

TikZ A ZOPÁR NÁPADOV

RUDOLF BLAŠKO (SK)

Abstrakt. Príspevok nemá za cieľ učiť čitateľa základné funkcionality TikZ-u, predpokladajú sa minimálne znalosti práce s TikZ-om. Cieľom je ukázať niektoré nápady na kreslenie obrázkov a ich automatizovanie pomocou jednoduchých makier, cyklov a zovšeobecnení. V príspevku je ukázaná tvorba jednoduchých schém, ktoré s obľubou kreslia tvorcovia rôznych modelov a algoritmov. Pre milovníkov teórie grafov je v príspevku uvedený jeden z možných návodov ako vykresliť úplný graf. Príspevok končí názornou ukážkou použitia knižnice TikZ-u, pomocou ktorej sa dajú nájsť priesečníky rôznych kriviek.

Kľúčové slová. L^AT_EX, TikZ, diagram, graph, intersection of curves.

TIKZ AND SOME IDEAS

Abstract. The article does not aim to teach the reader the basic functionalities of TikZ, minimal knowledge of working with TikZ is assumed. The aim is to show some ideas for drawing images and automating them using simple macros, cycles and generalizations. The article shows the creation of simple diagrams that are popularly drawn by creators of various models and algorithms. For lovers of graph theory, the article provides one of the possible instructions on how to draw a complete graph. The article ends with a demonstration of the use of the TikZ library, which can be used to find intersections of various curves.

Úvod

Na kreslenie obrázkov, schém, grafov matematických funkcií existuje mnoho viacmenej dobrých programov. Niektoré dokážu zobrazovať 2D aj 3D objekty, niektoré nie, niektoré dokážu vizualizovať pripravené zadané dáta, niektoré ich dokážu aj spracovať a následne zobrazovať, niektoré dokážu vykresľovať manažérske a programátorské schémy a vývojové diagramy. Niektoré používajú formát bitmapa a niektoré sú orientované na vektorovú grafiku. Takto by som mohol pokračovať ešte dlho, ale dôležité je, že so všetkými spomenutými činnosťami si poradí vektorový grafik TikZ. Spolu s balíčkami `pgf*` a tlačiarňou `pgfplot` predstavuje veľmi silnú podporu pre štatistické spracovanie a vizualizáciu údajov.

TikZ sa síce načítava ako štandardné balíčky pomocou príkazu `\usepackage`, ale skôr by sa dal charakterizovať ako účinná grafická nadstavba nad L^AT_EX-om. Ak používame L^AT_EX na vlastnom počítači (štandardne sa nainštaluje aj celá dokumentácia ku všetkým súčastiam T_EX-u), manuál si zobrazíme na príkazovom riadku spustením `texdoc tikz` (vo všetkých OS). Ak dokumentáciu

nemáme k dispozícii, môžeme sa obrátiť na domovský T_EX-ovský portál `ctan`. Uvedený dokument nájdeme na adrese <https://www.ctan.org/pkg/pgf> alebo samozrejme aj inde. Manuál TikZ-u stále s novými verziami priberá strany a pri aktuálnej verzii 3.1.10 má 1321 strán, čo tiež o niečom svedčí. Napríklad manuál verzie 1.18 z roku 2007 mal 405 strán (viď <https://www.bu.edu/math/files/2013/08/tikzpgfmanual.pdf>),

Veľmi dobrý pomocník pre prácu s TikZ-om, ktorý vytvoril Jean Pierre Castelyn v roku 2018 (TikZ a aj T_EX a L^AT_EX sú nadčasové záležitosti a aj staršie príručky a návody, aj tridsaťročné, majú čo povedať aj dnes) a nazýva sa Visual TikZ. Nájdeme ho na už spomínanom portáli `ctan` na adrese <https://mirrors.nic.cz/tex-archive/info/visualtikz/VisualTikZ-fr.pdf> alebo tiež na adrese <https://invimath.fri.uniza.sk/images/slides/LaTeX/VisualTikZ.pdf>. Tento dokument je písaný po francúzsky, ale to neprekáža ani našincovi neznalému reči, pretože je tam množstvo praktických ukážok, návodov a hlavne rôznych nastavení parametrov ku príkazom TikZ-u.

1. Krátky príklad z matematickej teórie grafov

Projekt TikZ, presnejšie povedané PGF/TikZ sa vyvja od roku 2005 a stále sa zdokonaľuje. Už dospel do štádia, že TikZ môžeme dať ako parameter do preambuly T_EX-ovského príkazu `\documentclass`, ale s triedou `standalone`, napr.

```
\documentclass[tikz,border=3mm]{standalone}
```

Je zrejmé, že parameter `border` definuje okraje dokumentu, ktorý bude vždy tvoriť iba jedna strana i keď divných rozmerov. Keď takto vytvoríme niekoľko obrázkov, zobrazia sa vedľa seba. Ak ich chceme zobrazit pod sebou, musíme situáciu riešiť pomocou tabuľky alebo inou násilnou formou.

Nasledujúci zdrojový kód ukazuje ako prakticky na dvoch riadkov vygenerovať úplný graf s n vrcholmi. Graf vygeneruje makro `\CompletGraph` a výsledok je zobrazený na obr. 1. Základom sú dva do seba vložené cykly `\foreach`, v prvom s premennou `\x` sa generuje n vrcholov a v druhom s premennou `\y` sa vygenerujú všetky hrany grafu.

```
\documentclass[tikz,border=3mm]{standalone}
\usetikzlibrary{arrows.meta}
\begin{document}
\newcommand\twodigits[1]{\ifnum#1<10 0#1\else #1\fi}
\colorlet{FaI}{black}    %% FaI nastavenie farby zobrazenia

%% definicia makra \UplnyGraf .. #1 pocet vrcholov %%
\def\UplnyGraf#1{\begin{tikzpicture}[rbA/.style={thick,FaI,fill=FaI!20,
draw,circle,minimum size=1.8em,inner sep=.1em,scale=.6}
,rbH/.style={FaI!80,line width=.5pt,{Stealth[scale=.6]}--{Stealth[scale=.6]}}]
%% generovanie vlastneho uplneho grafu %%
\foreach \x in {1,...,#1}
{
\node[rbA] (g\x) at (90-\x*360/#1:2cm) {\bf\twodigits{\x}};
\foreach \y in {1,...,\x} {\ifx\x\y \else\path (g\x) edge[rbH] (g\y);\fi}}
\end{tikzpicture}}
```

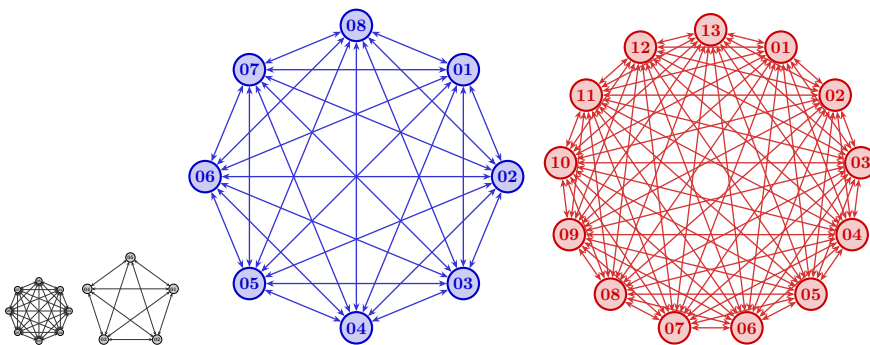
```

\end{tikzpicture}}

%% spustanie makra \UplnyGraf s roznymi parametrami %%
\scalebox{.2}{\UplnyGraf{8}}      %% L cierny graf n= 8 skala=20%
\scalebox{.3}{\UplnyGraf{5}}      %% LS cierny graf n= 5 skala=30%
\colorlet{FaI}{blue!80!black}\UplnyGraf{8}    %% PS modry graf n= 8
\colorlet{FaI}{red!80!black} \UplnyGraf{13}    %% P cerveny graf n=13
\end{document}

```

Kedže sa v makre kreslia medzi vrchmi grafu krajšie šípky **Stealth** ako sú štandardné, je v preambule zdrojového súboru nevyhnutné načítať knižnicu `\usetikzlibrary{arrows.meta}`. Tieto šípky sú definované v TikZ-ovskom štýle `rbH` a sú nastavené na orientovaný graf, ak chceme graf neorientovaný, stačí tento štýl prepísať na tvar `rbH/.style={FaI!80,line width=.5pt,-}`.



Obr. 1. Generovanie úplného grafu s rôznym počtom vrcholov

Zväčšenie a zmenšenie grafu je riešené pomocou príkazu `\scalebox{}{}`, ktorý má dva parametre. Prvý parameter určuje násobok zmeny veľkosti a druhý parameter obsahuje objekt tejto zmeny (napr. `.25`, resp. `0.25` predstavuje zmenšenie na 25 % pôvodnej veľkosti a parameter `4` znamená štvornásobné zväčšenie).

Vrcholy grafu sú označené vždy dvomi ciframi, čo rieši makro `\twodigits`, ktoré je definované príkazom:

```
\newcommand\twodigits[1]{\ifnum#1<10 0#1\else #1\fi}
```

Pred spustením makra `\UplnyGraf` je potrebné nejakým farbiacim príkazom definovať farbu zobrazenia `FaI`, napr. príkazom `\colorlet`, ktorý je definovaný v balíčku `xcolor`. Balíček `xcolor` nie je potrebné načítavať zvlášť, TikZ si ho aktivuje sám. Uvedené makro má iba jeden parameter, ktorý určuje počet vrcholov a musí byť prirodzené číslo. Jeho hodnota sa nikde netestuje a môže byť aj `#1=1`. Vtedy sa zobrazí iba jeden vrchol $\textcircled{01}$. Ak chceme zmeniť číslovanie na kladný smer, stačí prepísať v riadku

```
{\node[rbA] (g\x) at (90-\x*360/#2:3cm) {\bf\twodigits{\x}};
```

zátvorku `(90-\x*360/#2:3cm)` na tvar `(90+\x*360/#2:3cm)`. Pre nastavenie `(0+\x*360/#2:3cm)` sa číslovanie začne od 3. hodiny.

Pred vykresľovaním hrán sa testuje príkazom `\ifx \else \fi`, či sú vrcholy rôzne alebo nie. Hrana sa vykreslí iba v zápornom prípade pomocou príkazu:

```
{\ifx\x\y \else\path (g\x) edge[rbH] (g\y);\fi}
```

Namiesto predchádzajúceho príkazu môžeme s rovnakým efektom použiť príkaz `{\ifx\x\y \else\draw[rbH] (g\x)--(g\y);\fi}`, pričom $(g\backslash x)$ a $(g\backslash y)$ sa postupne nahrádzajú vrcholmi $g_1, g_2, \dots$.

2. Podpivníky a schémy

Pred niekoľkými rokmi som kreslil modely, ktoré pozostávali z niekoľkých poprepletaných kruhov, ktoré obsahujú jednotlivé pojmy. Pokúsil som sa tieto schémy zautomatizovať na ľubovoľný počet vrcholov s možnosťou jednoduchých zmien vnútorných textov a výsledkom je makro `\Podpivnik`.

Už dlhšie majú slovenčina a čeština niekedy problém s expandovaním pomlčky – v makrách. Používateľ $\text{\LaTeX}$ -u sa s tým najčastejšie stretne pri počiarkovaní v tabuľkách `tabular` pomocou `\cline`. Riešením je zadanie príkazu

```
\shorthandoff{-}
```

hneď za začiatok dokumentu, t. j. za príkaz `\begin{document}`. Toto je aj náš prípad. `TikZ` nevyžaduje žiadnu dodatočnú knižnicu.

Na písanie textov vo vnútri pivných tácoch som použil sanserifové písmo, ktoré je vhodnejšie na incidenčné materiály. Ak budeme takéto písmo používať v celom dokumente, môžeme ho nastaviť v preambule zdrojového dokumentu príkazom

```
\renewcommand*\familydefault{\sfdefault}
```

V tomto prípade sa nám príkazom `{\bf text}`, zobrazí **text**. Ak takéto písmo nepoužívame v celom dokumente a uvedený príkaz v preambule nie je, t. j. je nastavené klasické pätkové písmo, uvedeným príkazom dostaneme **text**. Ak chceme použiť tučné sanserifové písmo, môžeme použiť dvojicu prepínačov `\sf\bfseries`. Týmto spôsobom je riešené písmo v nasledujúcej schéme `\Podpivnik` pomocou makra `\def\bfs{\sf\bfseries}`. V nasledujúcich obrázkoch je použitá iná možnosť pomocou makra `\def\bfs{\bf\sfseries}`.

`\Podpivnik` je konštruovaný tak, aby sa dal ľubovoľne zmenšovať a zväčšovať. Zmena veľkosti sa zadáva ako prvý parameter škála **#1**. Rozmery v prostredí `tikzpicture` sú implicitne nastavené na **cm**, ale hrúbky čiar na body **pt**. Zmenou parametra `scala` sa zmení veľkosť obrázka, ale nezmení sa hrúbka čiar. To znamená, že pri veľkom zmenšení je čiara prihrubá a pri veľkom zväčšení naopak pritenká. Tento parameter **#1** sa hneď na začiatku makra `\Podpivnik` nastaví pomocou príkazu `\def\Psc{#1}` do parametra `\Psc`, ktorým sa násobia všetky nastavenia čiar. Je tu ešte jeden problém s textom, kde musí byť škála tiež násobená týmto parametrom, pretože škála nastavená na začiatku prostredia `tikzpicture` sa ignoruje v texte definovanom za `node`. To znamená, že týmto parametrom sú

vynásobené aj všetky škály v celom prostredí `tikzpicture` a veľkosť obrázku sa reguluje iba týmto parametrom.

Pred spustením vlastného makra `Podpivnik` musia byť definované všetky texty, ktoré sa použijú. Definujú sa v parametroch `\PP*`. Podobne, ako pri predchádzajúcom príklade, musíme pred spustením definovať farbu zobrazenia `FaI`.

```
%% text do stredneho kruhu a po okrajoch velkeho kruhu %%
\def\PPc{\begin{tabular}{@{}c@{}}OSS Conf\ 2025\ Žilina\end{tabular}}
\def\PPe{} \def\PPn{UTOROK}
\def\PPw{} \def\PPs{1.\,7.\,2025}
%% text do jednotlivych vnutornych kruhov ... pre n>10 treba dalsie \PPxi atd %%
\def\PPi{čapuje Rudo} \def\PPii{čapuje Aleš\ čapuje Milan\ čapuje Karol}
\def\PPiii{čapuje Pavel} \def\PPiv{čapuje Peter}
\def\PPv{čapuje Miro} \def\PPvi{čapuje Tomáš}
\def\PPvii{čapuje Milo} \def\PPviii{čapuje Michal}
\def\PPix{dnes sa nečapuje} \def\PPx{dnes sa nečapuje}
%% definicia pisma \bfs %% farby zobrazenia TaI %% pocneho textu \Poo %%
\def\bfs{\sf\bfs} \colorlet{FaI}{blue!60!black} \def\Poo{Pult~}
```

Druhý parameter `#2` nastavuje počet vnútorných kruhov, ak zvolíme počet väčší ako je 10, musíme dodefinovať aj textové zásobníky `\PP*` do požadovaného počtu. Makro funguje tak, že sa v cykle `\foreach` načítavajú postupne čísla od 1 po `#2`, arabské čísla sa premenia na malé rímske čísla, pridá sa prefix `PP` a celé sa do zloží do príslušného názvu príkazu `\PP*`, ktorý sa načíta do parametra `\PQs` a ten sa následne vytlačí pomocou `node`. Celý text sa tvorí konštrukciou

```
\def\PQs{\phantom.\smash{\begin{tabular}[#3]{@{}c@{}}
\scalebox{1.5}{\color{FaI}\Poo\MakeUppercase{\romannumeral\xi}}\
\csname PP\romannumeral\xi\endcsname \end{tabular}}\phantom.}
```

príčom `\csname PP\romannumeral\xi\endcsname` postupne vyrobí `\PPi`, `\PPii`, atď., a `\Poo\MakeUppercase{\romannumeral\xi}` vytvorí texty **Pult I**, **Pult II**...

Parametre `#3` a `#4` definujú zarovnanie textov v jednotlivých kruhoch navzájom (t zarovnané na vzdialený okraj od stredu, b zarovnanie na najbližší okraj od stredu, resp. c zarovnané na stred výšky textu) a vzdialenosť textu od stredu. Overené nastavenia týchto parametrov `[#3|#4]` sú `[b|.23]`, `[c|.31]`, resp. `[t|.38]`. Cyklus `\foreach` je použitý dvakrát z praktických dôvodov. Najskôr sa vykreslia jednotlivé kruhy aj s výplňou, aby potom netienili vlastné texty.

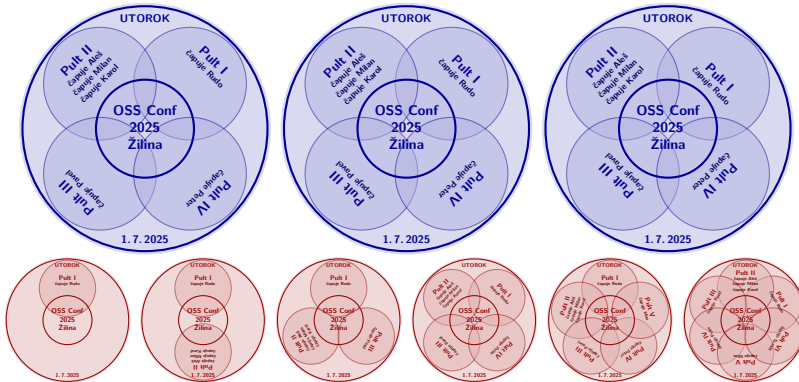
```
%% #1..skala -- zmenenie/zvacsenie #2..pocet vnutornych kruhov
%% #3..btc..pozicia textu od stredu #4..vzdialenost textu od stredu
\def\Podpivnik#1#2[#3|#4]{\def\Psc{#1}\colorlet{FaI}{black!40!FaI}%
\def\PPof{\ifodd#2 0\else 1\fi}\def\Ppoo{\ifodd#2 90\else 0\fi}%
\begin{tikzpicture}[FaI,scale=.6*\Psc,rbS/.style={scale=.75*\Psc},
,rbDm/.style={line width=2*\Psc pt},rbDv/.style={line width=4*\Psc pt},
,rbFi/.style={FaI!15,fill},rbFiii/.style={FaI,fill=FaI!30,line width=1*\Psc pt}]
\draw[rbFi,rbDv] (0,0) circle (.5\textwidth+2pt);
\draw[rbDm] (0,0) circle (.5\textwidth);
\foreach \xi in {1,...,#2}
{ \draw[rbFiii,opacity=.5]
({\PPoo+\PPo+180/#2+360*(\xi-1)/#2}:.26\textwidth) circle (.23\textwidth);}
\foreach \xi in {1,...,#2}
```

```

\def\Pqs{\phantom.\smash{\begin{tabular}[#3]{@{}c@{}}
\scalebox{1.5}{\color{FaI}\Poo\MakeUppercase{\romannumeral\xi}} \\\
\cename PP{\romannumeral\xi}\endcsname \end{tabular}}\phantom.}
\draw ({\PPoo+\Ppo*180/#2+360*(\xi-1)/#2}:#4\textwidth)
node[FaII,rbS,rotate={\PPoo+\Ppo*180/#2+360*(\xi-1)/#2-90}] {\bfs\Pqs};}
\draw[rBm] (0,0) circle (.2\textwidth) node[scale=1.2*\Psc] {\bfs\PPc};
\draw (0:.45\textwidth) node[scale=.95*\Psc,rotate=-90] {\bfs\PPe};
\draw (90:.45\textwidth) node[scale=.95*\Psc] {\bfs\PPn};
\draw (180:.45\textwidth) node[scale=.95*\Psc,rotate=90] {\bfs\PPw};
\draw (270:.45\textwidth) node[scale=.95*\Psc] {\bfs\PPs};
\end{tikzpicture}}

```

Na obrázku 2 sú postupne zobrazené modré podpivníky vygenerované s parametrami $\{.4\}\{4\}\{t|.38\}$, $\{.4\}\{4\}\{c|.31\}$, $\{.4\}\{4\}\{b|.24\}$ a pod nimi v spodnom riadku po zmene farby `\colorlet{FaI}{red!60!black}` červené s parametrami $\{.2\}\{x\}\{c|.31\}$, kde x predstavuje počet vrcholov od 1 po 6.



Obr. 2. Konferenčný Podpivnik v rôznych vyhotoveniach

Druhý podpivník by som skôr nazval podsudovník a nie je tvorený samostatným makrom. Skladá sa z piatich kruhov pravidelne umiestnených po obvodu kružnice, ktoré obsahujú po päť vnútorných kruhov s textom (obr. 3).

Najprv sú definované vlastné TikZ podmakrá, ktoré sa potom dajú spúšťať v prostredí `tikzpicture`. Majú názvy `\rbK` a `\rbM`. Vlastného textu je menej, ale rôzne rotuje vo vnútorných kruhoch. Názvy jednotlivých príkazov v makre `\rbM` sa opäť generujú v cykle pomocou `\cename TT\makealph\xi\endcsname`, skladajú z písmen TT a z písmen, na ktoré sa transformujú čísla pomocou príkazu `\makealph`, ktorý je nutné definovať v preambule príkazom

```

\makeatletter
\newcommand\makealph[1]{\@alph{#1}}
\makeatother

```

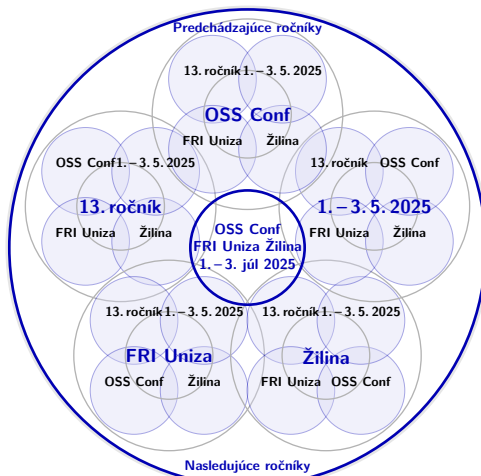
Pri kreslení samotného obrázku v príkaze cyklu `\foreach \x in {1,...,5} {\rbK at {90+\x*72:.7\DX} }` je nutná na konci aspoň jedná medzera alebo prechod na nový riadok medzi `\DX} }`, inak preklad havaruje.

Pred `\begin{tikzpicture}` musia byť definovaná farba kreslenia `FaI`, polomer obvodového kruhu `\DX` a samotné texty `\Ta` – `\Te`. Ešte je tam definované makro `\bfs` na prevod do tučného sanserif písma.

```
\colorlet{FaI}{blue!70!black}\def\FBI{\color{FaI}}
\newlength{\DX}\setlength{\DX}{14cm}
\def\Ta{OSS Conf} \def\Tb{1.\,--\,3.\,5.\,2025} \def\Tc{13.\,ročník}
\def\Td{FRI Uniza} \def\Te{Žilina} \def\bfs{\bf\sffamily}
```

Makrá `\rbK` a `\rbM` spúšťateľné v prostredí `tikzpicture` vyzerajú nasledovne:

```
%.....
\def\rbK at #1 {\begin{scope}[scale=.4]
  \foreach \xi in {1,...,4}
    {\draw[FaI,fill=FaI!15,opacity=.35] (#1)+(45+\xi*90:.26\DX) circle (.23\DX);}
  \draw[black!30,line width=1.pt] (#1) circle (.23\DX) (#1) circle (.5\DX);
  \end{scope}}
%.....
\def\rbM at #1#2#3#4#5#6 {\def\TTx{\bfs#2}\def\TTa{\bfs#3}
  \def\TTb{\bfs#4}\def\TTc{\bfs#5}\def\TTd{\bfs#6}
  \begin{scope}[scale=.4,black]
    \draw (#1) node[scale=1.3] {\FBI\TTx};
    \foreach \xi in {1,...,4} {\draw (#1)+({45+(\xi-1)*90}:.26\DX)
      node[above,scale=.9] {\csname TT\makealph\xi\endcsname};}
  \end{scope}}
```



Obr. 3. Podsudovník OSS Conf 2025

Samotný zdrojový kód prostredia `tikzpicture` je:

```
\begin{tikzpicture}[scale=.9]
\draw[black!10, line width=4pt] (0,0) circle (.5\DX+1pt);
\draw[FaI,line width=2pt] (0,0) circle (.5\DX);
\foreach \x in {1,...,5} {\rbK at {90+\x*72:.7\DX} }
  \rbM at {90+0*72:.7\DX}{\Ta}{\Tb}{\Tc}{\Td}{\Te}
  \rbM at {90+1*72:.7\DX}{\Tc}{\Tb}{\Ta}{\Td}{\Te}
  \rbM at {90+2*72:.7\DX}{\Td}{\Tb}{\Tc}{\Ta}{\Te}
```

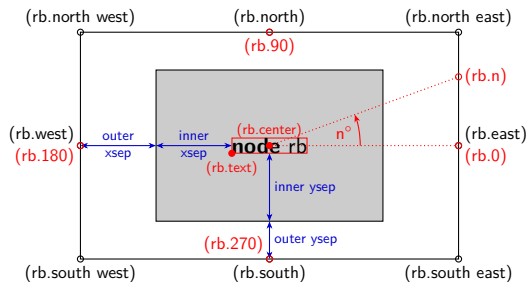
```

\rbM at {90+3*72:.7\DX}{\Te}{\Tb}{\Tc}{\Td}{\Ta}
\rbM at {90+4*72:.7\DX}{\Tb}{\Ta}{\Tc}{\Td}{\Te}
\draw[FaI,line width=2pt] (0,0) circle (.12\DX) node[FaI,scale=1.05]
{\bfs{\begin{tabular}{@{}c@{}}OSS Conf\\ FRI Uniza Žilina\\
1.\,--\,3.\ júl 2025\end{tabular}}};
\draw (90:.46\DX) node[FaI,scale=.95] {\bfs Predchádzajúce ročníky}
(-90:.46\DX) node[FaI,scale=.95] {\bfs Nasledujúce ročníky};
\end{tikzpicture}

```

3. Použitie knižnice positioning

TikZ obsahuje mnoho rôznych knižníc na riešenie rôznych problémov. Jedna z nich je **positioning** pomocou, ktorej môžeme umiestňovať objekty na rôzne pozície v závislosti od predchádzajúcich. Tieto objekty môžeme umiestňovať tak, že zadáme iba smer (svetové strany) alebo môžeme zadať aj vzdialenosť medzi objektami (viď obr. 4). Uhol n sa počíta v kladnom zmysle v stupňoch v rozsahu od 0° po 360° , pričom (rb.0) zodpovedá (rb.360).



Obr. 4. Okolité pozície okolo daného vrcholu **rb**

Pre vytvorenie modelu na obr. 5 potrebujeme načítať knižnice:

```

\usetikzlibrary{arrows.meta}
\usetikzlibrary{positioning}
\usetikzlibrary{shapes.multipart}

```

Základom uvedeného modelu sú rámčeky a poschodové rámčeky (vytvorené pomocou `.style`), ktoré sú prepájané šípkami. Správne dorazenie šípiek na okraje rámčekov zabezpečuje TikZ, našinec iba nastavuje medzery a pozície medzi jednotlivými rámčekami. Aby rámčeky mali rovnakú výšku, niekedy je použitý príkaz `\smash{text}`, ktorý zabezpečí nulovú výšku textu v zátvorkách. Problémy robia dolné presahy písmen `pjqgz` a horné presahy niektorých písmen s diakritikou `íšž` ap. Podobne ako pri podpivníku je potrebné hneď za `\begin{document}` zadať príkaz `\shorthandoff{-}`. Na horizontálne posunutie šípiek je v niektorých prípadoch použitý príkaz na posunutie `xshift`, napr. konštrukcia

```

([xshift=-.5cm]rbL01.north)--([xshift=-.5cm]rbL01 |- rb01.south)

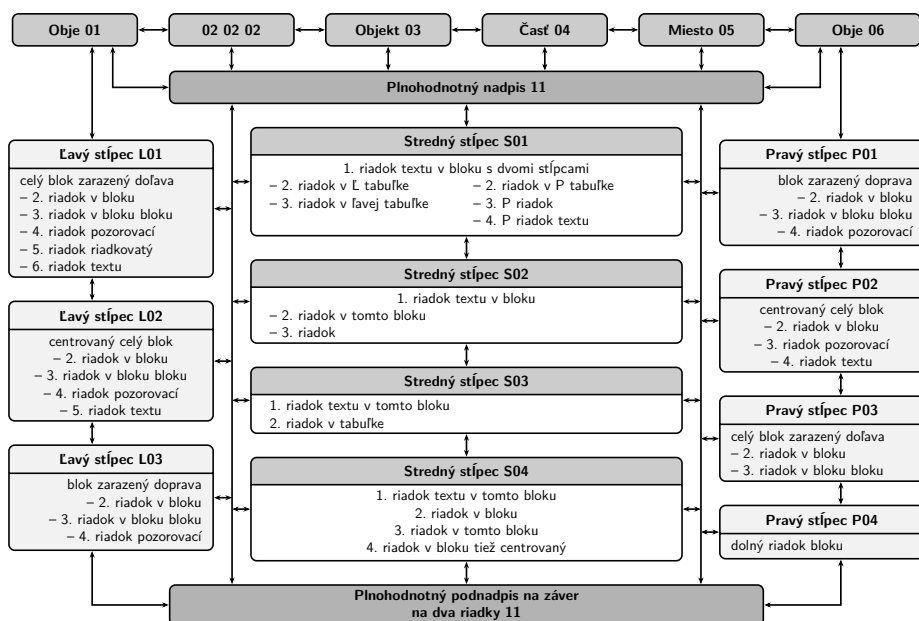
```

zabezpečí, že sa pozícia (`rbL01.north`) posunie o pol centimetra doľava. Druhá časť príkazu (konkrétne `| -`) spôsobí, že sa na horizontálnu úroveň (`rb01.south`) vytvorí kolmica z pozície (`[xshift=-.5cm]rbL01.north`). Opačné poradie

(`[xshift=-.5cm]rbL01.north`)--(`[xshift=-.5cm]rbL01 -| rb01.south`)

by vytvorilo kolmicu na vertikálnu úroveň (`rb01.south`).

Rámčeky sa konštruujú od pozície (0,0) smerom nadol v troch stĺpcoch, preto sú na začiatku definované tri fiktívne body pomocou príkazu `\coordinate`. Samotný text je zapisovaný priamo v rámčekoch do tabuľky `tabular`, ktorá je kvôli väčšej prehľadnosti preddefinovaná v makre `\rbT`.



Obr. 5. Poprepletaný dvojrozmerný model

Nasledujúci zdrojový kód nájde čitateľ na adrese <https://invimath.fri.uniza.sk/images/slides/OSSConf/2025-obr05.tex> ako standalone súbor.



```
\def\rbT[#1]#2{\begin{tabular}[t]{@{}#1@{}}#2\end{tabular}}
%.....
\sfbf\def\bfs{\bf\sffamily}\begin{tikzpicture}[%
  rbto/.style={line width=1.25pt,{Stealth[scale=.6]}--{Stealth[scale=.6]}},
  rbi/.style={rectangle,fill=black!20,minimum width=3.3cm,minimum height=2.2em
    ,very thick,draw=black,rounded corners=.4em},
  rbiI/.style={rectangle,fill=black!30,minimum width=15.7cm,minimum height=2.2em
    ,text width=14.7cm,text centered,very thick,draw=black,rounded corners=.4em},
  rbs/.style={rectangle split,rectangle split parts=2
    ,rectangle split part fill={black!20,white}}
```

```

,draw,minimum width=11.4cm,text width=10.4cm,very thick,rounded corners=.4em},
rbLR/.style={rectangle split,rectangle split parts=2,draw,minimum width=5.4cm
,fill=black!5,very thick,draw,text width=4.8cm,inner sep=4pt,rounded corners=.4em}]

%koordinaty fiktivnych bodov na odrazenie-----
\coordinate (rb00) at (0,0);
\coordinate (rbL0) at (-9.4,0);\coordinate (rbP0) at (9.4,0);

%horny riadok-----
\node[rbI,right] at (.4,0) (rb04) {\bfs \smash{\Časť} 04};
\node[rbI,right =.8 of rb04] (rb05) {\bfs Miesto 05};
\node[rbI,right =.8 of rb05] (rb06) {\bfs \smash{Objekt} 06};

\node[rbI,left] at (-.4,0) (rb03) {\bfs \smash{Objekt} 03};
\node[rbI,left =.8 of rb03] (rb02) {\bfs 02 02 02};
\node[rbI,left =.8 of rb02] (rb01) {\bfs \smash{Objekt} 01};

\foreach \xi/\xj in {1/2,2/3,3/4,4/5,5/6}
{\draw[rbto] (rb0\xi.east)--(rb0\xj.west);}

%nadpis pod hornym riadkom-----
\node[rbII,below=2.8em of rb00] (rb11) {\bfs\smash{Plnohodnotný nadpis} 11};
\draw[rbto] ([xshift=1cm]rb01.south)|-(rb11.west);
\draw[rbto] ([xshift=-1cm]rb06.south)|-(rb11.east);
\foreach \xi in {2,...,5} {\draw[rbto] (rb0\xi.south)--(rb11.north -| rb0\xi);}

%stredny stlpec-----
\node[rbS,below =.6 of rb11] (rbS01) {\bfs\centerline{Stredný stĺpec S01}
\nodepart{second}\centering 1. riadok textu v bloku s dvomi stĺpcami \\
\rbT[1]{-- 2. riadok v ľ tabuľke \\ -- 3. riadok v ľavej tabuľke}\hfill
\rbT[1]{-- 2. riadok v P tabuľke \\ -- 3. P riadok\\
-- 4. P riadok textu} \hfill\phantom. };
\path (rb11) edge[rbto] (rbS01);

\node[rbS,below =.6 of rbS01] (rbS02) {\bfs\centerline{Stredný stĺpec S02}
\nodepart{second}\centerline{1. riadok textu v bloku} \\
-- 2. riadok v tomto bloku \\
-- 3. riadok \\
};
\path (rbS01) edge[rbto] (rbS02);

\node[rbS,below =.6 of rbS02] (rbS03) {\bfs\centerline{Stredný stĺpec S03}
\nodepart{second}1. riadok textu v tomto bloku \\
2. riadok v tabuľke \\
};
\path (rbS02) edge[rbto] (rbS03);

\node[rbS,below =.6 of rbS03] (rbS04) {\bfs\centerline{Stredný stĺpec S04}
\nodepart{second}\centering 1. riadok textu v tomto bloku \\
\centering 2. riadok v bloku \\
\centering 3. riadok v tomto bloku \\
\centering 4. riadok v bloku tiež centrovateľný\par
};
\path (rbS03) edge[rbto] (rbS04);

%lavy-----
\node[rbLR,below =2.9 of rbL0] (rbL01) {\bfs\centerline{Ľavý stĺpec L01}

```

```

\nodepart{second}celý blok zarazený doľava          \\
-- 2. riadok v bloku                                \\
-- 3. riadok v bloku bloku                          \\
-- 4. riadok pozorovací                             \\
-- 5. riadok riadkovatý                             \\
-- 6. riadok textu                                   };
\draw[rbto] ([xshift=-.5cm]rbL01.north)--([xshift=-.5cm]rbL01 |- rb01.south);

\node[rbLR,below =.6 of rbL01] (rbL02) {\bfs\centerline{Ľavý stĺpec L02}
\nodepart{second}\centering  centrovaný celý blok    \\
-- 2. riadok v bloku                                \\
-- 3. riadok v bloku bloku                          \\
-- 4. riadok pozorovací                             \\
-- 5. riadok textu\par                               };
\draw[rbto] ([xshift=-.5cm]rbL02.north)--([xshift=-.5cm]rbL02 |- rbL01.south);

\node[rbLR,below =.6 of rbL02] (rbL03) {\bfs\centerline{Ľavý stĺpec L03}
\nodepart{second}\raggedleft blok zarazený doprava  \\
-- 2. riadok v bloku                                \\
-- 3. riadok v bloku bloku                          \\
-- 4. riadok pozorovací\par                         };
\draw[rbto] ([xshift=-.5cm]rbL03.north)--([xshift=-.5cm]rbL03 |- rbL02.south);

%pravy-----
\node[rbLR,below =2.9 of rbP0] (rbP01) {\bfs\centerline{Pravý stĺpec P01}
\nodepart{second}\raggedleft blok zarazený doprava  \\
-- 2. riadok v bloku                                \\
-- 3. riadok v bloku bloku                          \\
-- 4. riadok pozorovací\par                         };
\draw[rbto] ([xshift=.5cm]rbP01.north)--([xshift=.5cm]rbP01 |- rb06.south);

\node[rbLR,below =.6 of rbP01] (rbP02) {\bfs\centerline{Pravý stĺpec P02}
\nodepart{second}\centering  centrovaný celý blok    \\
-- 2. riadok v bloku                                \\
-- 3. riadok pozorovací                             \\
-- 4. riadok textu\par                               };
\draw[rbto] ([xshift=.5cm]rbP02.north)--([xshift=.5cm]rbP02 |- rbP01.south);

\node[rbLR,below =.6 of rbP02] (rbP03) {\bfs\centerline{Pravý stĺpec P03}
\nodepart{second}celý blok zarazený doľava          \\
-- 2. riadok v bloku                                \\
-- 3. riadok v bloku bloku                          };
\draw[rbto] ([xshift=.5cm]rbP03.north)--([xshift=.5cm]rbP03 |- rbP02.south);

\node[rbLR,below =.6 of rbP03] (rbP04) {\bfs\centerline{Pravý stĺpec P04}
\nodepart{second}dolný riadok bloku                 };
\draw[rbto] ([xshift=.5cm]rbP04.north)--([xshift=.5cm]rbP04 |- rbP03.south);

%dolny podnadpis-----
\node[rbII,below=.6 of rbS04] (rb22)
{\bfs Plnohodnotný podnadpis na záver \\ na dva riadky 11};
\path (rb22) edge[rbto] (rbS04);
\draw[rbto] ([xshift=-6.2cm]rb22.north)--([xshift=-6.2cm]rb11.south);

```

```

\draw[rbto] ([xshift=6.2cm]rb22.north)--([xshift=6.2cm]rb11.south);
\draw[rbto] ([xshift=-.5cm]rbL03.south)|-(rb22.west);
\draw[rbto] ([xshift=.5cm]rbP04.south)|-(rb22.east);

%sipky k zvislym ciaram-----
\draw[rbto] (rbL01.east)--([xshift=-6.2cm]rb22.north |- rbL01.east);
\draw[rbto] (rbL02.east)--([xshift=-6.2cm]rb22.north |- rbL02.east);
\draw[rbto] (rbL03.east)--([xshift=-6.2cm]rb22.north |- rbL03.east);

\draw[rbto] (rbS01.west)--([xshift=-6.2cm]rb22.north |- rbS01.west);
\draw[rbto] (rbS02.west)--([xshift=-6.2cm]rb22.north |- rbS02.west);
\draw[rbto] (rbS03.west)--([xshift=-6.2cm]rb22.north |- rbS03.west);
\draw[rbto] (rbS04.west)--([xshift=-6.2cm]rb22.north |- rbS04.west);

\draw[rbto] (rbS01.east)--([xshift=6.2cm]rb22.north |- rbS01.east);
\draw[rbto] (rbS02.east)--([xshift=6.2cm]rb22.north |- rbS02.east);
\draw[rbto] (rbS03.east)--([xshift=6.2cm]rb22.north |- rbS03.east);
\draw[rbto] (rbS04.east)--([xshift=6.2cm]rb22.north |- rbS04.east);

\draw[rbto] (rbP01.west)--([xshift=6.2cm]rb22.north |- rbP01.west);
\draw[rbto] (rbP02.west)--([xshift=6.2cm]rb22.north |- rbP02.west);
\draw[rbto] (rbP03.west)--([xshift=6.2cm]rb22.north |- rbP03.west);
\draw[rbto] (rbP04.west)--([xshift=6.2cm]rb22.north |- rbP04.west);

\end{tikzpicture}

```

4. Použitie knižnice intersection

Použitie hľadanie priesečníkov kriviek pomocou balíčka ilustruje obr. 6. Jednotlivé krivky sa označia pomocou príkazu `\path` a následne pomocou konštrukcie `\draw[name intersections={of=...}` sa nájdu a vykreslia priesečníky. V prvom prípade sa vykreslí jeden priesečník (`intersection-1`), v druhom prípade sa vykreslia dva priesečníky (`intersection-1`) a (`intersection-2`). Posledný príklad v ukážke nájde všetky priesečníky, ich počet dá do premennej `\t` a následne ich v cykle `\foreach` vykreslí.

```

\begin{tikzpicture}[scale=.6,rbA/.style={red!60!black,fill}]
\clip (-3,-3) rectangle (3,3);
\path[save path=\ppA,name path=ppA] (-3.2,-2) .. controls (-3,2) and (0,-2) .. (3.2,2);
\path[save path=\ppB,name path=ppB] (-3.2,1) .. controls (0,2) .. (3.2,-1);
\path[save path=\ppC,name path=ppC,yscale=1.2] (0,0) circle (2.2);
\path[save path=\ppD,name path=ppD] plot[domain=-3:3,samples=50] (.8*\x*\x-2.75,\x-.5);
\fill[blue!20,opacity=.5] [use path=\ppC];
\fill[red!10,opacity=.5] [use path=\ppD];
\draw[blue!40!black,thick] [use path=\ppA] (2.8,1.9) node[scale=.7] {$a$};
\draw[red!40!black,thick] [use path=\ppB] (-2.7,.9) node[scale=.7] {$b$};
\draw[blue!40!black,thick] [use path=\ppC] (1.6,-2.1) node[scale=.7] {$c$};
\draw[red!40!black,thick] [use path=\ppD] (1.6,-2.5) node[scale=.7] {$d$};

\draw[name intersections={of=ppA and ppB,by={ppAB}}]
(intersection-1) [rbA] (ppAB) circle (.075) node[below,scale=.5] {$ab$};

```

```

\draw[name intersections={of=ppA and ppC,by={ppAC1,ppAC2}}]
  (intersection-1) [black] (ppAC1) circle (.075)
    ++(0,-.1) node[right,scale=.5] {$ac_2$}
  (intersection-2) [black] (ppAC2) circle (.075) node[below right,scale=.5] {$ac_1$};

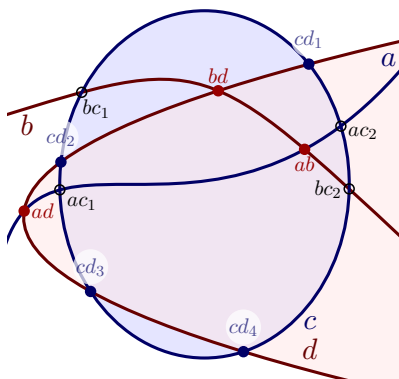
\draw[name intersections={of=ppA and ppD,by={ppAD}}]
  (intersection-1) [rbA] (ppAD) circle (.075) node[right,scale=.5] {$ad$};

\draw[name intersections={of=ppB and ppC,by={ppBC1,ppBC2}}]
  (intersection-1) [black] (ppBC1) circle (.075)
    ++(-.1,0) node[below right,scale=.5] {$bc_1$}
  (intersection-2) [black] (ppBC2) circle (.075) node[left,scale=.5] {$bc_2$};

\draw[name intersections={of=ppB and ppD,by={ppBD}}]
  (intersection-1) [rbA] (ppBD) circle (.075) node[above,scale=.5] {$bd$};

\draw[name intersections={of=ppC and ppD,name=ppCD,total=\t},blue!40!black,fill]
  foreach \i in {1,...,\t} {(ppCD-\i) circle (.075) ++(0,.1)
    node[circle,fill=white,inner sep=.2pt,above,scale=.5,opacity=.7] {$cd_{\i}$}};
\end{tikzpicture}

```

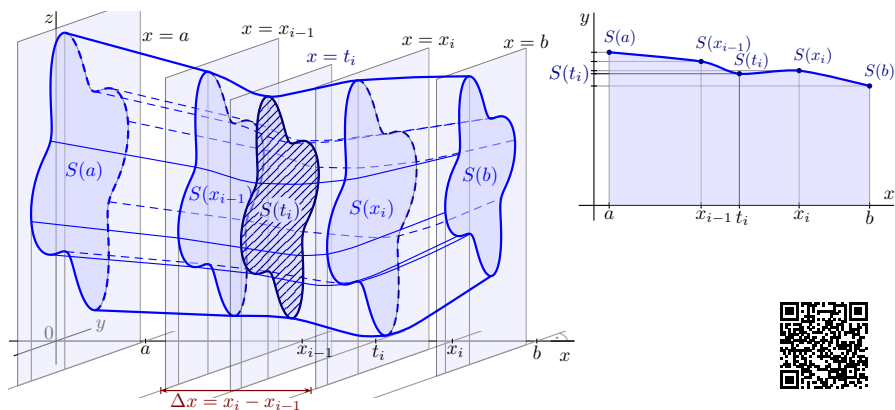


Obr. 6. Použitie knižnice `intersection` na hľadanie priesečníkov kriviek

Na záver ešte ukážem jeden obrázok 7, pri tvorbe ktorého som použil balíček `intesection`. Obrázok ilustruje známe tvrdenie pre výpočet objemu telesa, kde poznáme funkciu $y = S(x) \geq 0$, $x \in \langle a; b \rangle$, ktorá vyjadruje plošný obsah prierezu telesa rovinou kolmou na súradnicovú os x :

$$V_x = \int_a^b S(x) dx.$$

Podakovanie. Príspevok vznikol s príspevom grantu KEGA 019ŽU-4/2023 „Inovatívne učenie matematiky s podporou Open Source“ podporeného Slovenskou kultúrno-edukačnou grantovou agentúrou.



Obr. 7. Objem telesa so známym prierezom kolmým na os x

Literatúra

- [1] Blaško, R., *Neurčitý a určitý integrál a funkcie viacerých premenných*, 2024, online, <https://frcatel.fri.uniza.sk/users/beerb/ma1/sa-2.pdf>.
- [2] Manuál TikZ, <https://www.ctan.org/pkg/pgf>.
- [3] Casteleyn, J. P., *Visual TikZ*, <https://mi-rrors.nic.cz/tex-archive/info/visualtikz/VisualTikZ-fr.pdf>.

Kontaktná adresa

RNDr. Rudolf Blaško, PhD., Katedra matematických metód, Fakulta riadenia a informatiky, Žilinská univerzita, Univerzitná 8215/1, 010 26 Žilina, Slovensko, *Aktuálna adresa*: SOIT, 010 01 Žilina, Slovensko,
E-mailová adresa: beerb@frcatel.fri.uniza.sk, <http://frcatel.fri.uniza.sk/~beerb/>