指定するべきでしょう。個別に指定するには，(texmf.cnf ファイルの中は shell_escape = f のままで)

  許可時：platex -shell-escape **ファイル名**

  不許可時：ふつうに platex **ファイル名**

  と使い分けます。

- 最近の TEX (目安は'09 年 3 月以降)

  他のプログラムの実行を「すべて許可」「すべて不許可」の他「許可リスト内のもののみ許可」の三択です。「すべて許可」より「リスト内のもののみ」の方が安全ですので，許可リスト内に Asymptote を追加します。具体的に

[2] X.XX の部分はバージョン名が入ります。
[3] ユーザー名はもちろんパソコンに登録されているあなたのユーザー名です。

は TEX の入っているディレクトリの share¥texmf¥web2c 内にある texmf.cnf ファイルの中の

```
shell_escape_commands = ¥
```

から始まるブロックの最後 (最後の行の ¥ の直前) に ,asy を追加します。TEX のバージョンによって登録されているプログラム名は違うかも知れませんが，たとえば次のようになります。

```
shell_escape_commands = ¥
bibtex,dvips,dvipsk,epstopdf,epspdf,etex,gnuplot,kpsewhich,¥
latex,luatex,lualatex,makeindex,mpost,metafun,¥
pdfcrop,pdflatex,pdfluatex,ps2pdf,pstopdf,pygmentize,¥
tex,texexec,texmfstart,asy¥
```

- なお，どうしても TEX から他のプログラムを起動することが気持ち悪いという人のために，Asymptote を手動で起動するモードもあります (後述) ので，その場合はここでの設定は不要です。— しかしその場合，Asymptote に付属の asymptote.sty でことは足りるので，AsyTEX を使う意味はないような。。。

これで準備はすべて完了です。

3 使い方その 1 — プリアンブルの記述

AsyTEX を使うには，LATEX ソースのプリアンブルに AsyTEX の読み込みを指示します。なお，AsyTEX は graphicx パッケージの読み込みも仮定していますので，AsyTEX に先だって graphicx パッケージの読み込みを指示する必要もあります。したがって，ソースの先頭部分はたとえば次のようになります。

```
¥documentclass{jarticle}
¥usepackage{graphicx}
¥usepackage{AsyTeX}

¥begin{document}
```

上の例は最低限の記述で，実際には各種パッケージオプションを追加することになります。

まず graphicx パッケージについて。

画像を扱う場合，ビューア (ドライバ) によって出すべき命令は結構違います。この違いを吸収してくれるのが graphicx パッケージで，パッケージオプションにビューア (ドライバ) 名を指定するだけでそのビューア (ドライバ) に最適な命令を生成してくれます。しかしこれは裏を返せば,「ビューア (ドライバ) 名を指定しなければ適切な命令が生成されないかも知れない」ということでもあります。具体的には次のようにします[4]。

dviout を使うとき	¥usepackage[dviout]{graphicx}
dvips(k) を使うとき	¥usepackage[dvips]{graphicx}
dvipdfm(x) を使うとき	¥usepackage[dvipdfm]{graphicx}

次に，AsyTEX について。

こちらは以下のパッケージオプションがあります。

- **実行形式指定系** — 次の 3 つは排反です。どれか 1 つを選択します (デフォルトは setrue)。
  - setrue — 図 1 個につき 1 回ずつ Asymptote を自動起動します。(目次や相互参照がなければ) 1 回の LATEX タイプセットですべてが終わります。
  - sefalse — Asymptote を一切起動しません。埋め込まれている Asymptote ソースは別ファイルにまとめて抽出されますので，手動で (コマンドプロンプトなどから) Asymptote を起動し，それで生成された画像を取り込むため，さらにもう一度 LATEX のタイプセットをかけます。

    TEX から別プログラムを起動するのが容認できない人向けです。

[4] 少し前までは dvipdfm と dvipdfmx はパッケージオプションが共用だったのですが，最近の TEX 配布セットでは個別の設定ファイルができて ¥usepackage[dvipdfmx]{graphicx} とできるようになったそうです。

- lump — sefalse と同様埋め込まれている Asymptote ソースを 1 つのファイルにまとめて抽出しますが，1 回目のタイプセット時 ¥end{document} に到ったときに自動で Asymptote が起動し，一気に画像を作ります。この画像を取り込むためにもう一度タイプセットします (2 回目は Asymptote が起動しません)。

ちなみに，setrue は shell escape true, sefalse は shell escape false, lump は一括払い (lump-sum) という意味です。

ユーザーの最小の操作回数は

setrue — LaTeX 1 回のみ
sefalse — LaTeX 2 回，Asymptote 1 回
lump — LaTeX 2 回

です。これだけ見ると lump が中途半端でメリットが見えないのですが，実は Asymptote の起動・実行には結構時間がかかります。一括にすれば少なくとも起動にかかる時間が節約できますので，かなり所用時間を短縮できます。(17 個の簡単な図で試してみたところ 40 % 近く短縮できました。)

あと，sefalse の場合をもう少し詳しく説明しておきますと，たとえばもとの LaTeX ソースが test.tex という名前だった場合，@test.asy のように先頭に @ のついた拡張子 .asy のファイルが生成されるため，それに対して Asymptote を実行します。したがって，次のような手順になります。

```
platex test
asy @test
platex test
```

- **更新スキップ系** — 次の 2 つは排反です。いずれか 1 つを選択します (デフォルトは renew)。
  - renew — すべての画像を作り直します。
  - notrenew — すでにある画像の作成をスキップします。なお,「すでにある」かどうかはファイルがあるかどうかだけで判断していますので，その図より前に別の図を挿入して図の番号が変わった場合などは，別の図を誤認してしまいます。図の番号が変わるときなどは renew を使って下さい。

上でも描きましたが，Asymptote の起動・実行はかなり時間がかかりますので，notrenew オプション無しではソースの作成にかなりのストレスを伴うでしょう。後述しますが特定の図のみ renew あるいは notrenew に固定することもできますので，うまく使い分けて下さい。

- **マクロ調整系** — ast と none は排反です。いずれか 1 つを選択します (デフォルトは ast)。reversepos は他のオプションと独立です。
  - ast — AsyTeX.asy を読み込みます。後述の ¥astLabel マクロを使うときは必須です。
  - none — AsyTeX.asy を読み込みません。できるだけ生のままの Asymptote を使いたいときに指定します。
  - reversepos — 図に文字列を入れるとき，LaTeX の picture 環境と Asymptote では上・下や左・右の指定が逆になります[5]。後述の ¥astput , ¥astLabel マクロでは LaTeX 流を採用していますが，reversepos を指定すれば Asymptote 流になります。

AsyTeX に対するパッケージオプションの設定方法も graphicx パッケージや他のパッケージと同じです。たとえば，sefalse と notrenew を指定したいなら

```
¥usepackage[sefalse,notrenew]{AsyTeX}
```

とします。

4 使い方その 2 — マクロについて

4.1 ASYpic 環境

ASYpic 環境は LaTeX の picture 環境を拡張したもので，内部に Asymptote ソースを書き込めます (書き方は後述)。また，図の (TeX が認識する) サイズなどを指定できます。

[5]たとえば配置先の点の右側に文字列を置くとき，LaTeX では「文字の左端を配置点に合わせる」と文字列主体で「左」と表現するのに対し，Asymptote では「配置点の右に文字列を合わせる」と配置点主体で「右」と表現します

使い方：

¥begin{ASYpic}<**横単位長**,**縦単位長**>(**幅**,**高さ**|**深さ**)(**横補正**,**縦補正**)

¥end{ASYpic}

または

¥begin{ASYpic}*<**横単位長**,**縦単位長**>(**幅**,**高さ**|**深さ**)(**横補正**,**縦補正**)

¥end{ASYpic}

必須の引数は**幅**と**高さ**。

省略したときのデフォルト値は，**横単位長** は ¥unitlength , **縦単位長** は **横単位長**，**深さ** は 0 , **横補正**，**縦補正** は 0 です。

図を含むボックスの (TEX が認識する) サイズは,

幅が **幅**× **横単位長**，高さが **高さ**× **縦単位長**，深さが **深さ**× **縦単位長**

で定まる値に設定されます。また，このボックスのベースライン左端の点が，内部の図の座標系でどの位置にあるかを指定するのが **横補正** と **縦補正** です。

なお，内部の図の座標系の単位長はもちろん **横単位長**，**縦単位長** です。この座標系を直接使うマクロは後述の ¥astput だけです。Asymptote の図は原点が位置しているだけで，必ずしも単位長は一致していません。

もう少し具体的にいうと，¥begin{ASYpic}* のように * つきで用いた場合，**横単位長** を w , **縦単位長** を h として Asymptote 内で参照できるようになりますが，単位長は MPpic 環境とは独立に設定できます。

また，¥begin{ASYpic} のように * 無しで用いた場合，**横単位長** を w , **縦単位長** を h として Asymptote 内で参照できるようにし，なおかつ unitlength(w,h) で Asymptote の座標系の単位長と ASYpic 環境の単位長を一致させます。

4.2 生モード

次のような形で ASYpic 環境内を使うことを生モードと呼ぶことにします。

¥begin{ASYpic}<1cm>(2,3)(0,0)% **ここの引数は適当**
¥openASY
(ここに生のままの Asymptote コード)
¥closeASY
(¥astput など TEX 側の命令)
¥end{ASYpic}

¥openASY と ¥closeASY の間は TEX の特殊文字の「特殊性」をすべて解除していますので，% や # などの特殊文字も普通の文字のように使えます。できるだけ自然なままの Asymptote コードを書きたいとき (たとえば Asymptote の関数などを作るとき) に便利なモードです。逆に言えばそれ以外にあまりメリットはありません。デメリットは,「% や # を TEX の意味では使えなくなる」「¥closeASY は必ず行頭から，空白等一切入れずに開始しなければならないため，ソースをインデントで見やすくすることができない」などが挙げられます。

ちなみに Asymptote で和文を取り込むのは結構面倒なので，LATEX の picture 環境に倣って ¥astput という命令を定義してあります。

使い方：

¥astput(**横座標**,**縦座標**)[**配置方向**]<**横補正**,**縦補正**>{**文字列**}

[**配置方向**] と <**横補正**,**縦補正**> は省略可能です。省略したときは前者は [c] , 後者は <0mm,0mm> が指定されたのと同じ結果になります。

(**横座標**,**縦座標**) は文字列の配置先の座標 (ASYpic の座標系)。

配置方向 が c のときは文字列の中央を配置先の座標に合わせます。B のときは文字列のベースライン中央，b のときは文字列の下辺中央，t のときは文字列の上辺中央，r のときは文字列の右辺中央，l のときは文字列の左辺中央を，配置先の座標に合わせます。tl のように組み合わせれば文字列左上頂点を配置先の座標に合わせます。同様に他の (矛盾しない) 組合せも可能です。

b と t , r と l の意味は，パッケージオプションの reversepos で逆転できます。

<**横補正**,**縦補正**> は配置点から少し離して文字列を置きたいときに使います。移動量を実寸法で指定します。

{**文字列**} は実際には文字列でなくてもかまいません — TEX で表現できるものなら何でも可です。

4.3 融和モード

生モードが「Asymptote ソース」と「TEX ソース」を完全に分離していたのに対し，融和モードは「あくまで TEX ソースが主で，Asymptote ソースが間借りしている」感じ。あるいは，TEX ソースの間にぽつぽつと Asymptote に送るソースが混じっている感じです。

おもに ¥sendASY と ¥astLabel の 2 つのマクロで実現されています。

使い方：

¥sendASY{ Asymptote に送るソース列 }　はソース列を展開せずにそのまま送ります。
¥sendASY*{ Asymptote に送るソース列 }　もソース列を展開せずにそのまま送ります。
¥sendASY**{ Asymptote に送るソース列 }　はソース列を完全に展開して送ります。

* なしと * 1 つの違いですが，* なしの方では % は通常の TEX の意味で解釈されるため，TEX の方で % 以降はコメントとしてカットされて Asymptote に送られます。しかし Asymptote では % が余りを表す演算子 (例：14 % 3 = 2) に使われますので割と使用頻度が高いものです。同様に ^^ も TEX の特殊文字でそのままでは使えないのですが，これも Asymptote では比較的使用頻度の高い通常の演算子です。そこで,「* なし」に「% と ^^ の特殊性のみの解除」を付加したものを用意しました。これが「* 1 つ」です。

使い方：

¥astLabel{**配置点**}[**配置方向**]<**横補正**,**縦補正**>{**文字列**}

配置点 が ASYpic の座標系ではなく Asymptote の座標系であること，**配置点** を Asymptote ソース内の変数名や関数式などで与えることができること以外は ¥astput と同じ使い方です。ただし，こちらは **文字列** 部分の背景をデフォルトで白抜きしてくれます。

文字列 の配置に対し，さらに細かい指示を出すこともできます。

¥astLabel[key=val **列**]{**配置点**}{**文字列**}

key=val 列に指定できるキーの種類と意味は次の通りです。

- origin — **配置方向** に相当します。たとえば **配置方向** が b であることを origin=b と表します。
- x — ラベルの文字列のどの位置を配置点に合わせるかをもっと細かく指定できます。文字列のベースライン左端からの「横座標」を実寸法で指定します。
- y — 同上。文字列のベースライン左端からの「縦座標」を実寸法で指定します。
- xshift — **横補正** に相当します。たとえば **横補正** が 1mm であることを xshift=1mm と表します。
- yshift — **縦補正** に相当します。
- labelsep — ラベル文字列は，実際には少し空白を周囲につけて出力されます。その空白の幅を指定できます。デフォルトは 1pt です。
- filltype — ラベルの背景をどうするか指定できます。Asymptote の **filltype** 値変数・関数の形で指定します。指定がなければ白で消去 (Fill(white) に相当) します。
- angle — ラベルを回転させられます。その回転角を指定するキーです。デフォルトは 0 です。

4.4 その他

¥astRenew と ¥astRenew* は通常 ASYpic 環境に入ってすぐ — ¥begin{ASYpic} から始まる行の末尾辺り — に入れます。パッケージオプションが renew , notrenew いずれであるかにかかわらず，* 無しの方は強制的に renew の意味に，* ありの方は強制的に notrenew の意味に変更します。効果範囲は ASYpic 環境に入ってから使った場合はその ASYpic 環境が終わるまで (つまりその図だけ) , ASYpic 環境の外で使った場合はそれ以降すべての図に対してです。

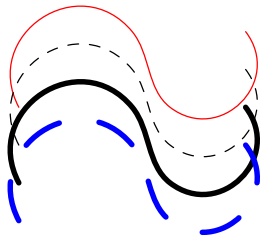
¥defaultASYinput は，以降に現れるすべての図の Asymptote ソースに同じコードを入れたいときなどに使います。たとえば以降すべての図で import graph; を行いたいときは，¥defaultASYinput{import graph;} とします。

¥mptAke は MEPOTEX からの完全流用で，段落の幅を調整して図のためのスペースを空けるマクロです。詳しくは MEPOTEX のマニュアルを御覧下さい。

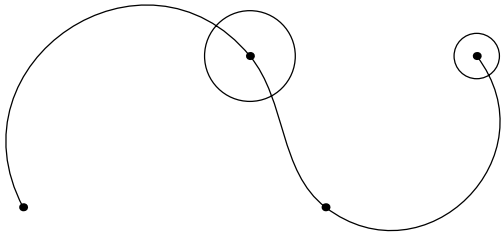
5 サンプル

まだろくなサンプルがありませんが，とりあえず 10 個ほど挙げておきます。

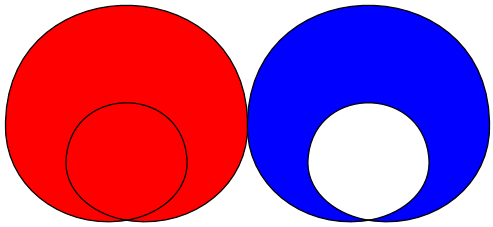
```
¥begin{ASYpic}<1cm>(0,0)(4,-3)
  ¥sendASY{pair[] z = {(0,0),(1.5,1),(2,0),(3,1)};
    path p=z[0]..z[1]..z[2]..z[3];
    draw(p,red);
    draw(shift(0,-0.5)*p,dashed);
    draw(shift(0,-1)*p,linewidth(2pt));
    draw(shift(0,-1.5)*p,blue+dashed+2pt);
  }
¥end{ASYpic}
```



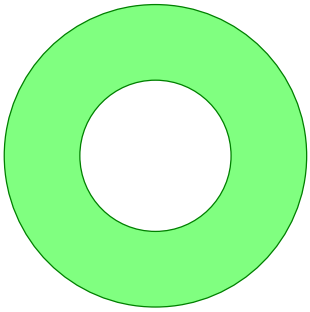
```
¥begin{ASYpic}*<1cm>(0,0)(7,-2)
  ¥sendASY{unitsize(2);
    pair[] z = {(0,0),(1.5w,h),(2w,0),(3w,h)};
    path p=z[0]..z[1]..z[2]..z[3];
    path q=scale(0.3w)*unitcircle;
    draw(shift(z[1])*q); draw(z[3],q);
    draw(p);
    dot(p);
  }
¥end{ASYpic}
```



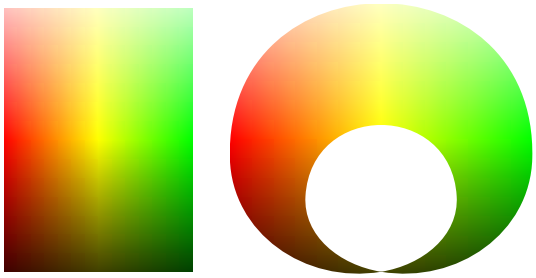
```
¥begin{ASYpic}<8mm>(0,0)(4,-0.5)
  ¥sendASY{pair[] z = {(0,0),(1,1),(-1,1),(2,1.5),(-2,1.5)};
    path p=z[0]..z[1]..z[2]..z[0]..z[3]..z[4]..cycle;
    filldraw(shift(-2,0)*p,red+zerowinding);
    filldraw(shift(2,0)*p,blue+evenodd);
  }
¥end{ASYpic}
```



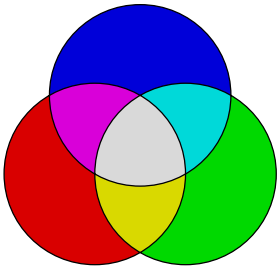
```
¥begin{ASYpic}<1cm>(0,0)(3,-2)
  ¥sendASY*{path p=unitcircle;
            path q=scale(2)*unitcircle;
            filldraw(p^^q,lightgreen+evenodd,deepgreen);
  }
¥end{ASYpic}
```



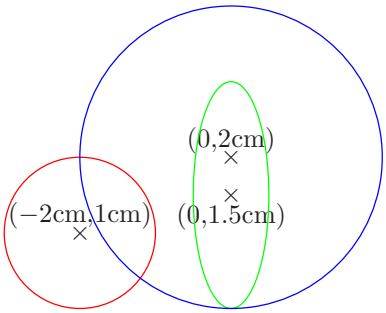
```
¥begin{ASYpic}<1cm>(0,0)(7,-0.8)
  ¥sendASY{pair[] z = {(0,0),(1,1),(-1,1),(2,1.5),(-2,1.5)};
    path p=z[0]..z[1]..z[2]..z[0]..z[3]..z[4]..cycle;
    pen[][] col={{palered,paleyellow,palegreen},
                 {red,yellow,green},
                 {darkred,darkolive,darkgreen}};
    latticeshade(scale(2.5,3.5)*unitsquare,col);
    latticeshade(shift(5,0)*p,evenodd,col);
  }
¥end{ASYpic}
```



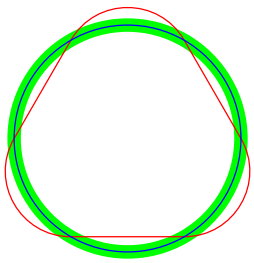
```
¥begin{ASYpic}<1.2cm>(0,0)(2,-2)
  ¥sendASY{picture pic = new picture;
    path[] pth = {circle(0,1), circle(right,1), circle(dir(60),1),circle(0,1)};
    pen[] pn = {0.85red,0.85green,0.85blue,0.85red};
    for(int i=0; i<3; ++i){fill(pth[i],pn[i]);}
    for(int i=0; i<3; ++i){fill(pic,pth[i],pn[i]+pn[i+1]);
      clip(pic,pth[i+1]);  add(pic);  erase(pic);}
    fill(pic,pth[0],pn[0]+pn[1]+pn[2]);
    clip(pic,pth[1]);  clip(pic,pth[2]);  add(pic);
    for(int i=0; i<3; ++i){draw(pth[i]);}
  }
¥end{ASYpic}
```



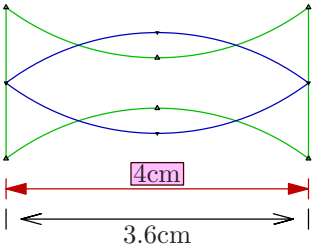
```
¥begin{ASYpic}*<1.2cm>(0,0)(2,-2)
  ¥sendASY{picture pic = new picture;
    unitsize(pic,0.5,1.5); unitsize(2);
    draw(pic,circle((0,1cm),1cm),red);  add(pic,(-1cm,0)); erase(pic);
    draw(pic,circle((0,1cm),1cm),green);  add(pic.fit());  erase(pic);
    draw(pic,circle((0,1cm),1cm),blue);   add(pic);
  }
  ¥astput(-2,1) {$¥times$}  ¥astput(-2,1)[b]<0mm,0.5mm>{($-2$cm,1cm)}
  ¥astput( 0,1.5){$¥times$} ¥astput(0,1.5)[t]<0mm,-1mm>{(0,1.5cm)}
  ¥astput( 0,2) {$¥times$}  ¥astput(0,2)[b]<0mm,0.5mm>{(0,2cm)}
¥end{ASYpic}
```




```
¥begin{ASYpic}<1.5cm>(0,0)(2,-2)
  ¥sendASY{pair[] z;
    for(int i=0; i<6; ++i){z[i]=dir(60i);}
    path p=z[0]..z[1]..z[2]..z[3]..z[4]..z[5]..cycle;
    path q=z[0];  for(int i=1; i<6; ++i){q=q..z[i];} q=q..cycle;
    guide g=z[0]; for(int i=1; i<6; ++i){g=g..z[i];} g=g..cycle;
    draw(p,green+linewidth(5pt)); draw(q,red); draw(g,blue);
  }
¥end{ASYpic}
```



```
¥begin{ASYpic}<2cm>(0,0)(2,-1)
  ¥sendASY{pair[] z = {(0,0),(1,1/3),(2,0)};
    path p=z[0]..z[1]..z[2];
    path[] pth={p--(shift(N)*yscale(-1)*reverse(p))--cycle,
                shift(0.5N)*p--(shift(0.5N)*yscale(-1)*reverse(p))--cycle,
                shift(0.2S)*(z[0]--z[2]), shift(0.4S)*(z[0]--z[2])};
    draw(pth[0],heavygreen,Mark[1]);
    draw(pth[1],heavyblue,MarkFill[4]);
    draw(pth[2],heavyred,Arrows,Bars);
    draw(pth[3],Bars,Arrows(SimpleHead),TrueMargin(2mm,2mm));
  }
  ¥astLabel[origin=b,yshift=0.5mm,
    filltype={FillDraw(palemagenta,darkred)}]{midpoint(pth[2])}{4cm}
  ¥astLabel{midpoint(pth[3])}[t]<0mm,-0.5mm>{3.6cm}
¥end{ASYpic}
```




```
¥begin{ASYpic}*<1cm>(0,0)(4,-3)
  ¥sendASY{size(200);
    pen[] p={red,green,blue,magenta,cyan};
    pair[] z={(-1,0),(0,0),(0,1),(1,0),(1,-1)};
    int[] edges={0,0,0,1,2};
    gouraudshade(z[0]--z[1]--z[4]--z[3]--z[2]--cycle,p,z,edges);
  }
¥end{ASYpic}
```