Ústav automatizace a informatiky
Fakulta strojního inženýrství
Vysoké učení technické v Brně

Přednáška č.2 z předmětu

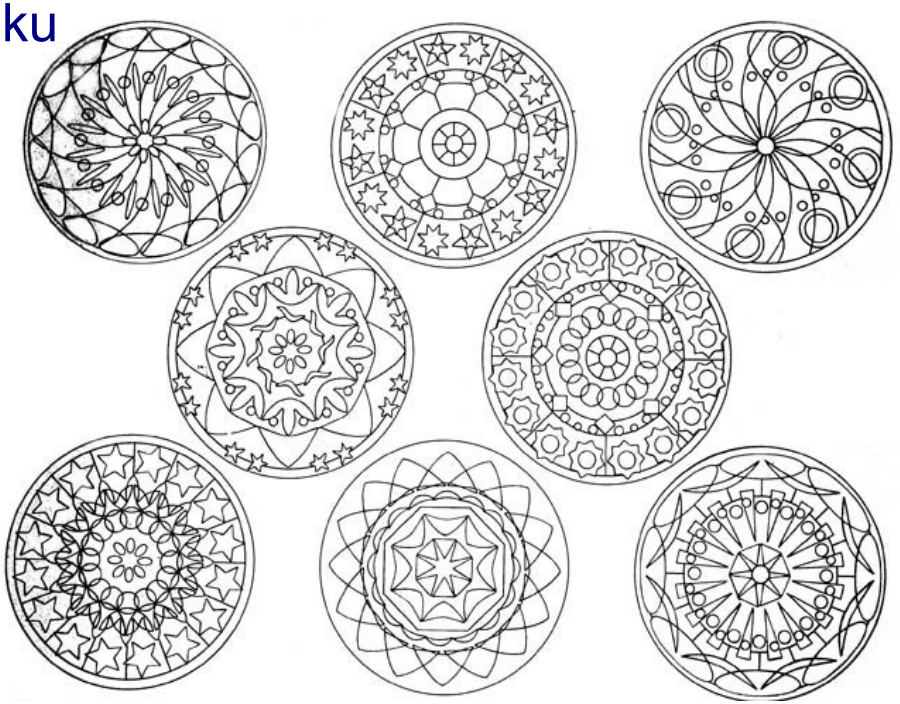
# Počítače a grafika

Ing. Radek Poliščuk, Ph.D.

esf evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

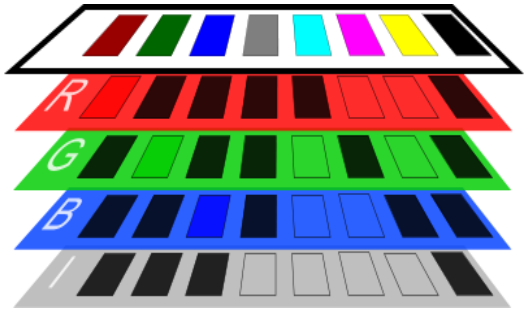
# Obsah přednášky

- **Přednáška 2 – 2D grafika:**
  - základní rastrové operace,
  - základní 2D objekty,
  - bitmapové a vektorové fonty,
  - základní souborové formáty pro bitmapovou a vektorovou grafiku



# Základní rastrové operace

- Dvourozměrné grafické operace se provádějí nad určitou plochou, reprezentující matici zobrazitelných obrazových bodů (plátno, canvas...).
- Tyto body jsou ve framebufferu grafické karty (a nebo v kontextu virtuálního okna GUI) reprezentovány maticí Bytů:
  - Planární uspořádání (EGA, VGA, 16-barevné režimy SVGA):
    - Každý byte obsahoval data pro několik obrazových bodů;
    - Historicky se toto uspořádání používalo jako bitmapa jednotlivých plánů pro monochromatické, 4-barevné, a nakonec i 16-barevné RGBI grafické režimy.
    - Dnes jde spíš o přežitek (nízká rychlost, zbytečně komplikované určování konkrétního obrazového bodu).

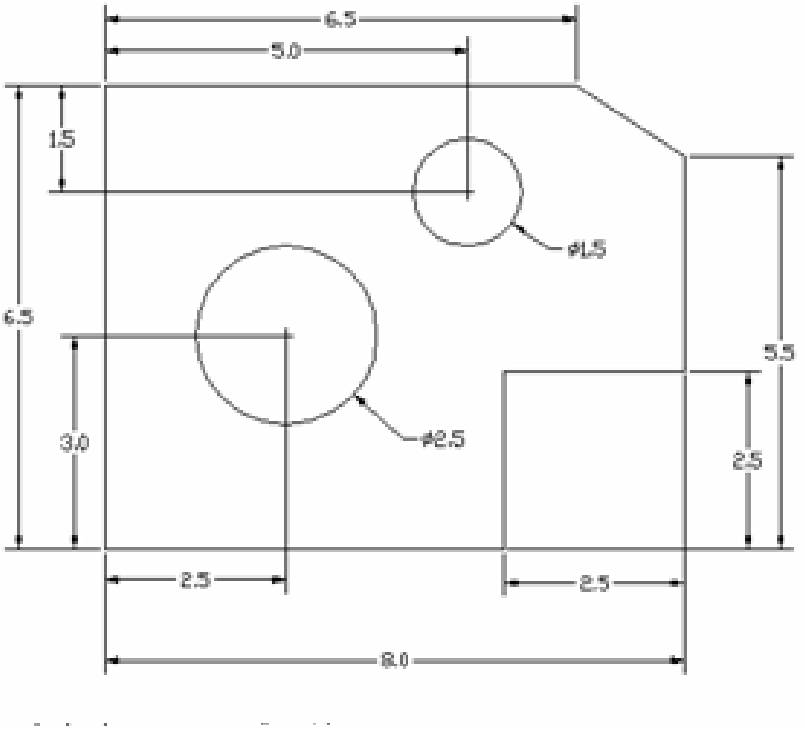



  - Kompozitní uspořádání (MCGA, SVGA v režimu 256+ barev):
    - Každý obrazový bod je definován 1-4 (sousedícími) byty.
    - Adresy jednotlivých bodů jsou jednoznačné, přístup k matici je rychlejší.


- Nad bitmapami je kromě zápisu a čtení možné provádět všechny běžné aritmetické (+-×÷...) i logické operace (AND, OR, XOR,...).

# Základní 2D objekty

- Body
- Obdélníky
- Bitmapy
- Výplně
- Úsečky
- Kružnice, elipsy a oblouky
- Obecné křivky
- Bitmapové a vektorové fonty



# 2D entity - body

- Vykreslení obrazového bodu = zápis příslušných bitů/bajtů na příslušnou adresu obrazové matice (***setpixel***)

$$a(x,y) = ofset + (w' \times y \times Bpp) + (x \times Bpp) ,$$

kde *Bpp* bývá 1 (paleta 256barev), 2 (Hi-Color RGB), 3 a 4 (True-color RGB) Byty/pixel a *w'* označuje šířku obrazové řádky („scanline“) v pixelech.
Z výkonnostních důvodů se šířka scanline zaokrouhluje na celá slova (32bit).

- Inverzní operací k vykreslení bodu je načtení jeho hodnoty (***getpixel***)
- Obě tyto operace jsou dnes standardní součástí GDI knihoven OS, zápis se obvykle provádí ve virtuálním RGB režimu, bez ohledu na skutečně nastavenou bitovou hloubku.

# 2D entity - obdélníky

- Obdélníky (a čtverce) se v normální poloze akcelerovaně vykreslují výplní jednotlivých scanline framebufferu vzorkem odpovídajícím barvě.
- Základní operací je přitom **BitBlt** („Bit Block Transfer“, Xerox ALTO) transformace, dnes podporovaná prakticky všemi grafickými kartami.
  - BitBlt je základem grafických operací založených na vyplňování ploch rastrem, v s daném zápisovém režimu (put, or, and, xor,..).
  - Rozšířením funkcionality vykreslování je režim optimalizovaného kopírování a přesouvání **sprajtů** (obdélníkových oblastí), pomocí kterého jsou dnes prováděny i operace typu rolování v grafickém režimu (obdoba scrollingu v textovém módu).
  - V Delphi:
    - `Canvas.Rectangle`
    - `Canvas.CopyMode`
    - `Canvas.CopyRect`
    - `Canvas.Draw`
    - `Canvas.StretchDraw`

# 2D entity - bitmapy

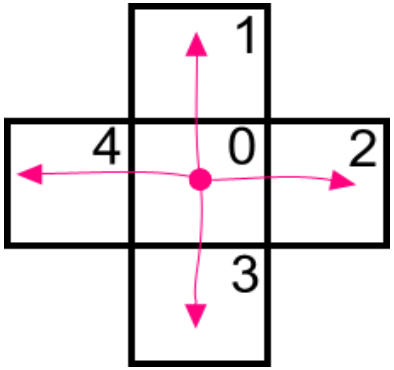
**Bitmapa** (v černobílé planární grafice šlo skutečně o mapu bitů v FB)
= obrazová data sprajtu, se kterými můžeme manipulovat.

- **S hlavičkou** (dnes už jen vyjímečně „Raw Data“ bez hlavičky):
  - RIFF Identifikátor (BM, GIF87a, JFIF,...);
  - Velikost hlavičky ( = offset na data) a vlastních dat;
  - Rozměry obrazové plochy (šířka, výška, příp. délka scanline);
  - Barevný režim (indexovaná barva, odstíny šedi, RGB, CMYK, Lab...);
    - V případě indexované barvy délka a typ palety, v ostatních „přímých“ režimech
    - bitová hloubka (1,2,4,8,16 bitů/kanál) a počet kanálů (1,2,3,4).
  - Typ použité komprese (žádná, RLE, PNG, … více viz přednáška č.6)
  - Rozlišení při tisku (72DPI, 300DPI,...);
- Barevná paleta (u indexovaných barev), obvykle v záznamech RGB ,
- V některých DTP formátech odkaz a nebo přímo vložený barevný profil.
- Vlastní obrazová data (planární nebo kompozitní, s danou kompresí)
  - Soubor/objekt bitmapy může obecně obsahovat obrazová data i více než jednoho obrázku (ICO), nebo i animační instrukce (anim. GIF).

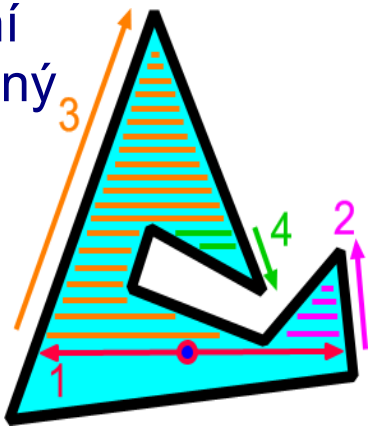
# 2D entity - výplně

Používá se několik algoritmů, s různou paměťovou náročností a rychlostí:

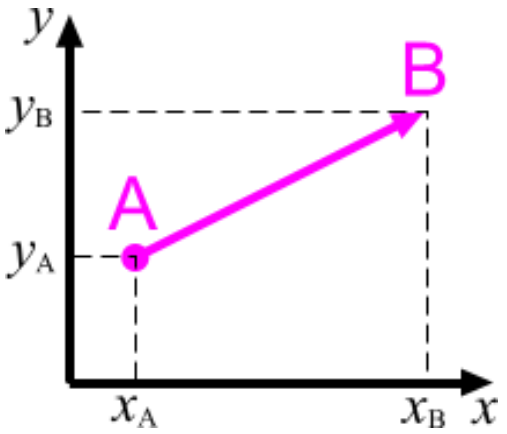
- **Semínkové vyplňování** spojité oblasti (Seed Fill)
  - Předpokládejme že daný bod leží uvnitř platné oblasti,
  - Otestujeme 4 směry šíření a všechny body splňující podmínky spojité výplně obarvíme a přidáme do fronty.
  - Výchozí bod z fronty odstraníme a postup opakujeme, dokud fronta není prázdná (fronta typu FIFO).



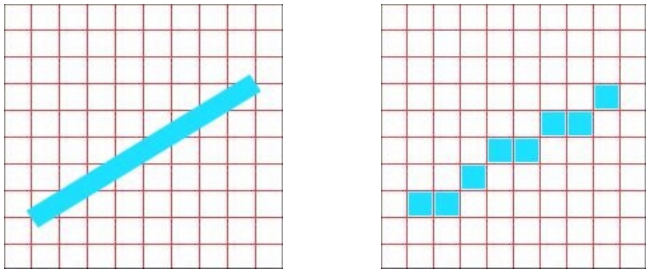

- **Scan-Line algoritmus**: `Canvas.FloodFill`
  - Z výchozího bodu nalezneme a spojíme krajní body platné oblasti ve vodorovném směru („vyplnění scanline“ - dobrá akcelerovatelnost).
  - Nad a pod vyplněnou linií se rekurentně provede vyhledání platných bodů, přičemž každý nalezený platný a neobarvený bod podél původní scanline se stává novým východiskem.
  - Algoritmus končí, jsou-li vyplněny všechny úseky (a k nim rekurentně nalezené oblasti) podél výchozí linie.



# 2D entity - úsečky




- Úsečka je obecně zadaná souřadnicemi počátku (A) a konce (B) v kartézských souřadnicích, resp. souřadnicemi počátku, úhlem a délkou úsečky (polární souřadnice, „želví grafika“).
- Pro vykreslení matematicky zadané úsečky je nutné provést tzv. rasterizaci:

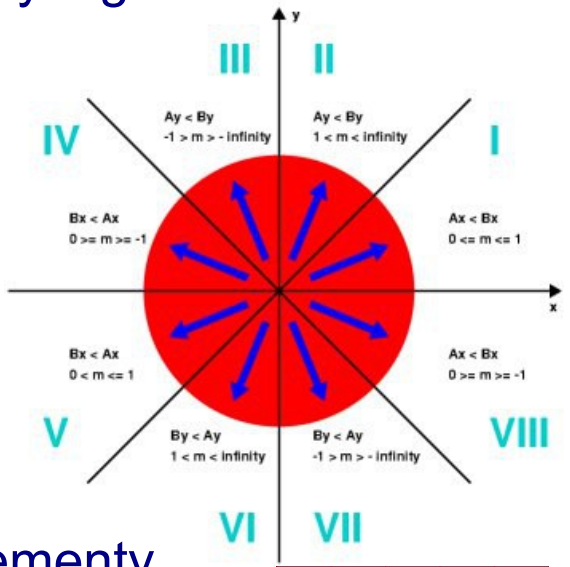


- Klasický „analytický“ vztah $$y(x) = a \cdot x + b = y_A + x \cdot \frac{y_B - y_A}{x_B - x_A}$$

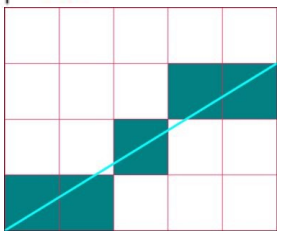
  je pro kreslení „bod po bodu“ prakticky nepoužitelný v okamžiku, kdy potřebujeme souvisle kreslit pod úhlem větším než 45° (nespojitost).
- Parametrická definice $x = f_X(t)$ a $y = f_Y(t)$, neboli $\mathbf{B}(t)=(1-t)\mathbf{P}_A+t\mathbf{P}_B$, $t \in \langle 0,1 \rangle$ je neefektivní, některé neceločíselné výpočty a kreslení se opakují.

# 2D entity - úsečky

Všeobecně užívaným řešením je **Bresenhamův** celočíselný algoritmus (Jack E. Bresenham, 1962):




- Jde o řešení založené na symetrii, kdy před začátkem vlastního kreslení určíme segment, ve kterém bude linie postupovat.
- Směr postupu (vpřed nebo šikmo?) je pak v každém bodě určen znaménkem rozhodovacího kritéria *P*.
- Výchozí hodnota $P_0 := 2d_Y - d_X$, kde
  $d_X = \mathrm{abs}(x_B - x_A)$ a $d_Y = \mathrm{abs}(y_B - y_A)$ jsou celočíselné inkrementy.
  - Pro $P > 0$ => Postup vpřed, $P := P + 2d_Y - 2d_X$;
  - Pro $P \leq 0$ => Postup šikmo, $P := P + 2d_Y$ .



- Veškeré výpočty jsou celočíselné, každý bod se kreslí jen jednou.
- Ostatní segmenty se řeší prohozením souřadnic, znamének, nebo A↔B.

# 2D entity - elipsy

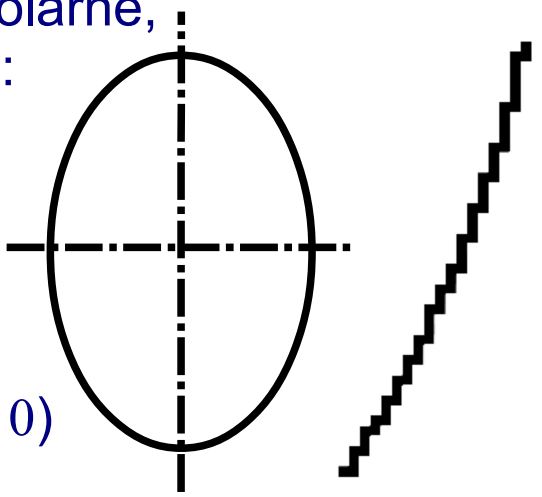
- Rasterizace kružnic a elips (případně jejich oblouků) se provádí
  - pomocí polygonů, složených z „dostatečně krátkých“ úseček. Souřadnice vrcholů tohoto polygonu se počítají polárně, s určitým nastaveným krokem (např. v Autocadu):

$$x(t) = x_C + \cos(t + \varphi) \cdot r_X,$$

$$y(t) = y_C + \sin(t + \varphi) \cdot r_X \text{ ), a nebo}$$

  - s použitím definice elipsy jako kuželosečky:

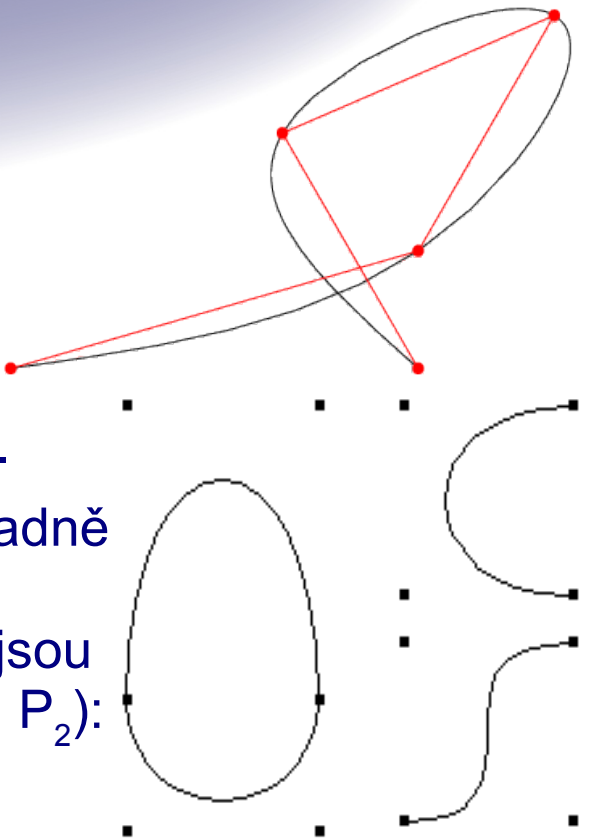
$$y(x) = r_y \sqrt{1 - \left(\frac{x - c_x}{r_x}\right)^2}$$ (normální poloha, $\varphi = 0$)



Řešení v normální poloze se optimalizuje Bresenhamovou metodou založenou na symetrii elipsy v 8 kvadrantech na celočíselném testovacím kritériu (standardní GDI procedury Ellipse a FillEllipse).

  - Popis funkce viz např. http://en.wikipedia.org/wiki/Midpoint_circle_algorithm
  - Složením z dvojice zrcadlově symetrických Bézierových křivek 3.st. (grafické programy – Illustrator, Corel, Xara...)

# 2D entity – křivky



- Pro specifické úlohy (průchod křivky danými body) se používají **kubické splajny** (B-spline, H-spline...), kreslené pomocí polygonů s vrcholy v dané frekvenci.
- Pro ostatní 2D aplikace se dnes používají téměř výhradně tzv. **Bézierovy křivky** (Pierre Étienne Bézier, 1962)**:**
  - **Kvadratická Bézierova křivka** (True-type fonty) jsou definované trojicí bodů (koncové $P_1$, $P_3$, řídící bod $P_2$):

    $$\mathbf{B}(t)=(1-t)^2\mathbf{P}_1+2t(1-t)^2\mathbf{P}_2+t^2\mathbf{P}_3, \text{ kde } t\in\langle 0,1\rangle.$$

  - **Kubická Bezierova křivka** je parametrickou funkcí souřadnic dvou koncových bodů a dvou řídících bodů. Křivka spojuje oba koncové body, řídícími body procházet může, ale nemusí.
    Obecná forma Bézierovy rovnice popisuje souřadnice každého bodu jako funkci časovače (nastavení kroku záleží na konkrétní aplikaci):

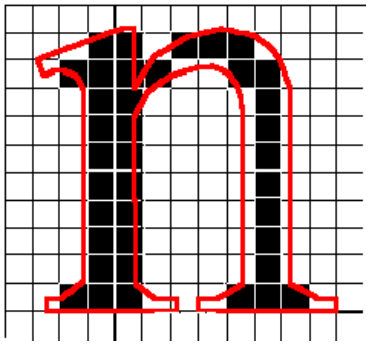
    $$\mathbf{B}(t)=(1-t)^3\mathbf{P}_1+3t(1-t)^2\mathbf{P}_2+3t^2(1-t)\mathbf{P}_3+t^3\mathbf{P}_4, \text{ kde } t\in\langle 0,1\rangle, \text{ kde}$$

    $P_{1,4}$ jsou koncové body, $P_{2,3}$ řídící body (vektorově).
    - V Delphi: Canvas.PolyBezier
    - Detaily viz http://en.wikipedia.org/wiki/Bezier_curve, ukázky viz http://method.ac

# 2D entity – fonty

**Font** je obecným předpisem pro vykreslování znaků na grafickém zařízení.

- **Pevné fonty**: Odlitky znaků nasázené v novinových rotačkách, kladívka v psacích strojích a dálnopisech … *to co nelze měnit softwarem*.
- **Bitmapové fonty** (znakový generátor v grafické kartě, soubory FON):
  - Znaky jsou definovány jako (obvykle monochromatické) sprajty,
  - Definice může popisovat buď jednu a nebo více velikostí znaku,
  - změna rozměrů písma je omezena na základní velikosti, ostatní je nutné odvozovat (zvětšovat, zmenšovat) interpolací (StretchDraw)
  - Vykreslení znaku je prováděno odkazem na příslušný sprajt v matici, indexované příslušnou znakovou sadou (ASCII, Unicode,...)



- **Vektorové fonty** (TrueType, ATM, PostScript...):
  - Řeší omezení změn velikosti u bitmapových fontů,
  - Font je definován jako skupina Bézierových křivek a výplní, odpovídajících příslušným znakům v dané znakové sadě.
  - Rychlost vykreslování je optimalizována dočasnou rasterizací celé znakové sady pro uživatelem zvolenou velikost.
  - Rasterizace s použitím odstínů šedi, RGB barev a průhlednosti je základem různých technik vyhlazování hran (**antialiasing**, **ClearType**).

- Více o vzhledu, funkci a určení fontů viz kapitola 10.

# Základní souborové formáty

- Rastrové
  - RAW: Planární nebo kompozitní data, dekódování řeší konkrétní aplikace.
  - BMP/ICO/CUR: Základní GDI formáty Windows, žádná nebo RLE komprese.
  - PCX: „faxový“ formát, optimalizovaná bezeztrátová RLE komprese, až 24bit.
  - TIFF: „DTP“ formát, různé varianty zápisu bitmap v RIFF formátu (Apple, Amiga..)
  - GIF(89a): až 256 barev, komprese LZW (Lempel-Ziv-Welch,'82), možnost animace
  - PNG: Public-Domain alternativa a rozšíření možností statického GIF (24bit+alpha)
  - JPEG/JFIF: formát pro ztrátový záznam fotografií: DCT, Downsampling, YUV...
  - ...více viz kapitola 6.
- Vektorové
  - WMF/EMF: Nativní Windows GDI stream s předpisem kreslení jednotlivých 2D entit
  - PostScript, EPS, PDF: Obecný programovací jazyk pro vykreslování grafiky...
  - ...více viz kapitola 10.

# Závěr přednášky

Byla probrána následující témata:

- základní rastrové operace (planární a kompozitní uspořádání),
- základní 2D entity (body, obdélníky+bitblt, bitmapy, výplně, čáry, křivky)
- bitmapové a vektorové fonty (organizace, vykreslování)
- základní souborové formáty pro bitmapovou a vektorovou grafiku.

- Úkol pro studenty na dnešní cvičení:

## Kontrola volby témat semestrální práce.

Příští hodina začne 10-15 min. prezentací na téma 2D grafika.