Zobrazovanie 2D kriviek
4. cvičenie IZG

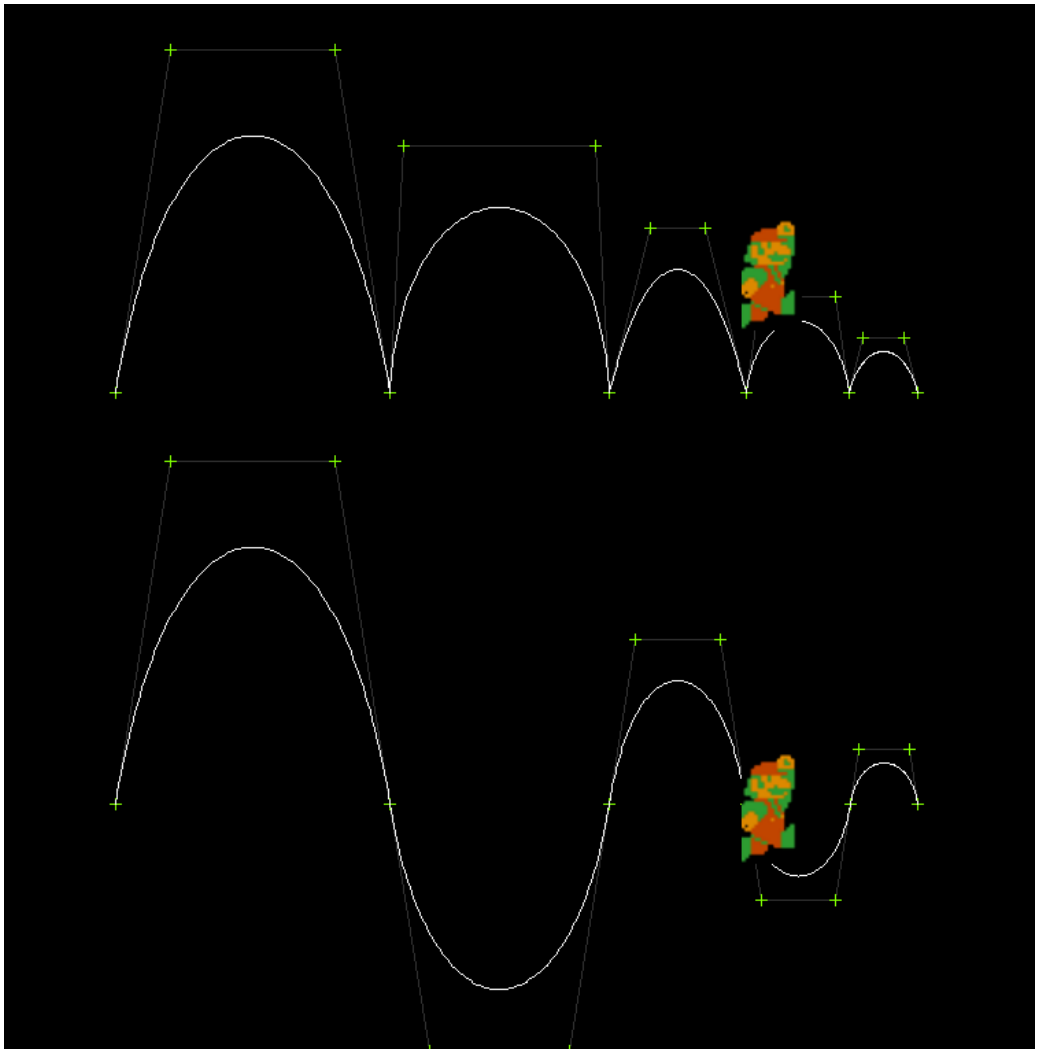
Martin Veľas

Ústav Počítačové Grafiky a Multimédií
ivelas@fit.vutbr.cz

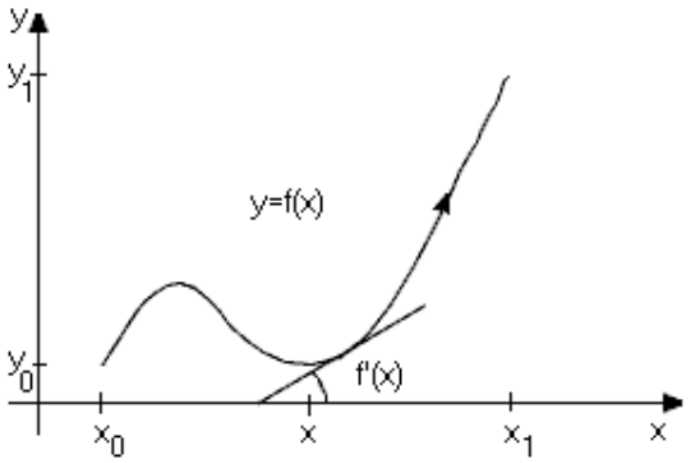
19. 3. 2018

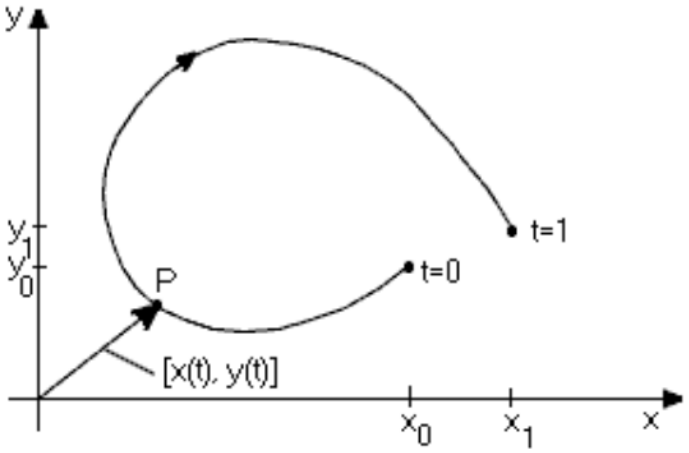
Obsah

- Úvodná časť
- Rekapitulácia 2D kriviek
- Beziérova kubika
- Úloha
  - Výpočet trajektórie pre animáciu pomocou Beziérových kubík.



- použití 2D křivek
  - definice fontů
  - šablonování
  - definice objektů
  - animační křivky
  - …
- vyjádření křivek
  - explicitní
  - implicitní
  - parametrické

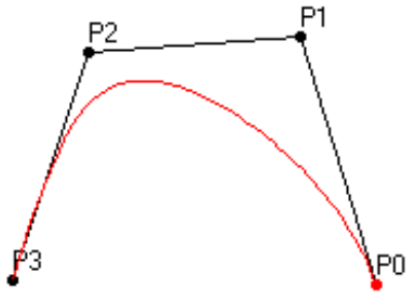


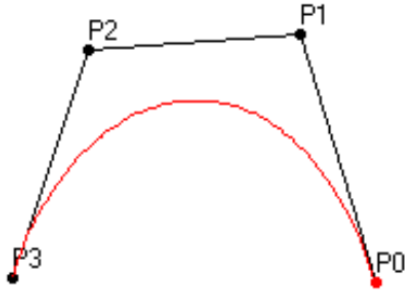


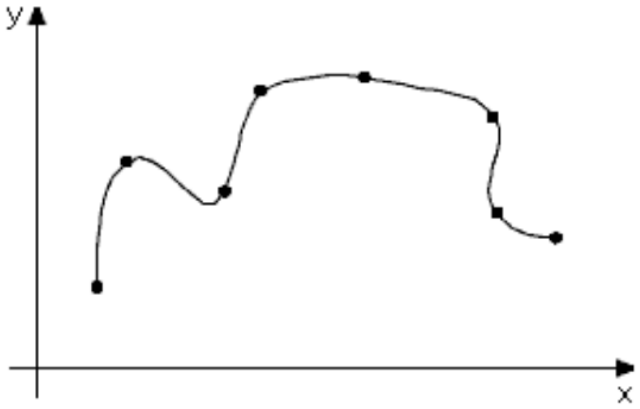
[F. Alexandr]

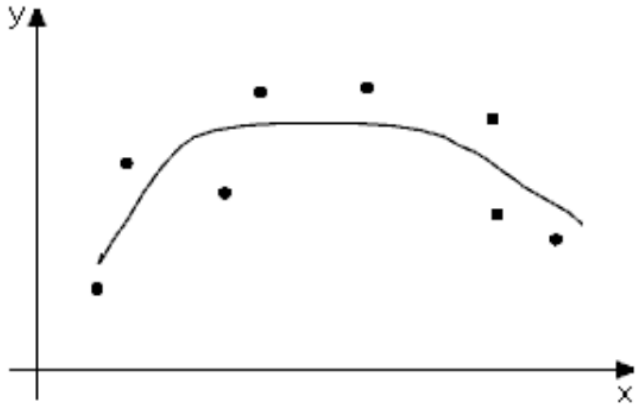
Typy křivek

- aproximační
- interpolační
- racionální
  - váhové koeficienty řídících bodů
- neracionální
  - váhové koeficienty rovny jedné





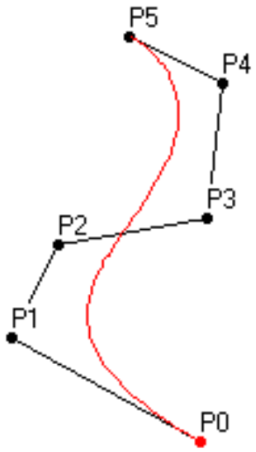




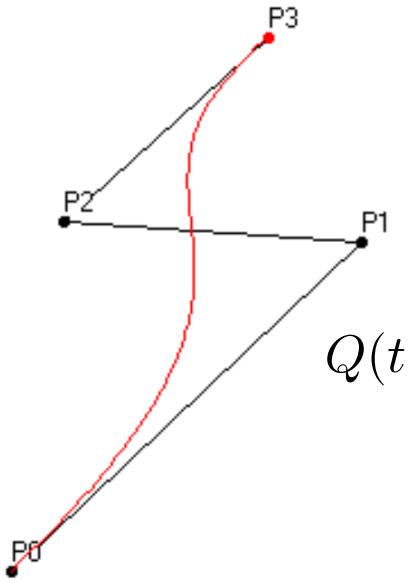
[F. Alexandr]

Bézierova křivka

- její první užití se datuje do 60 let – CAD aplikace
  - automobilový design
- TrueType i Postscript písma
- parametrické vyjádření
  - řád polynomu = stupeň křivky je počet řídících bodů křivky - 1
- aproximační křivka
- vždy prochází počátečním a koncovým řídícím bodem
- racionální vs. neracionální



- počet riadiacich bodov = 4
  - P_0, P_1, P_2, P_3



$$Q(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

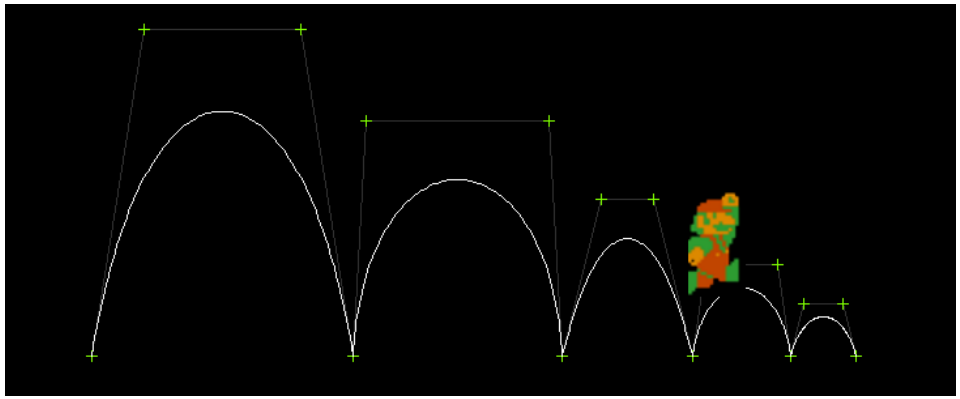
$$Q(t) = P_0 B_0^3(t) + P_1 B_0^3(t) + P_2 B_2^3 + P_3 B_3^3(t) = \sum_{i=0}^{3} P_i B_i^3(t)$$

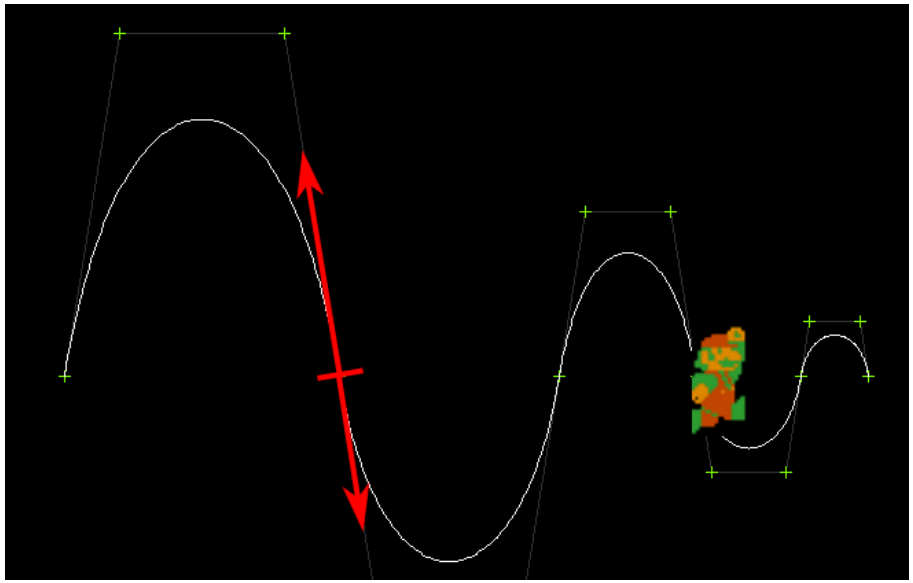
- náväznosť (spojitosť C0) ak sa zhodujú počiatočné a koncové riadiace body segmentov

$$B_0^3(t) = (1-t)^3$$
$$B_1^3(t) = 3t(1-t)^2$$
$$B_2^3(t) = 3t^2(1-t)$$
$$B_3^3(t) = t^3$$

Samostatná práce

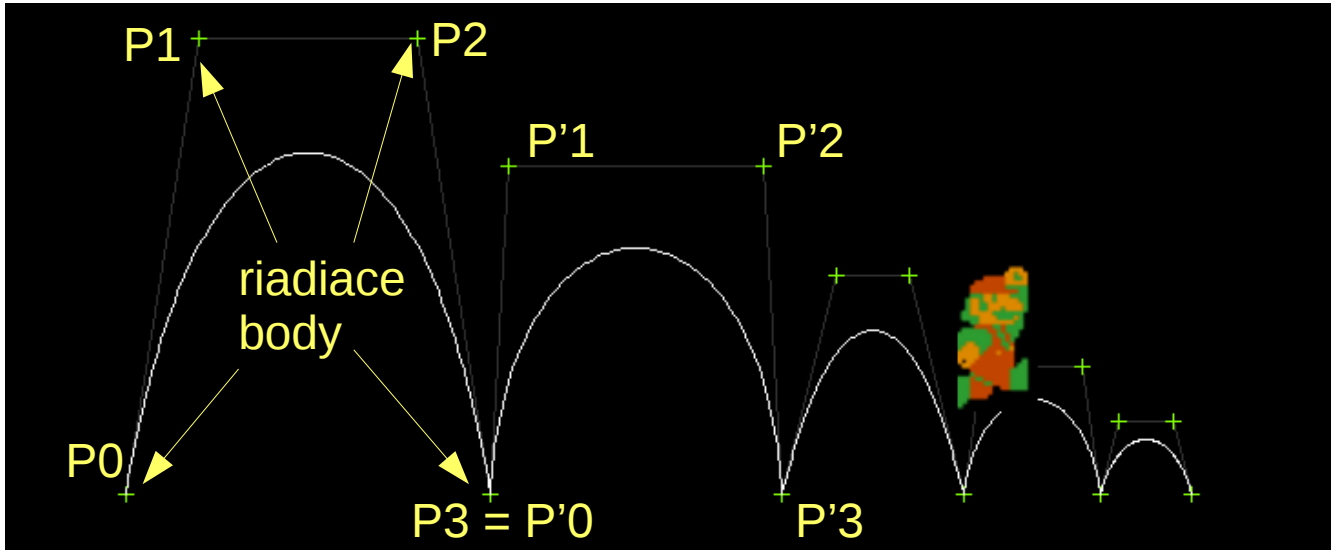
- Upravujte LEN súbor **student.cpp**
- **Úloha 1** (2b) (viď. prednáška - slide 31)
  - Výpočet trajektórie pomocou Beziérových kubík
- **Úloha 2** (2b) (viď. prednáška - slides 18, 19)
  - Úprava riadiacich bodov pre C1 spojitosť krivky





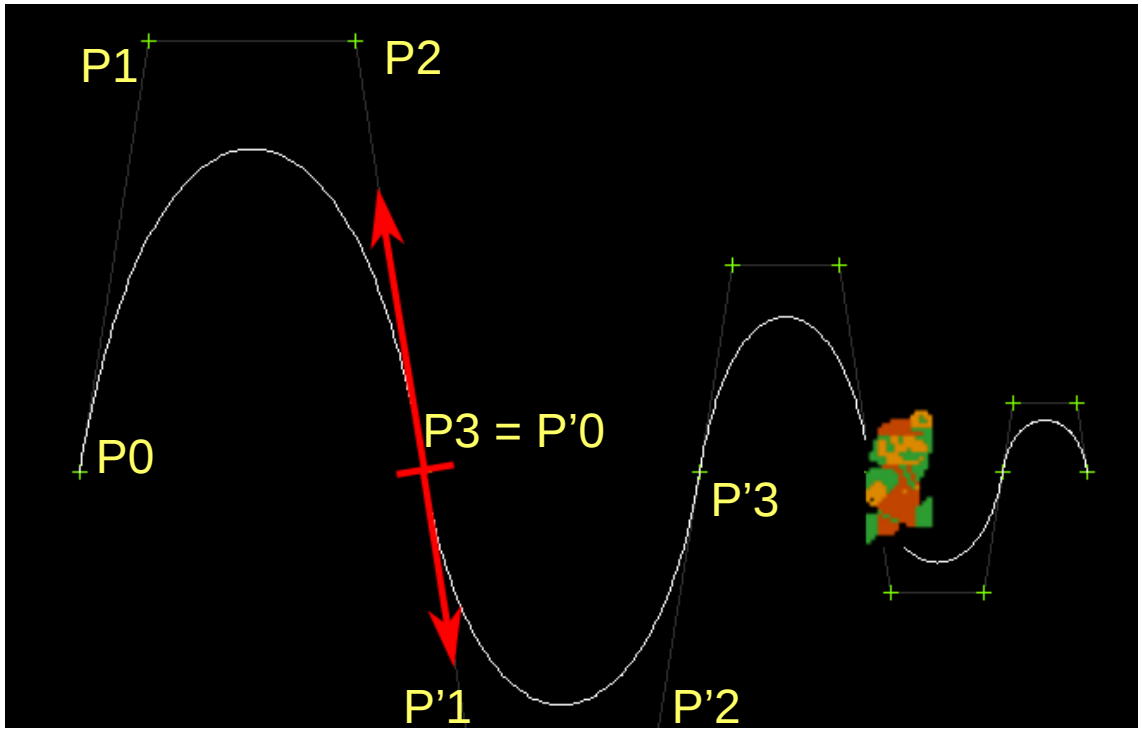
Úloha 1 (2b) – Zakrivená trajektória

- `bezierCubicsTrajectory()` výpočet trajektórie z riadiacich bodov. Volá:
- `bezierCubic()` výpočet kubiky zo 4 riadiacich bodov
  - Parameter $t \in (0;1)$, pre každú hodnotu vypočítame 1 bod
- C0 spojitosť – potrebné, opakovane použiť riadiace body. Počiatočný bod kubiky = koncový bod predošlej.



Úloha 2 (1b) – C1 spojitosť trajektórie

- úprava pozícií riad. bodov `initControlPointsDown()`
- skopírujte obsah funkcie `initControlPointsUp()`, ručne spočítajte nové súradnice a zmeňte hodnoty v kóde
- Vektor $(P_2 - P_3)$ má opačný smer ako $(P'_1 - P'_0)$



Cvičení – vector.h

- Vektor je typu **S_Vector** (vektor prvků Point2d)
  - struct Point2d { float x, y, w};
- **#define point2d_vecGet(pVec, i) (*point2d_vecGetPtr((pVec), (i)))**
  - definice makra pro získání i-tého prvku s vektoru pVec
- **S_Vector* vector = point2d_vecCreateEmpty()**
  - vytvoří prázdný vektor pro prvky typu Point2d
- **point2d_vecSize(pVec)** – vrátí velikost vektoru
- **point2d_vecGetPtr(pVec, i)** – ukazatel na i-tý prvek vektoru pVec
- **point2d_vecPushBack(pVec, p)** – vloží na konec vektoru pVec prvek p
- **point2d_vecSet(pVec, i, p)** – vloží prvek p do vektoru pVec na index i
- **point2d_vecRelease(&pVec)** vs. **point2d_vecClean(pVec)** – zruší kompletně celý vektor pVec – již s ním nelze pracovat vs. smaže prvky vektoru – vektor je prázdný o velikosti 0
- Další funkce viz vector.h