

Rastrová 2D grafika

- Určite kresliace programy používať viete ...
- Ako fungujú algoritmy?
 - Viete nakresliť úsečku z bodu A do bodu B?
 - Viete nakresliť kružnicu?
 - Elipsu?
 - Viete vyplniť farbou uzavretý objekt?
 - Viete nahradiť nejaký rozsah farieb od nejakého bodu s nejakou toleranciou?
 - Ako funguje odstraňovanie červených očí?
 - Viete otáčať časti obrazu?
 - Viete zväčšovať/zmenšovať?
 - Viete napísať písmeno?

Základný algoritmus na nakreslenie čiary [[LDA](#)][[CGR](#)]

```
dx = x2 - x1
dy = y2 - y1
for x from x1 to x2 {
    y = y1 + dy * (x - x1) / dx
    plot(x, y)
}
```

Predpokladom je, že

1) $dx > dy$

2) $x_2 > x_1$

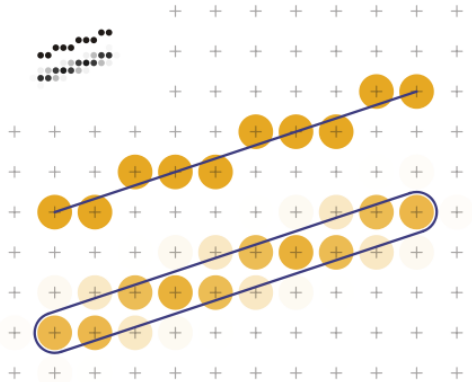
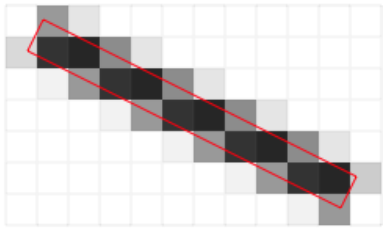
Sú aj rýchlejšie algoritmy, napr. Bresenhamov

Aliasing a anti-aliasing - revízia

- Čo je v rastrovej grafike **aliasing**?
 - pixely/body môžu byť iba na pevných celočíselných pozíciách
 - ak chceme posunúť pixel „medzi“ nedá sa to – body/pixely šikmej čiary robia schodíky
 - tento „jav“ posunu bodov sa volá aliasing, resp „schodiskový efekt“
 - skoky na čiarach sa volajú aj zúbky (jaggies)
- Čo je v rastrovej grafike **anti-aliasing**?
 - zmiernenie viditeľnosti aliasingu
 - napr. v okolí bodu umiestni body s proporcionálne nižšou intenzitou

Príklad antialiasingu

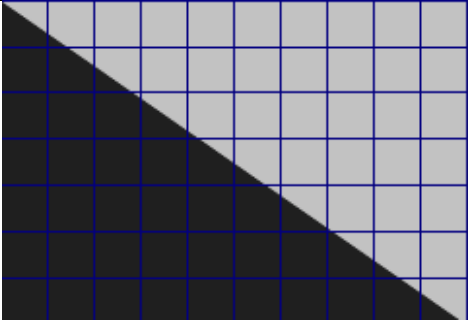
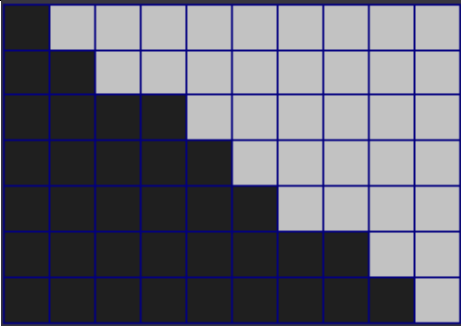
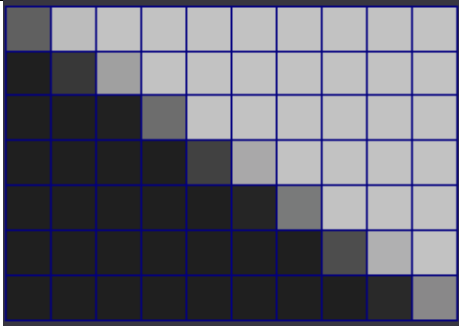
Čiary

Princíp	Príklad čiary s antialiasingom
	

Fonty

	
---	---

Objekty

Ideálna hrana	Raster-bes použitia antialiasingu	S použitím antialiasingu
		

Kreslenie kružnice [CGR]

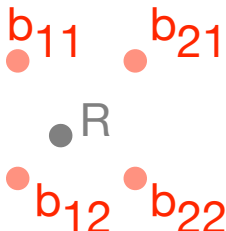
- napr. Bresenhamov algoritmus
- stačí vypočítať 45° , ostatné sa dokreslia pomocou symetrií
- počíta sa bez reálnej aritmetiky, bez odmocnín a mocnín
- keď p = x-ová-súradnica a q =y-ová súradnica, r =polomer

```
1. Initialize  $p = 0$ ,  $q = r$  and  $g = 2(1 - r)$ . Plot  $(p, q)$ .
2. Start loop while  $p \leq q$  for octant, or while  $q > 0$  for quadrant.
  2.1. If  $g < 0$ , then set  $d = 2g + 2q - 1$ , else go to Step 2.2.
    2.1.1. If  $d \leq 0$ , then set  $p' = p + 1$ ,  $q' = q$  and  $g' = g + 2p + 1$ . Go to Step 2.4.
    2.1.2. If  $d > 0$ , go to Step 2.3
  2.2. If  $g > 0$ , then set  $d = 2g - 2p - 1$ , else go to Step 2.3.
    2.2.1. If  $d \leq 0$ , go to Step 2.3.
    2.2.2. If  $d > 0$ , then set  $p' = p$ ,  $q' = q - 1$  and  $g' = g - 2q + 1$ . Go to Step 2.4.
  2.3. Set  $p' = p + 1$ ,  $q' = q - 1$  and  $g' = g + 2p - 2q + 2$ .
  2.4. Plot  $(p', q')$ .
  2.5. Reset  $p = p'$ ,  $q = q'$  and  $g = g'$ .
  2.6. If  $p \leq q$  for octant, or if  $q > 0$  for quadrant, then go to Step 2.1, else go to Step 3.
3. End loop.
```

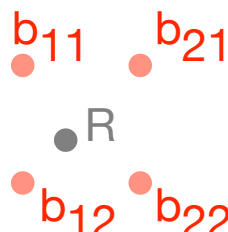
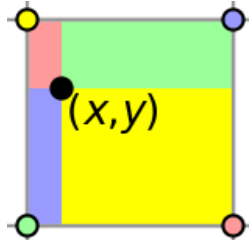
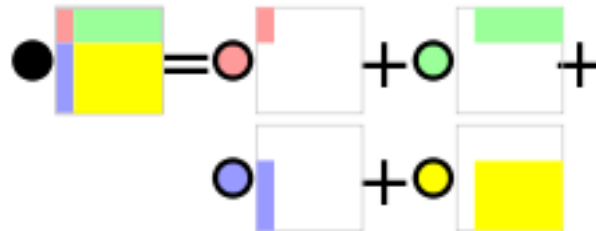
Otáčanie obrazu

- hustota pixelov sa v princípe nemení, len súradnice sa preskupia (spravidla do nie celočíselných pozícií)
- nové hodnoty sa „interpolujú“ z okolitých (stratová, nevratná operácia)
 - v špeciálnych prípadoch 90° , 180° , 270° je otočenie bezstratové
- Základné metódy
 - interpolácia najbližším susedom (nearest neighbor interpolation): hodnota pixelu v destinácii je hodnota vybraného pixelu, ktorý k danému bodu padol najbližšie (resp. naopak, pre každý cieľový pixel sa pýtame ku ktorému zdrojovému pixelu by padol najbližšie)
 - chyba je maximálne pol pixelu
 - chyba sa prejavuje najmä na líniách, ktoré sú natočené šikmo k rastru (budú schodovité)
 - bilinéárna interpolácia
 - využíva 4 okolité hodnoty (2×2), výsledná hodnota je lineárnou kombináciou týchto bodov
 - vplyv jednotlivých bodov je úmerný ich blízkosti k spracovanému bodu
 - bikubická interpolácia – využíva 4×4 okolitých bodov

Princíp interpolácie najbližším susedom

	Keď máme bod R vnútri obĺžnikovej oblasti ohraničenej bodmi $b_{11}...b_{22}$. Hodnota jasu $f(R)$ v bode R je nahradená hodnotou v najbližšom bode b_{ij} V príklade na obrázku je to bod b_{12}, t.j. $f(R) = f(b_{21})$
---	--

Princíp bilineárnej interpolácie

	x,y súradnice bodov: <table><tr><td></td><td>x_1</td><td>x</td><td>x_2</td></tr><tr><td>y_1</td><td>$f(b_{11})$</td><td></td><td>$f(b_{21})$</td></tr><tr><td>y</td><td></td><td>$f(R)$</td><td></td></tr><tr><td>y_2</td><td>$f(b_{12})$</td><td></td><td>$f(b_{22})$</td></tr></table>		x_1	x	x_2	y_1	$f(b_{11})$		$f(b_{21})$	y		$f(R)$		y_2	$f(b_{12})$		$f(b_{22})$	Hodnota jasu v bode R je $f(R) = \frac{1}{(x_2 - x_1)(y_2 - y_1)} [(x_2 - x)(y_2 - y)f(b_{11}) + (x - x_1)(y_2 - y)f(b_{21}) + (x_2 - x)(y - y_1)f(b_{12}) + (x - x_1)(y - y_1)f(b_{22})]$
	x_1	x	x_2															
y_1	$f(b_{11})$		$f(b_{21})$															
y		$f(R)$																
y_2	$f(b_{12})$		$f(b_{22})$															
Mnemotechnicky-princíp váhovania.	Pri nasledovnom farebnom značení: 	Platí: 																

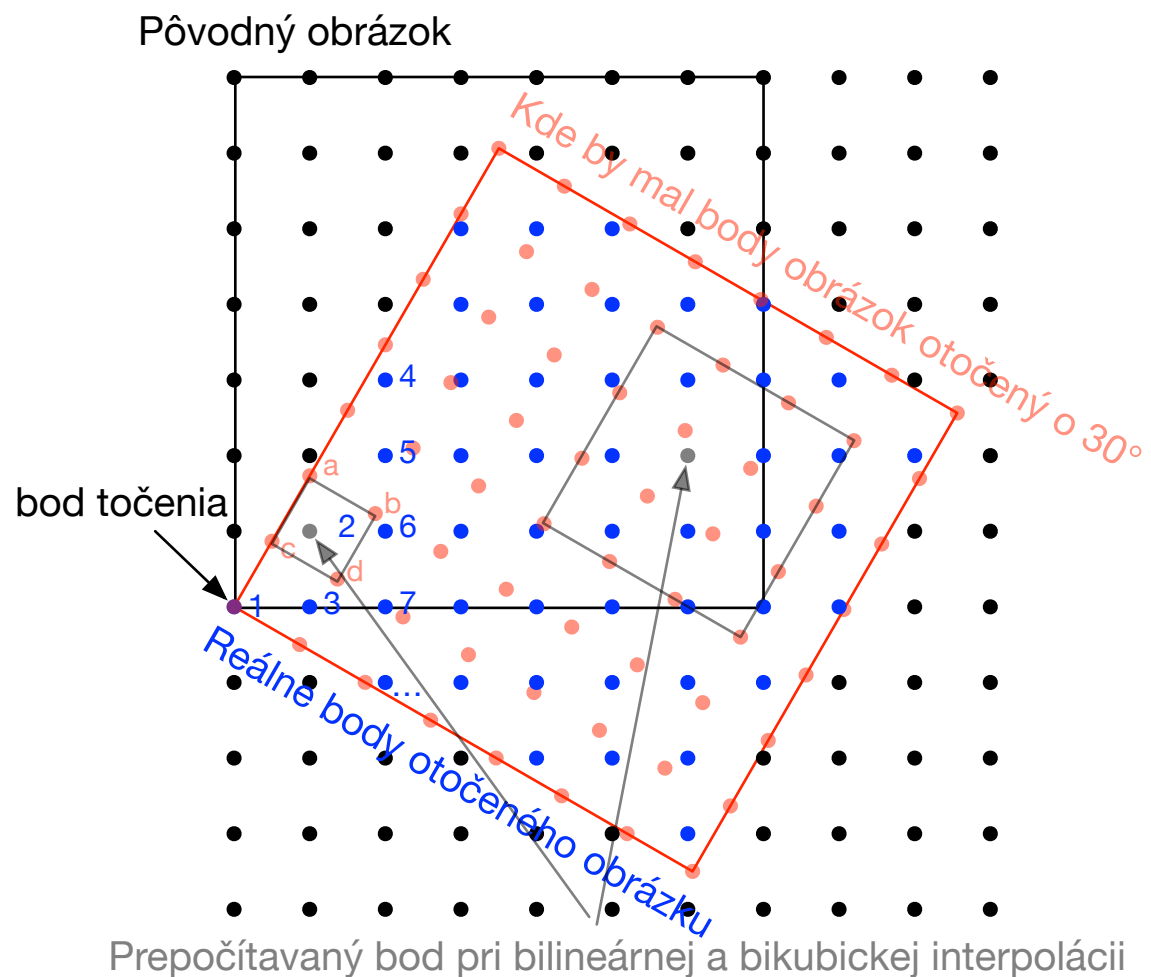
Príklad:

	$x_1 = 0$	$x = 0,25$	$x_2 = 1$
$y_1 = 0$	$f(b_{11}) = 0$		$f(b_{21}) = 200$
$y = 0,75$		$f(R)$	
$y_2 = 1$	$f(b_{12}) = 100$		$f(b_{22}) = 250$

$$f(R) = \frac{1}{(x_2 - x_1)(y_2 - y_1)} [(x_2 - x)(y_2 - y)f(b_{11}) + (x - x_1)(y_2 - y)f(b_{21}) + (x_2 - x)(y - y_1)f(b_{12}) + (x - x_1)(y - y_1)f(b_{22})]$$

$$f(R) = \frac{1}{1} [(0,75)(0,25)0 + (0,25)(0,25)200 + (0,75)(0,75)100 + (0,25)(0,75)250] = 115,625$$

Otočenie obrazu – aplikácia bilineárnej a bikubickej interpolácie



Príklad – výsledná degradácia obrazu po rotácii $8 \times 45^\circ$

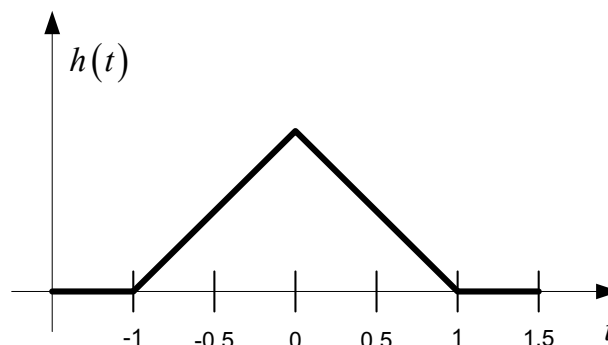
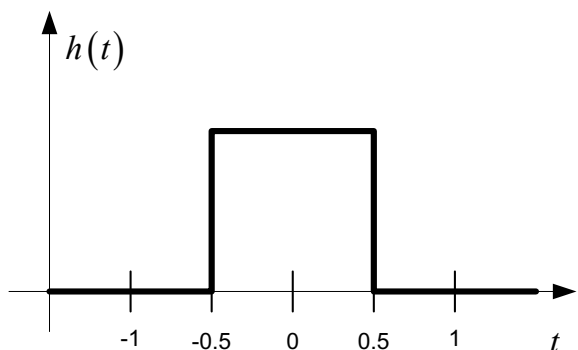


```
clear all;  
close all;  
I = imread('wpeppers.jpg');  
imshow(I)  
ang=45  
for idx=1:360/ang  
    I=imrotate(I,ang,'bicubic');    %'nearest'/'bilinear'/'bicubic'  
    figure; imshow(I)  
    idx*ang  
end
```

Konvolučná metóda interpolácie

Spomeňme na ADSS1 (posledná prednáška): $x(t) = h(t) * \sum_n x_d(n) \delta(t - nT_{vz})$

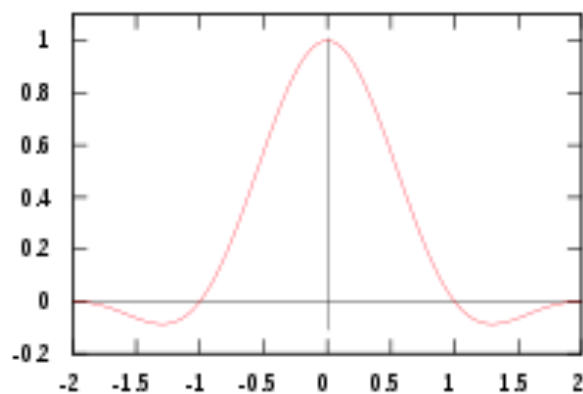
kde, $h(t)$ je konvolučné jadro (kernel) a môže byť napr (konštantná, lineárna interpolácia):



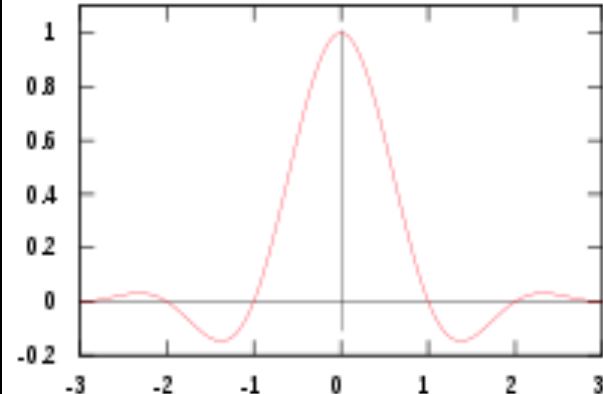
, alebo si funkcia $h(t) = \text{si}(\frac{t\pi}{T_{vz}})$

,alebo napr. Lanczosova funkcia:

$$h(t) = \begin{cases} \text{si}(t)\text{si}\left(\frac{t}{a}\right), & \text{ak } -a < t < a \\ 0, & \text{ináč} \end{cases}$$



$h(t)$ ak $a=2$

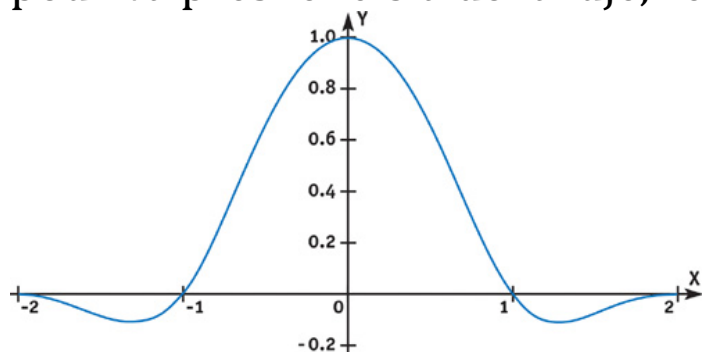


$h(t)$ ak $a=3$

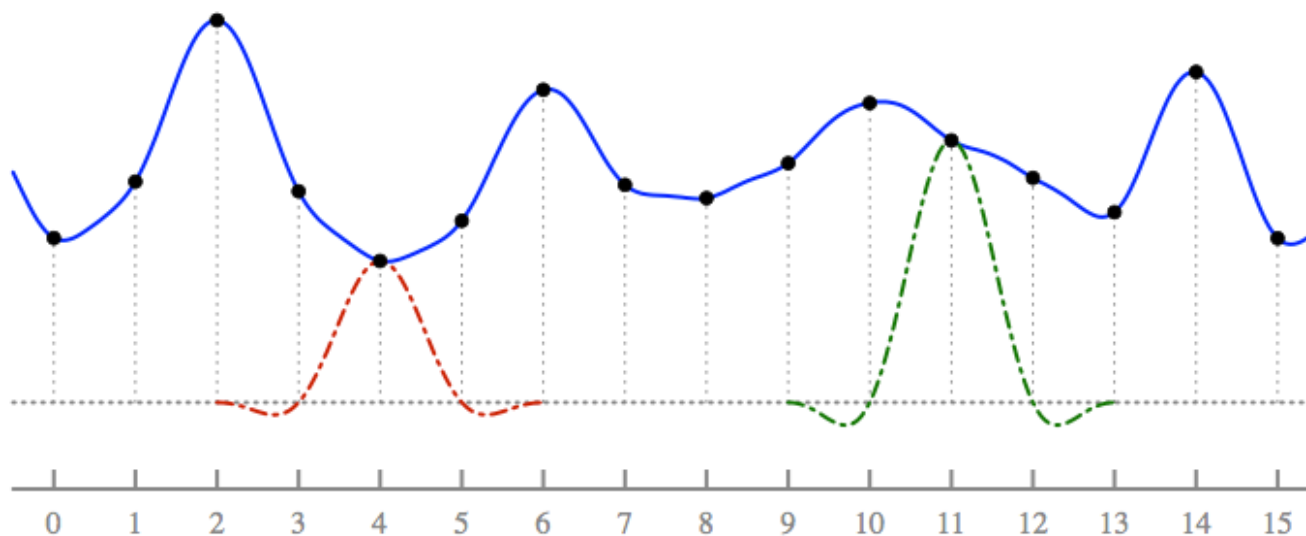
,alebo napr. Bikubické jadro (t.j. kubický spline[PARK93], pozor, nie B-spline, ten vyzerá trochu ináč):

$$h(t) = \begin{cases} (a+2)|t|^3 - (a+3)|t|^2 + 1 & \text{pre } |t| \leq 1 \\ a|t|^3 - 5a|t|^2 + 8a|t| - 4a & \text{pre } 1 < |t| < 2 \\ 0 & \text{ináč} \end{cases}$$

pričom a je obvykle nastavené na -0.5 až -0.75 (v [R.G.Keys, 1980: Cubic Convolution Interpolation] používa presne -0.5 a dokazuje, že táto hodnota je optimálna).



Výsledkom konvolúcie je spojitý signál (napr s použitím Lanczosovej funkcie s $a=2$) – z čiernych bodiek vyrobí modrú funkciu:



Keď prejdeme zo vzťahu:

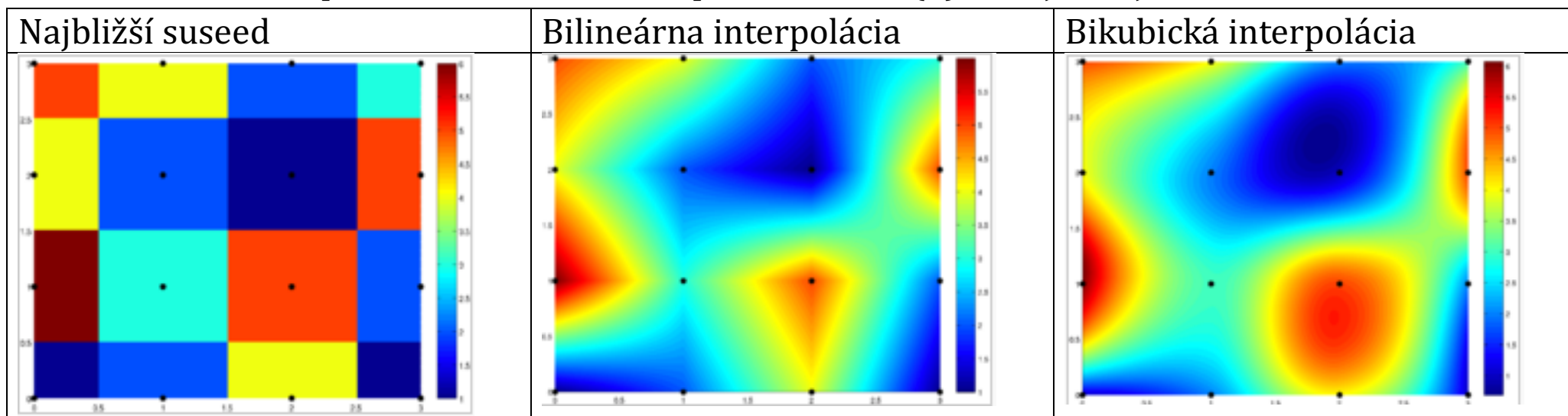
$$x(t) = h(t) * \sum_n x_d(n) \delta(t - nT_{vz})$$

do dvoch rozmerov, potom:

- náš obrázok je $obr_{disk}(x, y) = \sum_{n1} \sum_{n2} x_d(n1, n2) \delta(x - n1, y - n2)$, pričom $n1, n2 \in Z$
- spojitá plachta je $obr_{spoj}(x, y) = h(x, y) * obr_{disk}(x, y)$ (1)

Platí:

- Najbližší sused = cca 2D konvoúcia podľa vzťahu (1), keď jadro je Konštantné
- Bilineárna interpolácia = 2D konvolúcia podľa vzťahu (1), keď jadro je Lineárne
- Bikubická interpoácia = 2D konvolúcia podľa vzťahu (1), keď jadro je Bikubické



- V praxi nemusíme vyrábať komplet celú plachtu, stačí nám urobiť výpočet iba pre špecifické body s daným x, y ľubovoľne vybrané z „plachty“.

Matlab a zväčšovanie/zmenšovanie obrazu

- Príkaz „imresize“
- Využíva interpoláciu a DP filtrovanie
- Pri zväčšovaní rozmerov
 - Použitá len interpolácia
- Pri zmenšovaní rozmerov
 - najprv aplikovaný DP filter, následne interpolácia
- Matlab implementuje
 - Metódu najbližšieho suseda, bilineárnu interpoláciu, bikubickú interpoláciu
 - Interpolácie použitím konvolučných jadier:
 - 'box' - konštantná interpolácia
 - 'triangle' - lineárna interpolácia (bilinear)
 - 'cubic' - kubická interpolácia – odpovedá bicubic interpolácii
 - 'lanczos2' – Lanczosova interpolácia pri a=2
 - 'lanczos2' – Lanczosova interpolácia pri a=3
- Na ďalšom slide sú príklady degradácie kvality pri zmenšení a spätnom zväčšení

```
I2 = imresize(I,1/16, konvolucne_jadro);  
I = imresize(I2,16, konvolucne_jadro);
```

Originál	Najbližší sused	Bilinear
		
Bicubic	Lanczos2	Lanczos3
		

Veľké obrázky

- Kuala Lumpur 846 gigapixels (2014)
- Moon surface 681 gigapixels (2010-2013)
- ...
- Londýnska panoráma 360° (r. 2012, 320 Gigapixelov) (<http://360gigapixels.com/london-320-gigapixel-panorama/>)

360 °obraz

Zdroj

- Z 3D modelu (nachádzame sa uprostred scény)
- Z 360 ° kamery/fotoaparátu (Ricoh Theta (2obj), Samsung Gear 360(2obj), Nokia OZO (má Wavelet-based raw kompresiu!) (veľa, veľa objektívov aj veľká cena)
- Panorámy vyskladané z mnohých obrázkov zošívaním

Cieľ

- Pozeráme na 2D signál monitor, ak tak chceme dostať 360°, musíme spraviť projekciu
 - **Projekcia:** spôsob/dohovor, ako je 3D objekt zobrazený na 2D povrchu
 - Používajú sa 2 hlavné projekcie:

Equirectangular (360°x180°) - Cylindrická panoráma (ekvidištančná valcová projekcia) – rovnobežky na zemeguli by boli horizontálne úsečky z ľava do prava	Cubic (kubická)
---	-----------------

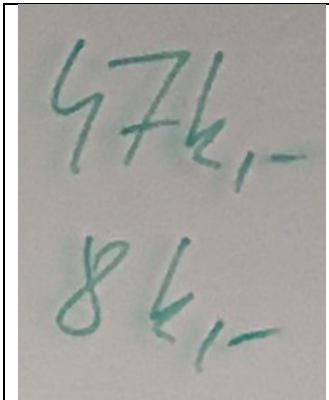
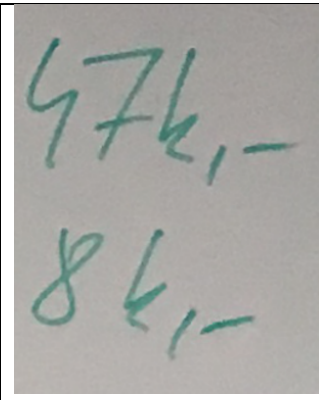


- 360° stereoskopický obraz obraz

Zošívanie obrazu

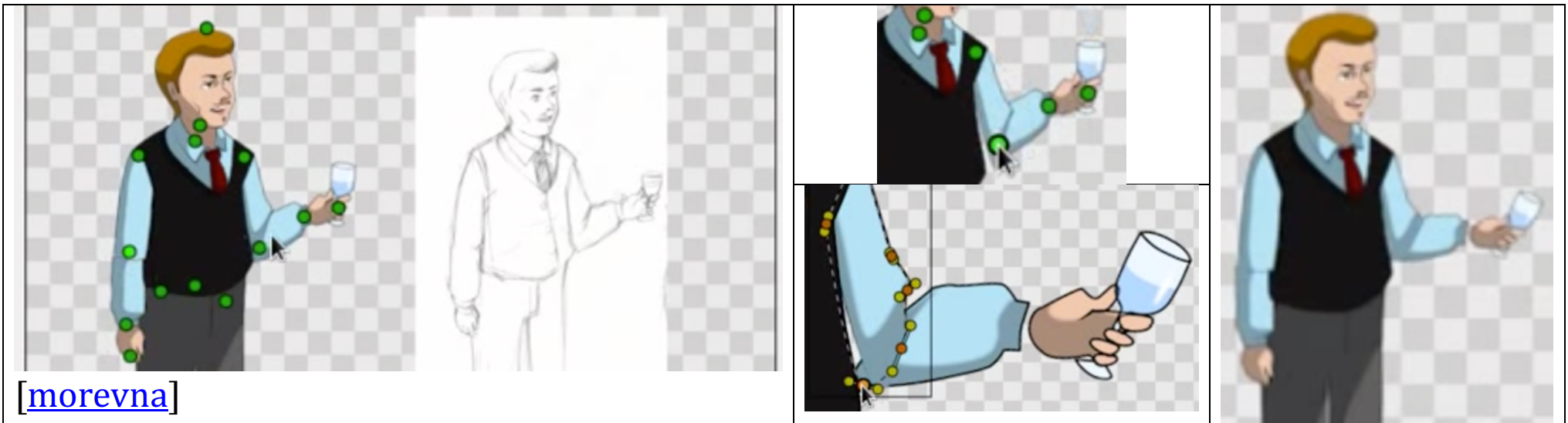
- chceme vytvoriť panorámu ... ako na to? Cieľom je zošívať
 - do lineárnej panorámy – vytvárané posuvným pohybom, posúvaním pozdĺž (pohyb sa volá "tracking", "travelling", resp. „laterar“, napr chcem obraz whiteboardu s pokresleným obsahom
 - niekedy volá aj „Route panorama“ Principiálne sa hovorí o použití „line camera“ (t.j. normálna kamera) alebo „slit camera“ (štrbinová kamera)
 - Štrbinové snímanie veľmi súvisí so skenovacou fotografiou, čo je fotografovanie pomocou klasických scannerov a nie fotoaparátov. Tento prístup má veľké špecifiká, najmä čo sa týka hustoty bodov a hĺbky ostrosti.

- do „panning“ panorámy (očakáva sa rotačný (panning) pohyb [PANNING]. Tento pohyb sa nazýva aj rotácia, alebo „yaw“)
- Zošívanie – programy: Hugin, Affinity Photo, Autostitch, ...
 - rôzne algoritmy zošívania pre lineárnu/panning panorámu
 - frekvenčné algoritmy za sebou nechávajú niekedy „závoj“, keď sa zošíva text ...

				Príklady zošitých obrázkov (Vľavo Autostitch – pridal závoj, pvravo Affinity Photo)
--	--	--	--	---

Animácia

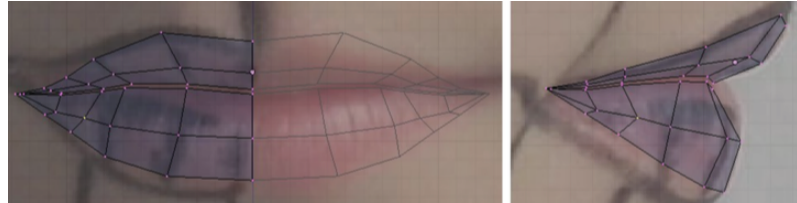
- Pri animácií sa vyrába sekvencia obrázkov, a pridáva sa aj audio = film
- Metodika výroby obrazovej časti – podľa typu animácie
 - Animovaná kresba 2D - všetko sa pracne kreslí znovu a znovu nanajvýš sa editujú rastrové objekty
 - Animovaná 2D vektorová grafika (odopručané Synfig Studio (Open source) [[Demo](#)])
 - Najprv sa vyrobí náčrtky (bitmapové kresby)
 - Potom sa obrázky vektorizujú (prekresľovanie kresby do vektorovej formy)
 - Animuje sa vektorová forma – objekty sa otáčajú, posúvajú, mení sa im tvar atď' ...
 - Pri postavách sa môže vytvoriť spojený mechanický model (viď zelené body)



Animácia 3D

Animované 3D (Odporúčaný Blender)

- výroba náčrtkov / fotiek(!) (narys, podorys bokorys ...) ako v 2D

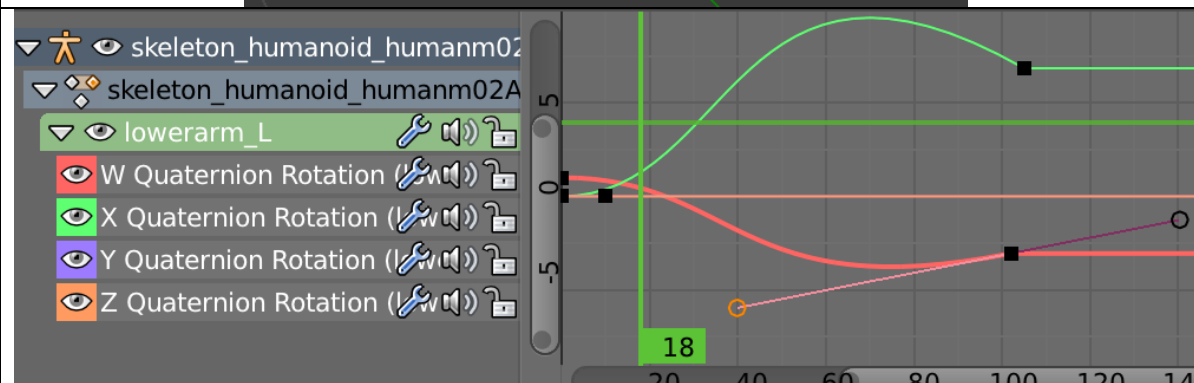
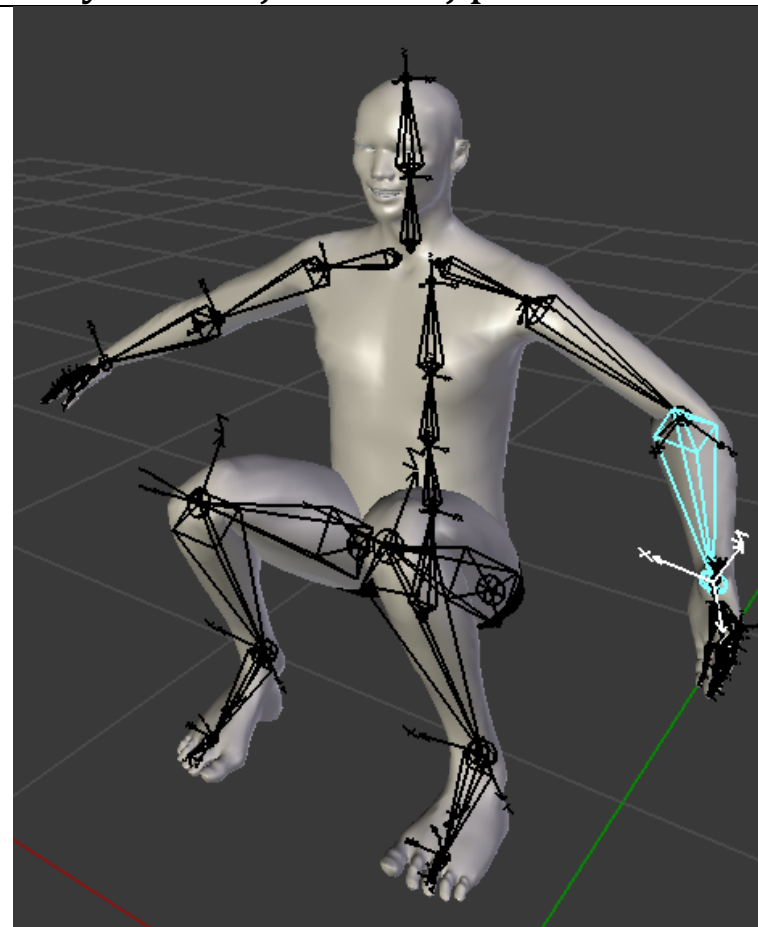


- výroba 3D modelov (tu pri postavách napr. pomôže program „Makehuman“, Add-onsy do Blenderu, ...)
 - objekty sa posúvajú po scéne, modeluje sa ich pohyb, hýbe sa kamerou, osvetlením atď' ...
 - veľa sa aj skriptuje pohyb (automaticky sa opakujúce pohyby, ...)
- Často používané metódy
 - keyframing – definuje medzné pozície, medzi ktorými počítač vytvorí plynulý prechod
 - pohyb človeka – animácia kostry (3 modelu postavy)
 - inverzná kinematika – nemusí sa určovať poloha každého kĺbu a otočena zvlášť, ale definuje sa poloha niektorých kľúčových bodov, polohy všetkých kĺbov sú následne dopočítané algoritmicky
 - motion capture – pre realistický pohyb je zachytený pohyb živého herca , následne je pohyb aplikovaný na 3D model postavy

Základná poloha



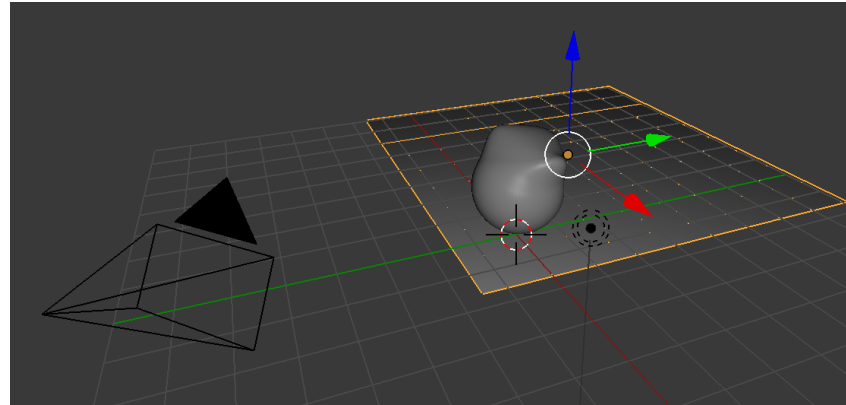
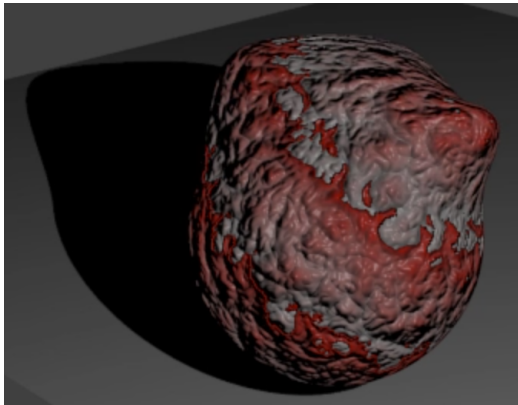
Kĺby v ďalšej medznej pozícii



Príklad priebehu polôh kĺbu medzi nastaviteľný pomocou beziérových kriviek

Tvorba animácií v Blenderi

- nastavenie výstupných parametrov (formát, framerate, ...)
- spustenie renderovania v príklade ... cca 2 frame/s ...
- môže sa použiť aj video editovanie nástroje



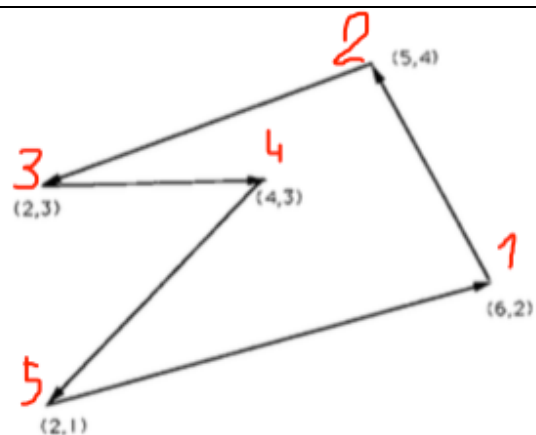
- Blender má implementovaný fyzikálny engine
 - objekty môžu byť pod vplyvom gravitácie
 - simulácia kolízií - ok sa objekty zrazia ..
 - časticový engine (particle engine) – ovládanie vlastností obrovského množstva častíc ...
 - vlasy a ich správanie ...
 - a mnoho ďalšieho

Vektorová grafika

- Súvisí s lineárnou algebrou – je o krivkách ☺
- Umožňuje precízne modelovanie objektov
 - Umožňuje prácu s údajmi a objektami
- Napr. Kde/či sa objekty dotýkajú, prekrývajú, ako sú ďaleko od seba
 - tvorba prienikov, súčtov objektov
- Napr. aký majú objekty obsah ...
 - Majme uzavretý útvar s bodmi $P_i = (x_i, y_i)$, kde $i = 1, \dots, n$ očíslovanými v protismere hodinových ručičiek.
 - Potom obsah útvaru je:

$$\blacksquare A = \frac{1}{2} \begin{vmatrix} x_1 & x_2 & \dots & x_{n-1} & x_n & x_1 \\ y_1 & y_2 & \dots & y_{n-1} & y_n & y_1 \end{vmatrix}$$

- Príklad:

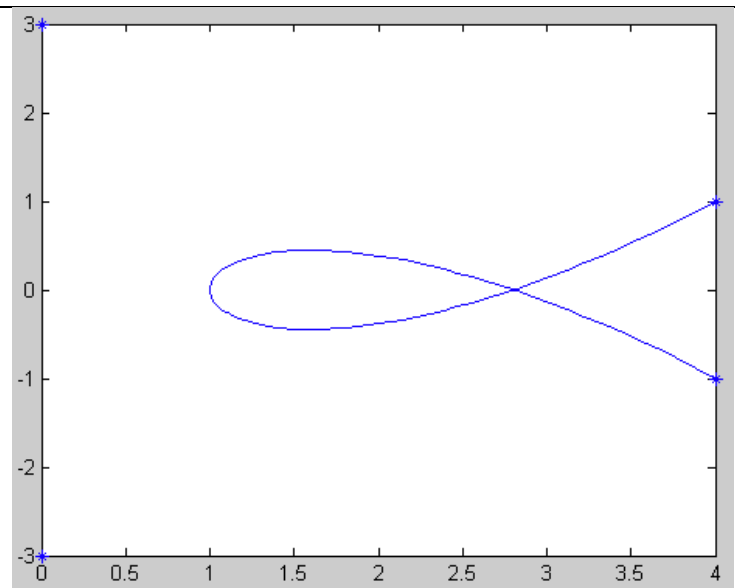


$$A = \frac{1}{2} \begin{vmatrix} 6 & 5 & 2 & 4 & 2 & 6 \\ 2 & 4 & 3 & 3 & 1 & 2 \end{vmatrix} = 1/2 [(6 \times 4 + 5 \times 3 + 2 \times 3 + 4 \times 1 + 2 \times 2) - (5 \times 2 + 2 \times 4 + 4 \times 3 + 2 \times 3 + 6 \times 1)] = 7.5$$

Krivky

- Aký je rozdiel medzi funkciou a parametrickou krivkou?
- Kam patria kružnice, elipsy ?
- Parametrické krivky
 - $P: R \rightarrow R^2$
 - $P(t) = (x(t), y(t))$
- Príklad zápisov:
 - $P(t) = (\cos t, \sin t)$
 - $P(t) = (1 - t)P_0 + tP_1$, Kde je krivka v čase $t=0$? Kde je v čase $t=1$?
 - V nižšie uvedenom príklade je kde krivka v uvedených časoch?

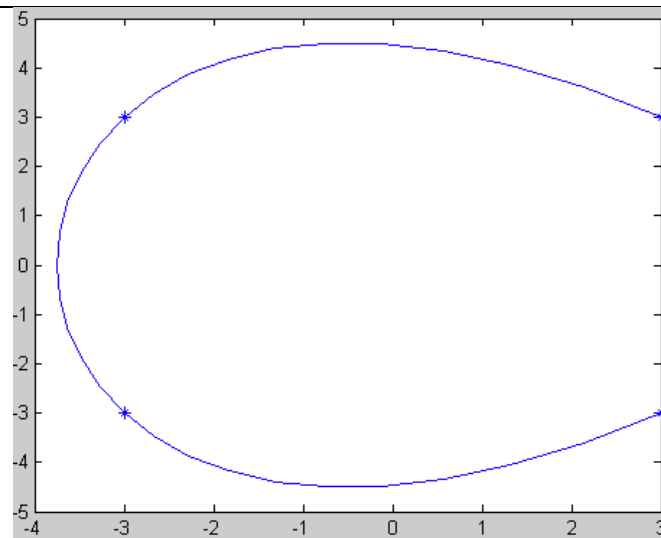
```
clear all;  
close all;  
t=0:0.01:1  
myx=4*(1-t).^3+0*3*(1-t).^2.*t+0*3*(1-t).*t.^2+4*t.^3;  
myy=1*(1-t).^3-3*3*(1-t).^2.*t+3*3*(1-t).*t.^2-1*t.^3;  
plot(myx, myy);  
hold on;  
plot([4, 0, 0, 4],[1, -3, 3, -1], 'x')
```



Interpolácia funkcií a kriviek

- polynommická interpolácia bodov $P_0, \dots, P_n$ zosnímaných v časoch $t_0, \dots, t_n$?
 - Cieľom je nájsť takú parametrickú krivku $P(t)$, že $P(t_k) = P_k$, pre každé k
 - napr. lineárna interpolácia
- ako prejsť od interpolácie funkcie k interpolácii krivky?
 - Robiť interpoláciu zvlášť pre obe súradnice ...
- Napr: interpolujme polynommickou interpoláciou body: $P_k = \{(3,3), (-3,3), (-3,-3), (3,-3)\}$

```
clear all;
close all;
t=0:0.1:3
myx=3*(t-1).*(t-2).*(t-3)/(-6)-3*(t-0).*(t-2).*(t-3)/(2)-3*(t-0).*(t-1).*(t-3)/(-2)+3*(t-0).*(t-1).*(t-2)/(6);
myy=3*(t-1).*(t-2).*(t-3)/(-6)+3*(t-0).*(t-2).*(t-3)/(2)-3*(t-0).*(t-1).*(t-3)/(-2)-3*(t-0).*(t-1).*(t-2)/(6);
plot(myx, myy);
hold on;
plot([3, -3, -3, 3],[3, 3, -3, -3], 'x')
```



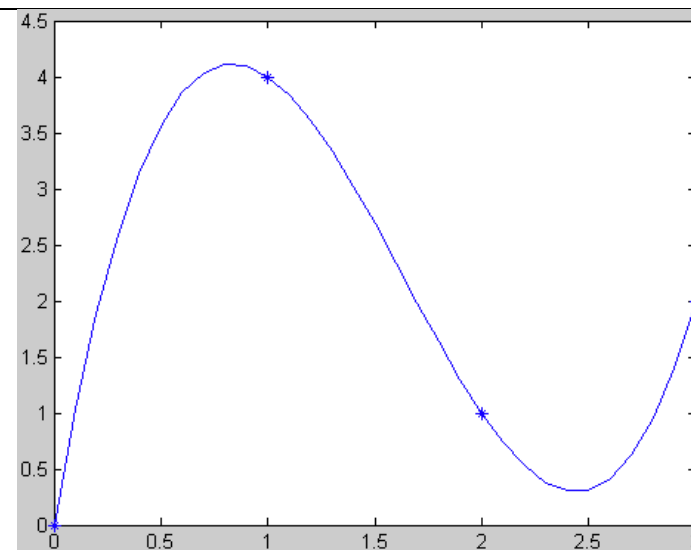
Hm ... čo že to je v tom programe ? Predsa Lagrangeova metóda na interpoláciu pomocou polynómov.

Lagrangeova interpolácia funkcií

- Majme $k + 1$ dátových bodov x_j s hodnotami y_j
- Potom sa dajú zostrojiť polynómy $l_j(x)$ nanajvýš k -teho stupňa, ktoré
 - majú hodnotu 1 vo vybranom dátovom bode z uvedených
 - vo všetkých ostatných dátových bodoch majú hodnotu 0
- Stačí ich naváhať príslušnými hodnotami y_j a výsledok je:
- $L(x) = \sum_{j=0}^k y_j l_j(x)$
- pričom $l_j = \prod_{0 \leq m \leq k, m \neq j} \frac{x - x_m}{x_j - x_m}$
- t.j. polynómy, ktoré pre $x_j = (0, 1, 2, 3)$ majú hodnotu 1 pri j a pre iné majú hodnoty 0:
 - $j = 0: l_0(x) = \frac{(x-1)(x-2)(x-3)}{(0-1)(0-2)(0-3)} = -\frac{1}{6} (x-1)(x-2)(x-3)$
 - $j = 1: l_1(x) = \frac{(x-0)(x-2)(x-3)}{(1-0)(1-2)(1-3)} = +\frac{1}{2} (x-0)(x-2)(x-3)$
 - $j = 2: l_2(x) = \frac{(x-0)(x-1)(x-3)}{(2-0)(2-1)(2-3)} = -\frac{1}{2} (x-0)(x-1)(x-3)$
 - $j = 3: l_3(x) = \frac{(x-0)(x-1)(x-2)}{(3-0)(3-1)(3-2)} = -\frac{1}{6} (x-0)(x-1)(x-2)$

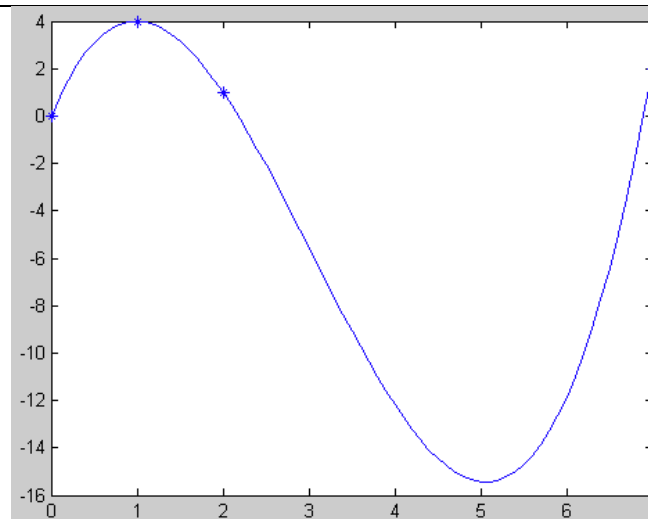
Príklad: Ekvidištančné vzdialenosti bodov

```
clear all;  
close all;  
x=0:0.1:3  
myy=0*(x-1).*(x-2).*(x-3)/(-6)+4*(x-0).*(x-  
2).*(x-3)/(2)+1*(x-0).*(x-1).*(x-3)/(-  
2)+2*(x-0).*(x-1).*(x-2)/(6);  
plot(x, myy);  
hold on;  
plot([0, 1, 2, 3],[0, 4, 1, 2], 'x')
```



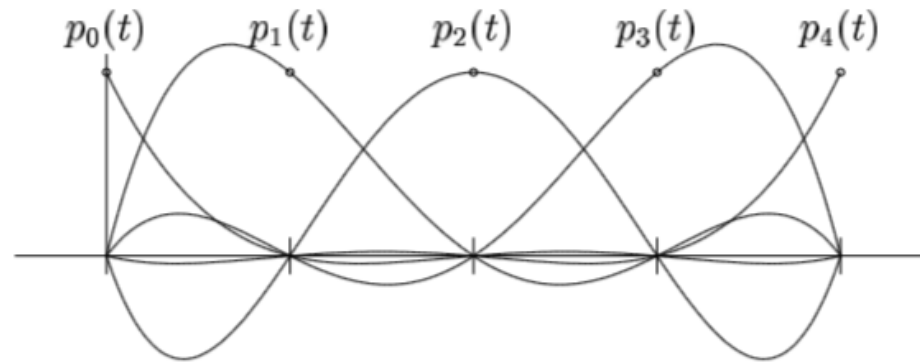
Príklad: Lubovoľné vzdialenosti bodov

```
clear all;  
close all;  
x=0:0.1:7  
myy=0*(x-1).*(x-2).*(x-7)/(-14)+4*(x-0).*(x-  
2).*(x-7)/(6)+1*(x-0).*(x-1).*(x-7)/(-  
10)+2*(x-0).*(x-1).*(x-2)/(7*6*5);  
plot(x, myy);  
hold on;  
plot([0, 1, 2, 7],[0, 4, 1, 2], 'x')
```



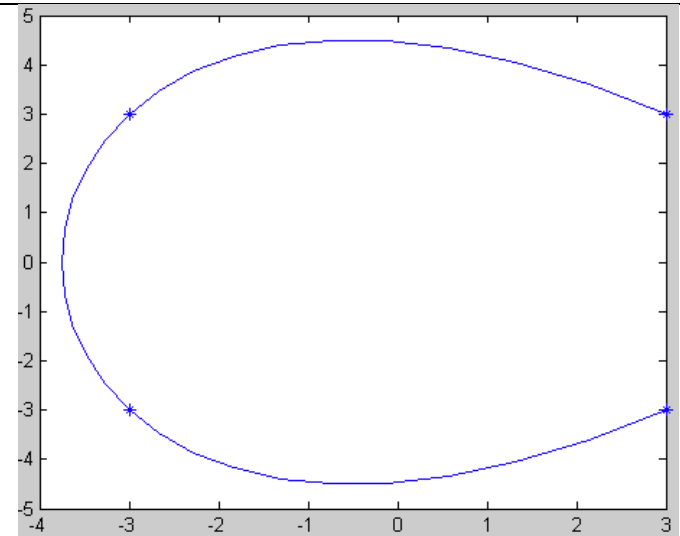
Lagrangeova interpolácia kriviek

- podobne, ako na funkcie sa dá recept aplikovať na krivky:
- $P(t) = p_0(t)P_0 + \dots + p_n(t)P_n$
- , kde $p_i(t) = \prod_{i \neq j} \frac{t-t_j}{t_i-t_j}$



Zopakovanie príkladu:

```
clear all;  
close all;  
t=0:0.1:3  
myx=3*(t-1).*(t-2).*(t-3)/(-6)-3*(t-0).*(t-  
2).*(t-3)/(2)-3*(t-0).*(t-1).*(t-3)/(-  
2)+3*(t-0).*(t-1).*(t-2)/(6);  
myy=3*(t-1).*(t-2).*(t-3)/(-6)+3*(t-0).*(t-  
2).*(t-3)/(2)-3*(t-0).*(t-1).*(t-3)/(-2)-  
3*(t-0).*(t-1).*(t-2)/(6);  
plot(myx, myy);  
hold on;  
plot([3, -3, -3, 3],[3, 3, -3, -3], '*')
```



Inteligentnejšie riešenie?

- Napr. splajny (spliny)
- Čo sú to splajny?
 - Po častiach polynomicke funkcie
 - napr. použijeme polynóm 3. Stupňa a budeme striehnuť na vlastnosti výsledku ...

Spline funkcie

Spliny sú po častiach polynomicke funkcie daného stupňa s plynulým prechodom medzi jednotlivými časťami. B-Spline $\varphi_{SM}(t)$ stupňa M je tvorený M násobnou konvolúciou „Box“ funkcie:

$$B(t) = \begin{cases} 1 & t \in \langle 0, 1 \rangle \\ 0 & \text{inde} \end{cases}$$

a má kompaktnú podporu na intervale $\langle 0, M+1 \rangle$, $M-1$ spojitých derivácií.

Platí: $\varphi_{S0}(t) = B(t) = \varphi_{Haar}(t)$, takže $\varphi_{S0}(t)$ môžeme generovať pomocou koeficientov $h(n) = (1, 1)$ resp. ich N -násobnými konvolúciami. To odpovedá Pascalovmu rojuholníku na určenie kombinačných čísiel. Koeficienty $h(n)$ pre B-spline funkcie mierky sú potom:

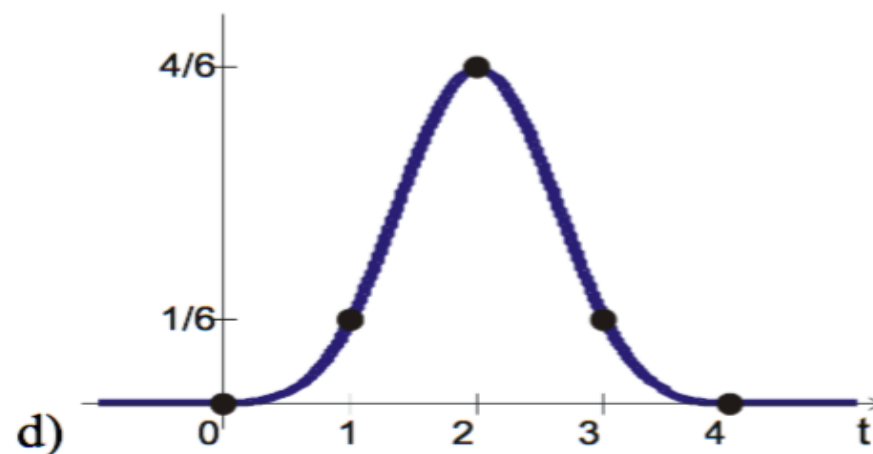
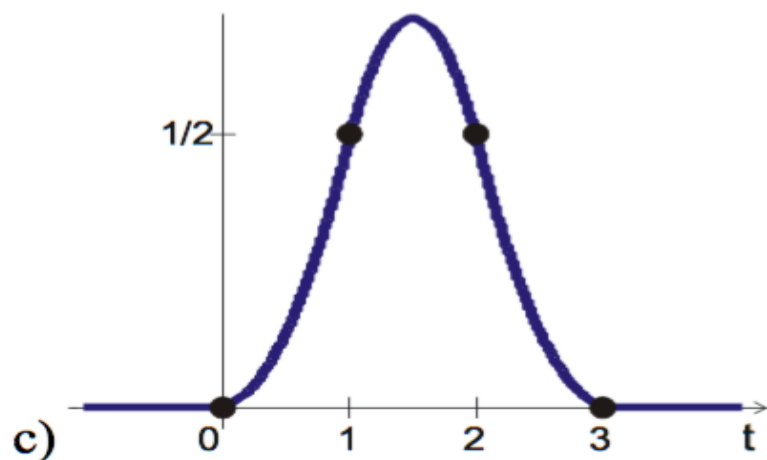
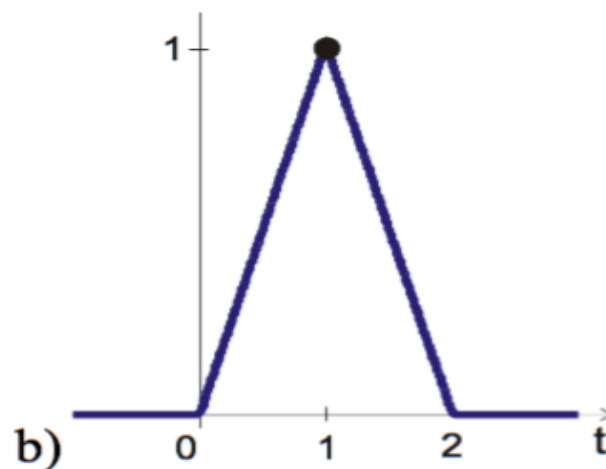
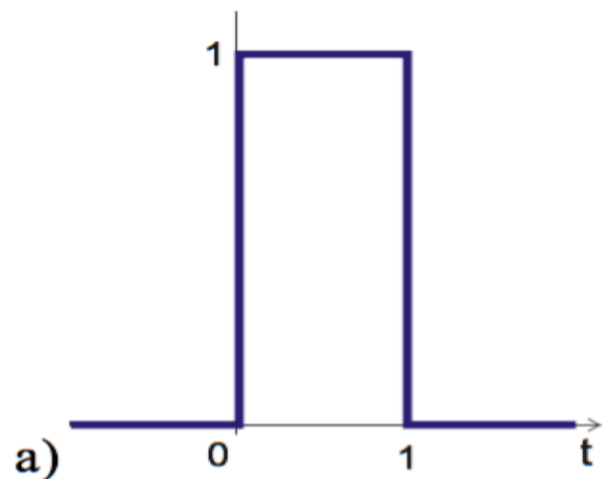
$$h_{S0}(n) = (1, 1) \frac{\sqrt{2}}{2} \quad - \text{konštantný (spline nultého rádu)}$$

$$h_{S1}(n) = (1, 2, 1) \frac{\sqrt{2}}{4} \quad - \text{lineárny (spline prvého rádu)}$$

$$h_{S2}(n) = (1, 3, 3, 1) \frac{\sqrt{2}}{8} \quad - \text{kvadratický(...)}$$

$$h_{S3}(n) = (1, 4, 6, 4, 1) \frac{\sqrt{2}}{16} \quad - \text{kubický(...)}$$

Konštantný, lineárny, kvadratický a kubický spline:



Uzavretý analytický tvar pre kubický B-spline:
[BSPLINE_TVAR]

$$h(x) = \frac{1}{6} \begin{cases} 3|x|^3 - 6|x|^2 + 4 & 0 \leq |x| < 1 \\ -|x|^3 + 6|x|^2 - 12|x| + 8 & 1 \leq |x| < 2 \\ 0 & 2 \leq |x| \end{cases}$$

Interpolácia kriviek? Pomocou kubických spline funkcií

- zabezpečia, že
 - výsledný signál je spojitý
 - jeho prvá derivácia je spojitá
 - jeho druhá derivácia $s''(x)$ je spojitá
 - zabezpečuje že $\int_{x_1}^{x_n} |s''(x)|^2 dx$ je minimálne
- parametre kubického spline sa počítajú pre jednotlivé intervaly (susedné dvojice bodov signálu) – iný prístup k návrhu ako pri návrhu interpolačného konvolučného jadra
 - existuje práve jedna priamka ktorá pretína ľubovoľnú dvojicu bodov
 - existuje nekonečne veľa parabol, ostáva 1 stupeň voľnosti
 - existuje nekonečne veľa kubických funkcií ostávajú 2 stupne voľnosti
 - existujúce stupne voľnosti sa využívajú aby bola rovnaká prvá derivácia pri nadväzovaní kriviek pre susedné intervaly
 - pri n interpolovaných bodoch je potrebné vyriešiť sústavu n lineárnych rovníc o n neznámych
- Prirodzený (natutal) kubický spline – má pri krajných bodoch zvolenú druhú deriváciu rovnú 0