Zpravodaj Československého sdružení uživatelů TeXu

Denis Roegel

Romantika v METAPOSTu po francouzsku: líbající se kružnice

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 32 (2022), No. 1-4, 18–34

Persistent URL: http://dml.cz/dmlcz/151106

Romantika v METAPOSTu po francouzsku: líbající se kružnice

Denis Roegel

Když se kružnice potkají, políbí se. Pokud se líbají tři, další se pokusí přidat a políbit je všechny naráz. V tomto článku se podíváme na tuto situaci podrobně z pohledu METAPOSTu a poradíme jim, jak se mají líbat, v libovolné pozici a velikosti. Naučíme je také líbat se rekurzivně.

Klíčová slova: METAPOST, Apolloniova úloha, navzájem dotýkající se kružnice, vnitřní a vnější Soddyho kružnice, Eppsteinova konstrukce, Apolloniův fraktál

1. Úvod

Apollonios z Pergy (3. století před naším letopočtem) byl řecký geometr, mimo jiné také autor díla *Conica* – Pojednání o kuželosečkách. Je mu připisováno, že jako první použil termíny *elipsa*, *parabola* a *hyperbola*. Jeho kniha *De Tactionibus* (*O dotycích*), citovaná Papposem Alexandrijským, definuje problém tečen jako problém nalezení kružnice dotýkající se třech dalších objektů, v libovolné kombinaci bodů, přímek a kružnic. Apollonius ukázal, jak lze tento problém vyřešit pomocí pravítka a kružítka. Nyní tyto úlohy známe pod označením Apolloniovy úlohy. Pokud jsou ony objekty tři kružnice, existuje až osm různých řešení [2].[1]

Za situace, že se tři kružnice dotýkají zvnějšku, se těchto osm řešení redukuje jen na dva případy, a to na vepsanou (vnitřní) a opsanou (vnější) dotýkající se kružnici, známé jako vnitřní a vnější Soddyho kružnice (viz Obrázek 1).

René Descartes našel jednoduché analytické řešení. Křivosti $e_1 = \frac{1}{r_1}$, $e_2 = \frac{1}{r_2}$, $e_3 = \frac{1}{r_3}$ tří dotýkajících se kružnic jsou ve vztahu ke křivosti e_4 Soddyho kružnice zapsané pomocí rovnice

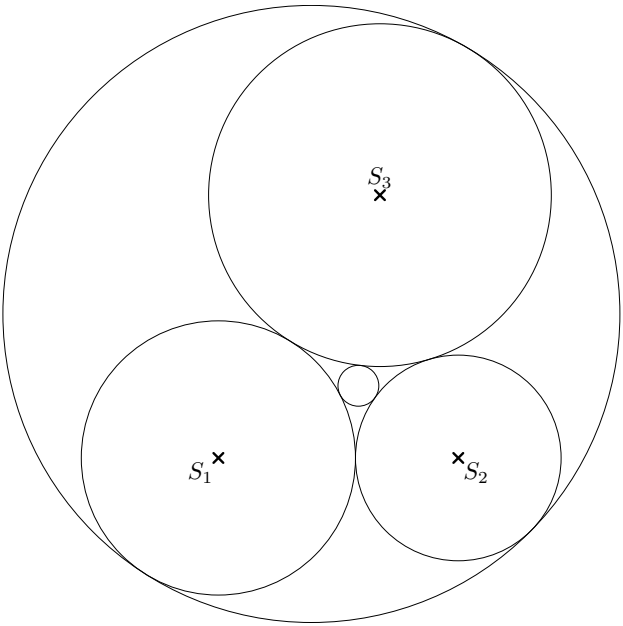
$$2(e_1^2 + e_2^2 + e_3^2 + e_4^2) = (e_1 + e_2 + e_3 + e_4)^2 \,. \tag{1}$$

V rovnici jsou e_1, e_2 a e_3 zadány, řešení e_4 jsou dvě. Kladné řešení představuje vnitřní Soddyho kružnici a záporné řešení vnější Soddyho kružnici, jejíž poloměr je dopočítán pomocí $-\frac{1}{e_4}$. Analytické řešení může být použito k iterování nákresu, musíme si však dát pozor na přetečení u aritmetických operací. Vnější Soddyho

Z anglického originálu *Kissing Circles: A French Romance in METAPOST* [1] přeložil Pavel Stříž.

[1] Soddyho kružnice buď nezahrnuje žádnou ze tří kružnic (1 řešení), jednu z nich (3 řešení), dvě z nich (3 řešení), nebo všechny tři (1 řešení). Celkem je tedy osm řešení; jejich grafické vyobrazení viz [2, str. 16]. (pozn. překl.)

DOI: 10.5300/2022-1-4/18



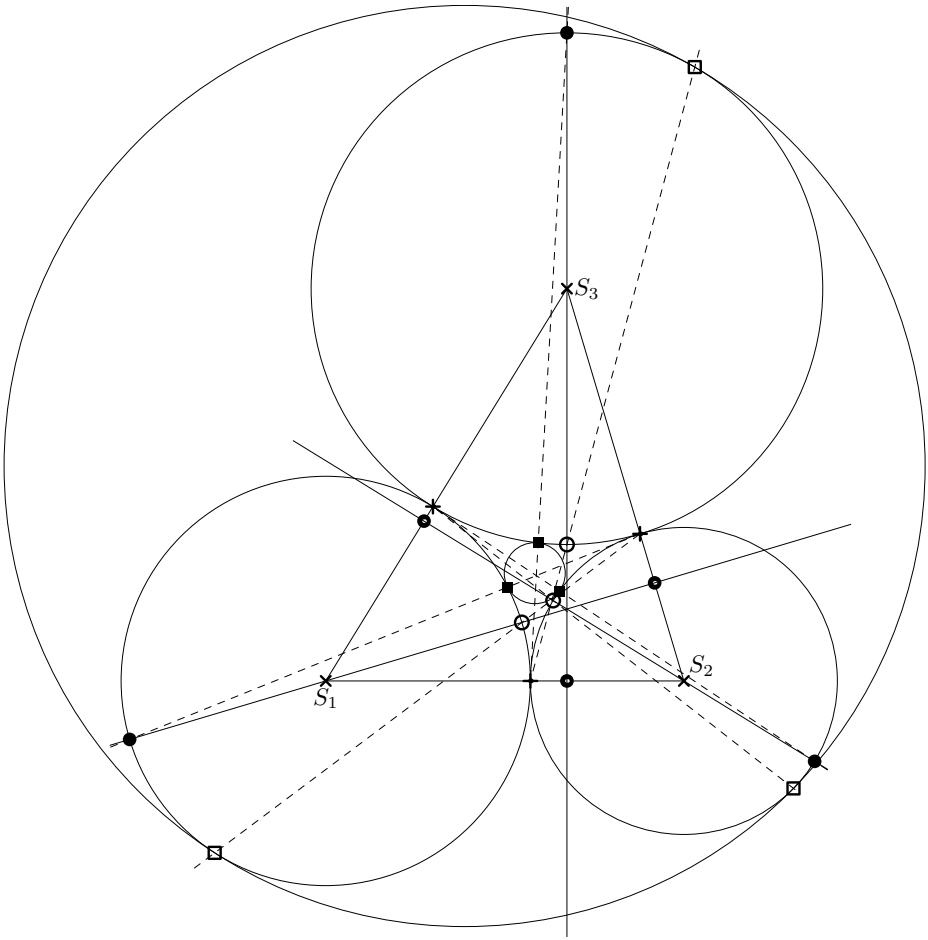
Obrázek 1: Vnitřní a vnější Soddyho kružnice ke kružnicím se středy S_1, S_2 a S_3.

kružnice obsahuje uvnitř další kružnice, a proto má malou hodnotu křivosti. Jak vměstnáváme stále menší a menší kružnice, dostáváme stále větší hodnoty křivosti. METAPOST sám o sobě není příliš vhodný na zpracování příliš malých ani příliš velkých čísel, v takovém případě je výhodnější využít geometrický přístup bez nutnosti výpočtů.

2. Konstrukce Davida Eppsteina

David Eppstein publikoval v roce 2001 nový způsob sestrojení vnitřní a vnější Soddyho kružnice [3]. Naším cílem nebude ověřit, zda je tato konstrukce správná, ale zjistit, jak nejlépe ji lze využít v METAPOSTu, a to zejména co nejobecněji.

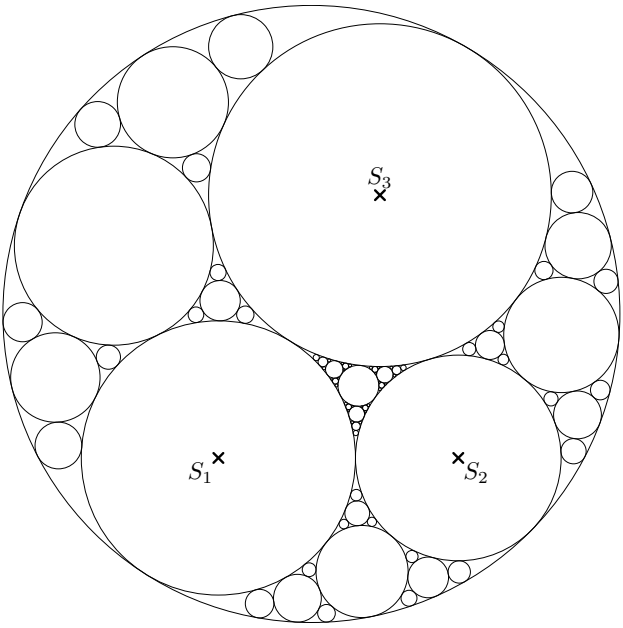
Eppsteinův postup si nyní popíšeme. Zadány máme tři navzájem dotýkající se kružnice se středy S_1, S_2 a S_3 (Obrázek 2). Z každého středu je spuštěna kolmice na protější stranu trojúhelníku. Paty získaných výšek jsou označeny puntíkem s průhlednou tečkou uprostřed (⊙). Tato protažená výška protne kružnici, jejímž středem prochází, na dvou místech. Bližší k průsečíku těchto výšek označíme značkou prstence (○), ty vzdálenější označíme puntíkem (●). Nyní každá z těchto



Obrázek 2: Eppsteinova konstrukce.

dvou značek může být spojena s bodem dotyku zbylých dvou kružnic, který označíme vertikálním křížkem (+). Přímka procházející přes body označené puntíkem a vertikálním křížkem protne původní kružnice ještě v bodech, které označíme plnými čtverci (■). A tyto tři body označené plnými čtverci tvoří body dotyku vnitřní Soddyho kružnice.

Podobně přímka procházející přes body označené prstencem a vertikálním křížkem protne původní kružnice ještě v bodech, které označíme prázdnými čtverci (□). Tyto tři body tvoří body dotyku vnější Soddyho kružnice.



Obrázek 3: Apolloniův fraktál hloubky 3 s 83 kružnicemi.

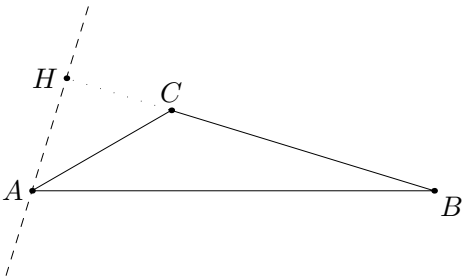
Tento postup lze použít k nalezení vnitřních dotýkajících se kružnic k vnější Soddyho kružnici a například kružnicím se středy S_1 a S_3. Takto lze sestrojit i Apolloniův fraktál (Obrázek 3).

3. Problémy se sestrojením a přípravy v METAPOSTu

Nalezení Soddyho kružnic je poměrně přímočaré, ačkoliv již v této fázi musíme dávat pozor na speciální případy. Skutečné problémy k vyřešení se objeví tehdy, jakmile budeme konstrukci iterovat, jinými slovy ve chvíli, kdy se rozmístění kružnic mění a objevuje se více situací. Hlavním zdrojem potíží je opakování se identických kružnic. Eppsteinovým postupem získáme šest bodů, musíme si však dát velký pozor na to, jak tyto body roztřídíme do dvou skupin po třech. Navíc musíme vyřešit situaci, která trojice bodů patří k té kružnici, kterou si zrovna přejeme nakreslit.

Začneme naši práci přípravou série robustních maker na typické dílčí úlohy. Některé ze zmíněných maker lze snadno využít v jiných programech.

• $A + \overrightarrow{CB}$ rotated 90

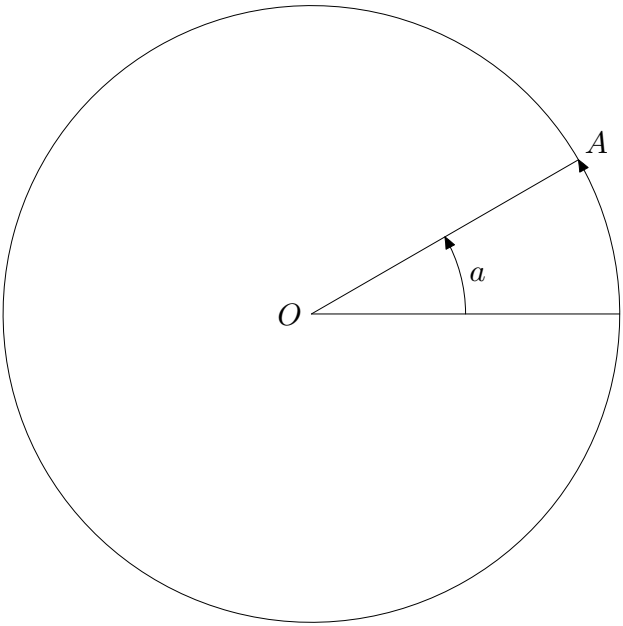


Obrázek 4: Nalezení výšky trojúhelníka ABC.

3.1. Výška trojúhelníku

Naše první makro (Obrázek 4) spočte výšku trojúhelníku ABC procházející bodem A. Výška nemusí protnout protější stranu trojúhelníku přímo, proto je dobrý plán použít konstrukci `whatever`. Makro tedy říká, že průsečík H leží *někde* ve směru úsečky $[B, C]$ a *někde* na protažené úsečce $[A, H]$, kde H je sestrojený bod za použití vektoru $\overrightarrow{BC}$. Makro nevrací průsečík H samotný, ale cestu (trajektorii) jdoucí dál za body A a H alespoň o délku r, což je hodnota, kterou makru poskytneme. Nápad spočívá v tom, že použijeme tyto cesty, abychom našli průsečík s původními kružnicemi, proto je potřeba cesty rozšířit alespoň o poloměr kružnice r a ještě o kousek víc, abychom si byli jisti, že průsečík bude nalezen. Jde o to, že dvě cesty, které mají společný bod, a ten by zároveň byl počátkem jedné z cest, by nemusel být METAPOSTem identifikován z důvodu chyb v zaokrouhlování.

```
vardef triangle_height(expr A,B,C,r)=
  save H,d;
  hide(
    pair H,d;
    H=whatever[B,C]
     =whatever[A,A+((B-C) rotated 90)];
    d=unitvector(H-A);
    )
  ((A-1.1r*d)--(H+r*d)) % výška spuštěná z vrcholu A
enddef;
```



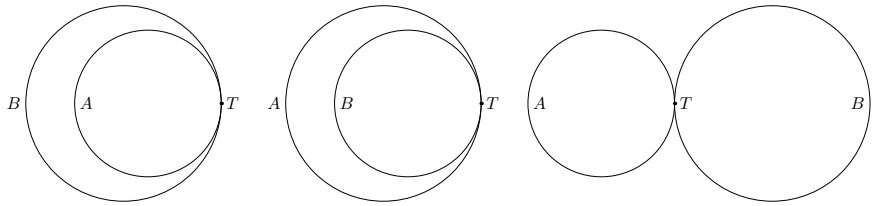
Obrázek 5: Kružnice otočená o a stupňů. Bod A je začátek i konec cesty (křivky) kružnice se středem O.

3.2. Kružnice

Kružnice můžeme získat pomocí makra `fullcircle` a zároveň bude toto makro použito k nalezení průsečíků. Kružnice však mohou být problém, neboť jejich průsečík s přímkou není obvykle jediný, a pokud si přejeme jen jediný průsečík, musíme zajistit, že je to ten, který potřebujeme. Další související problém je nespojitost kružnice jako křivky v METAPOSTu. Ačkoliv to není okem viditelné, kružnice vykreslená METAPOSTem má svůj začátek a konec. Je poměrně rozumné se této nespojitosti vyvarovat, neboť je to zdroj řady potíží.

Jedna konvenční cesta, jak se vyvarovat numerickému problému hledání průsečíku funkcí `instersectionpoint` u nespojité kružnice je otočit tuto kružnici tak, aby její nespojitost byla tam, kde to nebude vadit (Obrázek 5). Vypadá to, jako kdyby tento krok neměl smysl, ale využívá se toho, že funkce vrací první průsečík, který minimalizuje parametry cesty. Pokud tedy zajistíme, že kružnice bude otočena tak, že průsečík s minimálním parametrem je ten chtěný, budeme sladce odměněni.

Z tohoto důvodu si naprogramujeme makro pro kružnici se středem O, poloměrem r a otočenou o úhel a.



Obrázek 6: Tři obecné případy dotyků.

```
def circle(expr O,r,a)=
  (fullcircle scaled 2r
              rotated a shifted O)
enddef;
```

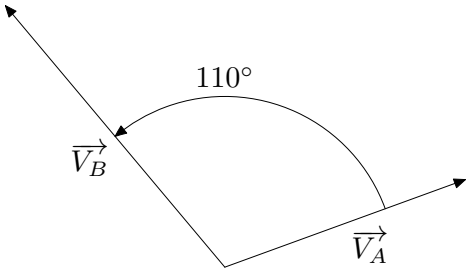
3.3. Body dotyku

Připravíme makro na výpočet bodu dotyku mezi dvěma kružnicemi, o kterých víme, že se dotýkají právě v jednom bodu (Obrázek 6). Makro navíc musí vyřešit situaci, kdy jedna kružnice leží uvnitř druhé. To nastane, když je vzdálenost mezi středy kružnic menší než oba poloměry. Potom kružnice s menším poloměrem leží uvnitř té druhé. Bod dotyku leží na přímce spojující středy kružnic. Například, když kružnice se středem S_a leží uvnitř kružnice se středem S_b, pak bod dotyku dostaneme vztahem

$$S_a + r_a \cdot \overrightarrow{S_aS_b} \,/\, \|\overrightarrow{S_aS_b}\| \,.$$

```
def tangency(expr Sa,ra,Sb,rb)=
 (if (arclength(Sa--Sb)<rb) or  % a uvnitř b nebo
     (arclength(Sa--Sb)<ra):    % b uvnitř a
      if ra<rb:                 % a uvnitř b
        Sa+ra*unitvector(Sa-Sb)
      else:                     % b uvnitř a
        Sb+rb*unitvector(Sb-Sa)
      fi
  else: circle(Sa,ra,0) intersectionpoint (Sa--Sb)
  fi)
enddef;
```

Když nahlédneme pozorně na zdrojový kód tohoto makra, zjistíme, že nám makro nezkolabuje, pokud není nalezen společný bod kružnic kvůli chybám v zaokrouhlování.



Obrázek 7: Úhel mezi dvěma polopřímkami definovaný vektory $\overrightarrow{V_A}$ a $\overrightarrow{V_B}$. Je upraven tak, aby patřil do intervalu $\langle 0, 180° \rangle$.

3.4. Úhel mezi dvěma polopřímkami

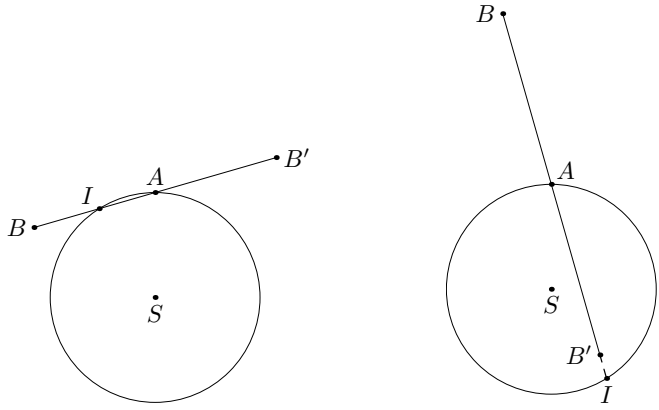
Úhel svíraný dvěma polopřímkami definovanými pomocí vektorů $\overrightarrow{V_A}$ a $\overrightarrow{V_B}$ lze získat tak, že použijeme konstrukci `angle` a zajistíme, aby byl v intervalu $\langle 0, 180° \rangle$ (Obrázek 7).

```
vardef angleof(expr Va,Vb)=
  save a;
  hide(
    a=angle(Vb)-angle(Va);
  forever:
    if a>=180:a:=a-180;fi;
    if a<0:a:=a+180;fi;
    exitif ((a<180) and (a>=0));
  endfor;
  )
  a % vypočtený úhel
enddef;
```

3.5. Průsečík přímky a kružnice

Pro Eppsteinovu konstrukci potřebujeme k danému průsečíku přímky a kružnice najít jejich druhý průsečík. Vymyslet makro pro tento požadavek, aby fungovalo pro všechny nastalé situace, není triviální záležitostí. Jedna ze situací, na kterou si musíme dát pozor, je tečna ke kružnici. Pak může nastat případ, kdy žádný průsečík není nalezen vinou zaokrouhlování.

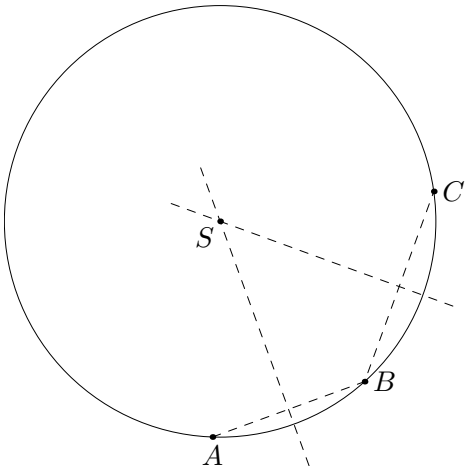
Když píšeme takové makro, musíme zároveň zajistit, že nám nezkolabuje, když budou dosti blízko sebe dva průsečíky (Obrázek 8). Máme-li bod A ležící na kružnici a jiný bod B, pak náš postup bude prvně nalézt úhel mezi vektorem $\overrightarrow{SA}$ a vektorem $\overrightarrow{AB}$. Pokud je tento úhel mezi 89 a 91 stupni, můžeme předpokládat, že přímka AB je tečnou kružnice. I v případě, kdy to tečna není, budou průsečíky dosti blízké a můžeme je ignorovat. V takovém případě makro vrací bod A.



Obrázek 8: Průsečík přímky a kružnice. Kružnice, bod na kružnici A a další bod B jsou zadány. Hledáme další průsečík I. Kratší ze vzdáleností $|SB|$ a $|SB'|$ určuje, na které straně poloroviny s hraniční přímkou SA, resp. na které z polopřímek AB, AB' leží I.

Pokud tomu tak není, najdeme druhý průsečík tak, že porovnáme vzdálenost středu kružnice se dvěma body B a B' ležícími na přímce AB symetricky vzhledem k bodu A. Druhý průsečík pak leží na polopřímce $\overrightarrow{AE}$, kde E je ten z bodů B a B', který je blíže ke středu kružnice. Mírně otočíme kružnicí proti směru hodinových ručiček tak, abychom se vyhli situaci, kdy funkce `intersectionpoint` vrátí průsečík A namísto průsečíku I. To by odpovídalo A s příliš velkým parametrem.

```
vardef intersection_circle_line(expr S,r,A,B)=
  save a,BP,I,uv;
  hide(
    pair BP,I,uv;
    a=abs(angleof(B-A,A-S)-90);
    if a<1: I=A;
    else:
      BP=A+(A-B);
      if arclength(S--B)<arclength(S--BP):
        uv=unitvector(A-B);
        I=circle(S,r,angle(A-S)+1) intersectionpoint
           ((B-2.1r*uv)--(1.1[B,A]));
      else:
        uv=unitvector(A-BP);
        I=circle(S,r,angle(A-S)+1) intersectionpoint
          ((BP-2.1r*uv)--(1.1[BP,A]));
```



Obrázek 9: Kružnice procházející třemi zadanými body A, B a C.

```
      fi;
    fi;)
  I % výsledný průsečík
enddef;
```

3.6. Kružnice procházející třemi body

Závěrečná fáze Eppsteinovy konstrukce bere na vstupu tři body a vykreslí kružnici procházející těmito body (Obrázek 9).

Slouží k tomu makro `circle_through`, které spočítá střed kružnice jako průsečík os dvou úseček. Využívá k tomu pomocné makro `whatevermedian`, které vrací neznámý bod na ose zadané úsečky.

```
def whatevermedian(expr A,B)=
  whatever[.5[A,B],
           .5[A,B]+(B-A) rotated 90]
enddef;
def circle_through
      (expr A,B,C)(text S)(text r)=
  S=whatevermedian(A,B)
   =whatevermedian(B,C);
  r=arclength(A--S);
  draw fullcircle scaled 2r shifted S;
enddef;
```

3.7. Existence průsečíku dvou úseček

Bude užitečné si připravit makro, které zjistí, zdali existuje, průsečík dvou úseček (ne přímek). Relativně snadná cesta k dosažení tohoto cíle je připravit si booleovskou verzi makra `intersectionpoint`. Vrátí `true` (pravda) místo průsečíku a `false` (nepravda) místo chybového hlášení METAPOSTu.

```
secondarydef p intersectionpoint_b q =
  begingroup save x_,y_;
    (x_,y_)=p intersectiontimes q;
    if x_<0:
      false
    else: true
    fi
  endgroup
enddef;
```

3.8. Vnitřní a vnější dotyky kružnic

Připravíme si ještě následující makro, které načte čtyři různé středy kružnic. Kružnice b a c jsou vně tečné a kružnice d je tečná k těmto dvěma vnitřně. Kružnice a je tečná vně k b a c, ale je tečná vnitřně vůči d. Jinými slovy řečeno je kružnice d vnější Soddyho kružnicí ke kružnicím a, b a c. (Obrázek 1)

A naopak, když zadáme kružnice b, c a d, pak Eppsteinova konstrukce spočte dvě kružnice, z nichž jedna je kružnice a. Ukazuje se, že určení bodu na konkrétní kružnici je záležitostí existence průsečíku úseček $[S_a, S_d]$ a $[S_b, S_c]$, kde S_a, S_b, S_c a S_d jsou středy příslušných kružnic. Tyto dvě kružnice tečné vně ke kružnicím b a c, ale tečné vnitřně ke kružnici d, nazveme *vnitřní*, pokud bude dříve zmíněný průsečík existovat, a *vnější*, pokud ne. Odpovídá to středu vnější Soddyho kružnice tehdy, když neexistuje průsečík dříve zmíněných úseček.

Z hlediska praktického rozhodování to vypadá tak, že vnitřní kružnice ze zvažovaných dvou je ta menší.

```
def is_inner(expr Sa,Sb,Sc,Sd)=
  ((Sb--Sc) intersectionpoint_b (Sa--Sd))
enddef;
```

3.9. Směrnice úsečky

Makro `slope` spočte směrnici zadané úsečky vyjádřenou jako úhel. Tohle se nám bude hodit ve chvíli, kdy budeme otáčet kružnici o potřebný úhel.

```
def slope(expr p)=
  angle((point 1 of p)-(point 0 of p))
enddef;
```

4. Hlavní makro

Vstupem pro hlavní část makra jsou středy čtyř kružnic, jejich poloměry a hloubka rekurze. První tři kružnice se navzájem vně dotýkají a hledáme u nich obě Soddyho kružnice. Čtvrtá kružnice nemá přímý význam a přiřadíme jí zápornou délku poloměru.

Během iteračního algoritmu nastanou dvě situace. První případ je ten, kdy kružnice mají navzájem vnější dotyk a hledá se vnitřní Soddyho kružnice. To znamená, že čtvrtá kružnice nebude mít pro nás význam i nadále.

Druhý případ je hraniční situací, kdy první kružnice je vnější Soddyho kružnice, druhá a třetí kružnice mají spolu vnější dotyk a zároveň k Soddyho kružnici vnitřní dotyk. Čtvrtá kružnice má vnější dotyk k druhé a třetí a zároveň vnitřní dotyk ke kružnici první. Čtvrtá kružnice již byla nakreslena, ale musíme ještě zjistit, která je ta poslední kružnice, která ještě nebyla nakreslena.

4.1. Výpočet bodů dotyku a výšek trojúhelníka

Zjištění bodů dotyku a výšek trojúhelníka je přímočará záležitost s využitím dříve nadefinovaných vztahů.

```
vardef tangent_circles(expr Sa,Sb,Sc,So,
           ra,rb,rc,ro,n)=
  save S,T,r,ht,Ihc,Ilc;
  pair S[];     % středy kružnic
  pair T[][];   % body dotyku
  numeric r[];  % poloměry
  path ht[];    % výšky trojúhelníka
  pair Ihc[][]; % průsečíky výšek a kružnic
  pair Ilc[];   % výsledné souřadnice průsečíků
  S1=Sa; S2=Sb; S3=Sc;
  r1=ra; r2=rb; r3=rc;
  T[1][2]=tangency(S1,r1,S2,r2);
  T[2][3]=tangency(S2,r2,S3,r3);
  T[3][1]=tangency(S3,r3,S1,r1);
  ht1=triangle_height(S1,S2,S3,r1);
  ht2=triangle_height(S2,S1,S3,r2);
  ht3=triangle_height(S3,S1,S2,r3);
```

Abychom zjednodušili některé výrazy, nadefinovali jsme makro `next`.

```
def next(expr i)=
  (if i+1<4:i+1 else: 1 fi)
enddef;
```

4.2. Průměry kružnic a další průsečíky

Průsečíky výšek s kružnicemi je snadné získat; klíčem k úspěchu je správně je seskupit. V našem případě prvně seskupíme průsečíky, které získáme ve směru paty výšky tak, že drobně otočíme kružnice ve směru hodinových ručiček. Tím získáme body `Ihc[i][1]`, které jsou na obr. 2 označeny značkou prstenců.

Druhou skupinu bodů tvoří body na opačné polopřímce, které jsou zaznačeny puntíkem.

Poté jsou značky prstenců a puntíků spojeny přímkou s bodem dotyku protějších kružnic. Na původní kružnici díky vzniklé sečně získáme body označené prázdným a plným čtverečkem.

```
for i:=1 upto 3:
  % prstenec
  Ihc[i][1]
     =circle(S[i],r[i],slope(ht[i])-5)
      intersectionpoint ht[i];
  % puntík
  Ihc[i][2]-S[i]=S[i]-Ihc[i][1];
  % čtverec
  Ilc[i]=intersection_circle_line(
        S[i],r[i],
        Ihc[i][1],
        T[next(i)][next(next(i))]);
  % plný čtverec
  Ilc[3+i]=intersection_circle_line(
        S[i],r[i],
        Ihc[i][2],
        T[next(i)][next(next(i))]);
endfor;
```

4.3. Závěrečná fáze

Na závěr postupu, který je v nejobecnějším tvaru rekurze zmíněn níže, musíme zjistit, jestli je makro voláno poprvé, či nikoliv. Pokud je voláno poprvé (větev první), vykreslíme jen vnitřní a vnější Soddyho kružnice.

```
if firststep: % větev 1
  firststep:=false;
  % vnější Soddyho kružnice
  circle_through(Ilc1,Ilc2,Ilc3)
                (S4)(r4);
  % vnitřní Soddyho kružnice
  circle_through(Ilc4,Ilc5,Ilc6)
                (S5)(r5);
```

```
    % zahájení rekurze
    if n>0: % okolní kružnice
      tangent_circles(S4,S1,S2,S3,
                      r4,r1,r2,r3,n-1);
      tangent_circles(S4,S2,S3,S1,
                      r4,r2,r3,r1,n-1);
      tangent_circles(S4,S3,S1,S2,
                      r4,r3,r1,r2,n-1);
    fi;
  else:
    if ro<0: % větev 2
      circle_through(Ilc4,Ilc5,Ilc6)
                    (S5)(r5)
    else:    % větev 3
      if is_inner(So,S2,S3,S1):
        circle_through(Ilc1,Ilc2,Ilc3)
                      (S4)(r4);
      else:
        circle_through(Ilc4,Ilc5,Ilc6)
                      (S4)(r4);
      fi;
    fi;
  fi;
  % zahájení rekurze
  if n>0:    % větev 4
    if ro>0: % větev 5
      tangent_circles(S1,S2,S4,S3,
                      r1,r2,r4,r3,n-1);
      tangent_circles(S1,S3,S4,S2,
                      r1,r3,r4,r2,n-1);
      tangent_circles(S2,S3,S4,origin,
                      r2,r3,r4,-1,n-1);
    else:    % větev 6
      tangent_circles(S1,S2,S5,origin,
                      r1,r2,r5,-1,n-1);
      tangent_circles(S1,S3,S5,origin,
                      r1,r3,r5,-1,n-1);
      tangent_circles(S2,S3,S5,origin,
                      r2,r3,r5,-1,n-1);
    fi;
  fi;
enddef;
```

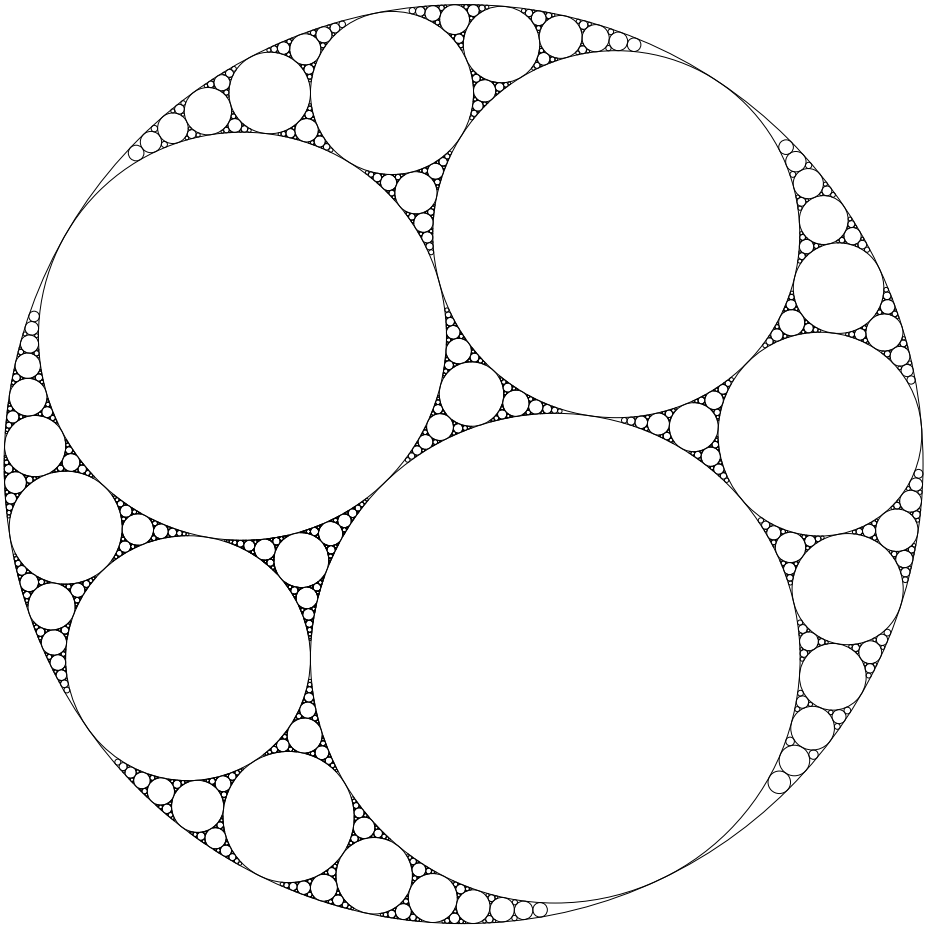
Jakmile máme vnější Soddyho kružnici se středem S_4, zavoláme makro na okolní kružnice. Každé takové iteraci poskytneme vnější Soddyho kružnici jako parametr první, poté dvě ze tří vnitřních kružnic, poté i třetí vnitřní kružnici. V průběhu dalšího volání makra bude aktivována třetí větev a nová kružnice bude vložena mezi dvě vnitřní kružnice a vnější Soddyho kružnici užitím dříve zmíněného kritéria.

Druhá větev se použije tehdy, když má být nalezena vnitřní Soddyho kružnice ze tří kružnic s vnějším dotykem.

Bez ohledu na to, co se již v programu událo, jakmile dosáhneme čtvrté větve, tak jsme již nalezli kružnici a nastanou dvě nové možnosti: buď se dotýkáme původní vnější Soddyho kružnice se středem S_1 (pátá větev), nebo je kružnice vnitřní Soddyho kružnicí tří kružnic s vnějším dotykem (šestá větev). Rozbor páté větve nás navede ke dvěma hraničním případům a jednomu vnitřnímu (vnitřní Soddyho kružnice). Rozbor šesté větve nás navede ke třem novým vnitřním případům (obvyklé vnitřní Soddyho kružnice). Všimněme si, že parametr `origin` chod makra neovlivní, když je poloměr záporný.

Hlavní makro se spouští následujícím kódem, jehož výsledkem je Apolloniův fraktál hloubky 7 (Obrázek 10).

```
pair S[];    % středy kružnic
numeric r[]; % poloměry kružnic
numeric n;   % hloubka fraktálu, konkrétně rekurze
n=7;         % hloubka rekurze, konkrétně fraktálu
S1=origin;
r1=4cm;
r2=3cm;
r3=6cm;
% nalezení středu kružnice S2
S2-S1=(r1+r2,0);
% dostáváme dvě řešení u kružnice se středem S3,
% ukládáme si jen jedno z nich
S3=circle(S1,r1+r3,0)
    intersectionpoint circle(S2,r2+r3,0);
draw circle(S1,r1,0);
draw circle(S2,r2,0);
draw circle(S3,r3,0);
firststep:=true;
tangent_circles(S1,S2,S3,origin,
                r1,r2,r3,-1,n);
```



Obrázek 10: Apolloniův fraktál hloubky 7 s celkem 6563 kružnicemi. Počet kružnic lze vypočítat ze vzorce $5 + 3 \cdot (3^n - 1)$, kde n udává hloubku rekurze. Pro $n = 0$ výpočtem zjistíme, že dostáváme pět kružnic, což jsou tři zadané, vnitřní a vnější Soddyho kružnice.

5. Závěr

Naše malá procházka napříč líbajícími se kružnicemi uvedla detaily geometrické konstrukce v METAPOSTu. Konkrétně jsme podrobně prošli Eppsteinovu konstrukci. Konečná verze kódu je jednoduchá, ale tato jednoduchost nebyla získána hned a bezprostředně. Navíc je kód podpořen robustností každého makra, které se navíc snaží být maximálně obecné. Zdrojový kód není nutné použít pouze pro uvedené hodnoty parametrů a měl by fungovat ve všech případech, kdy nedojde k překročení mezí METAPOSTu.

Odkazy

1. ROEGEL, Denis. Kissing Circles: A French Romance in METAPOST. *TUGboat*. 2005, roč. 26, č. 1, s. 10–17. Dostupné také z: `https://tug.org/TUGboat/Articles/tb26-1/tb82roegel.pdf`.
2. GISCH David; Ribando, Jason M. Apollonius' Problem: A Study of Solutions and Their Connections. *American Journal of Undergraduate Research*. 2004, roč. 3, č. 1, s. 15–25. ISSN 1536-4585. Dostupné také z: `http://www.ajuronline.org/uploads/Volume%203/Issue%201/31D-GischArt.pdf`.
3. EPPSTEIN, David. Tangent Spheres and Triangle Centers. *American Mathematical Monthly*. 2001, roč. 108, č. 1, s. 63–66. ISSN 0002-9890. Dostupné také z: `https://arxiv.org/abs/math.MG/9909152v1`.

Summary: Kissing Circles: A French Romance in METAPOST

When circles meet, they kiss. If three of them kiss, others can try to join and kiss all of them at once. In this article, we look at this problem from the METAPOST point of view, and we try to tell circles how to kiss, no matter their position and size. Recursive kissing will also be attempted.

Keywords: METAPOST, kissing circles, inner and outer Soddy circles, David Eppstein's construction, Apollonius' problem, Apollonius' gasket.

Denis Roegel, `roegel@loria.fr`
`http://www.loria.fr/~roegel`
LORIA – Campus Scientifique, BP 239
F-54506 Vandœuvre-lès-Nancy Cedex, France