Zobrazování 2D křivek

5. cvičení IZG

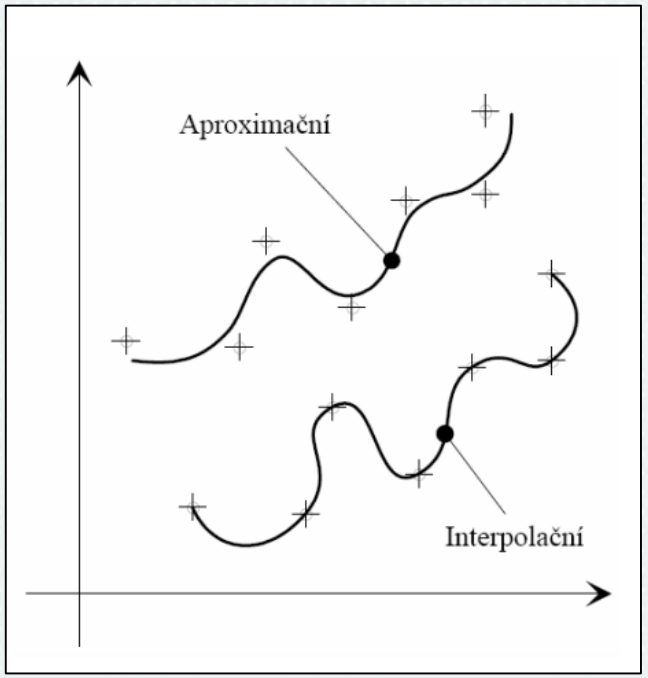
Autor: Lukáš Teuer

4. duben 2019

Opakování – 2D křivky

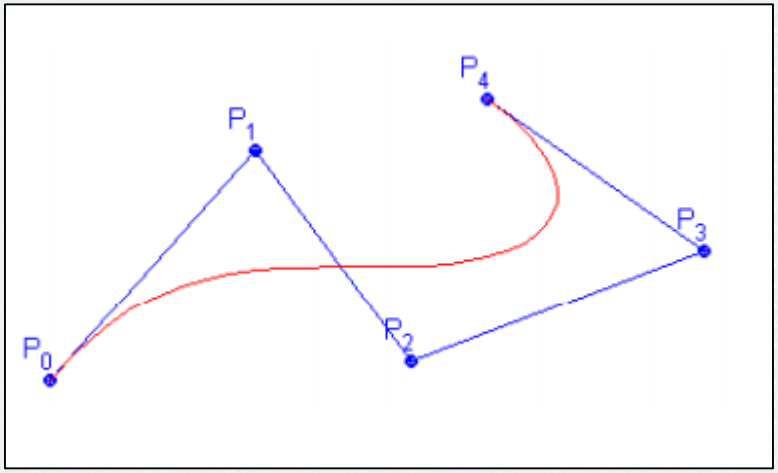
- Aproximační vs. interpolační
- Racionální vs. neracionální

- Různé druhy zápisu
  - matematicky, parametricky, maticově, a další



Opakování – Bezierové křivky

- Parametrické vyjádření
  - řád popisného polynomu – počet řídících bodů - 1
- Aproximační křivka
  - + prochází počátečním i koncovým bodem
- Může být racionální i neracionální

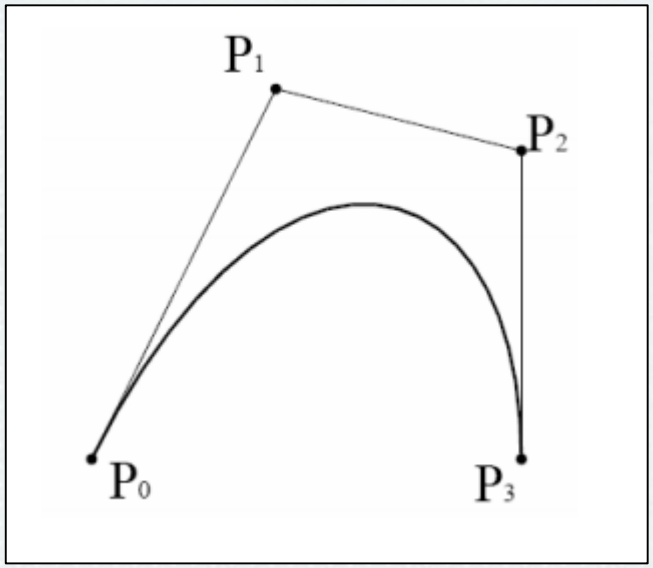


Opakování – Bezierové kubiky

- Bezierova křivka 3 řádu, popsaná čtyřmi řídícími body
- Segment, který se dá spojovat s dalšími Bezierovými kubikami
    - používá se zde vlastnost spojitost, která ukazuje jak dobře jednotlivé segmenty na sebe navazují (C0,C1,C2)

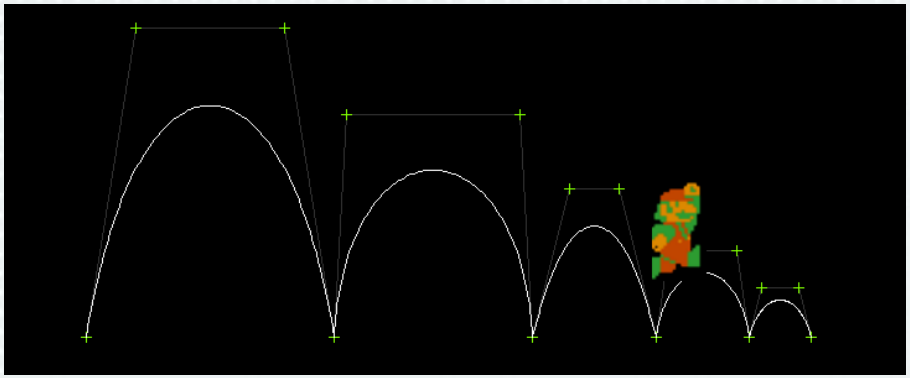
$$Q(t) = P_0 B_0^3(t) + P_1 B_0^3(t) + P_2 B_2^3 + P_3 B_3^3(t) = \sum_{i=0}^{3} P_i B_i^3(t)$$

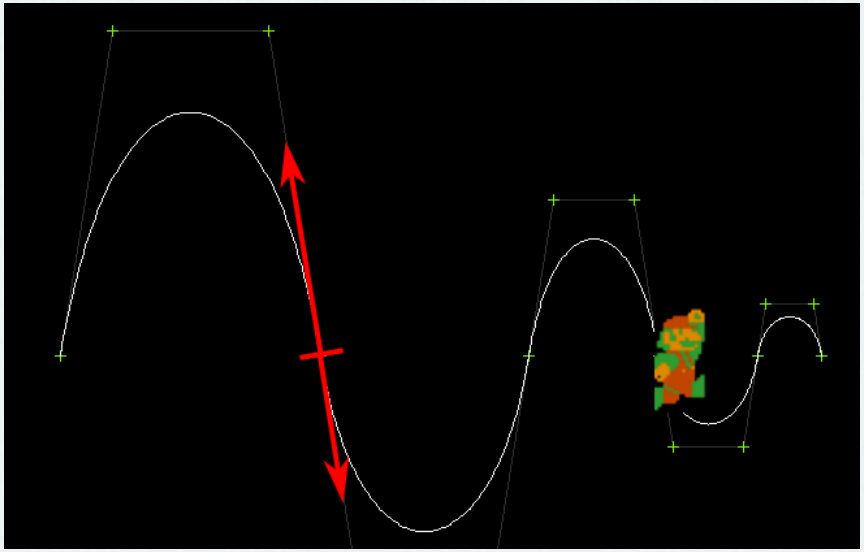
$$B_0^3(t) = (1-t)^3$$
$$B_1^3(t) = 3t(1-t)^2$$
$$B_2^3(t) = 3t^2(1-t)$$
$$B_3^3(t) = t^3$$



Obsah cvičení

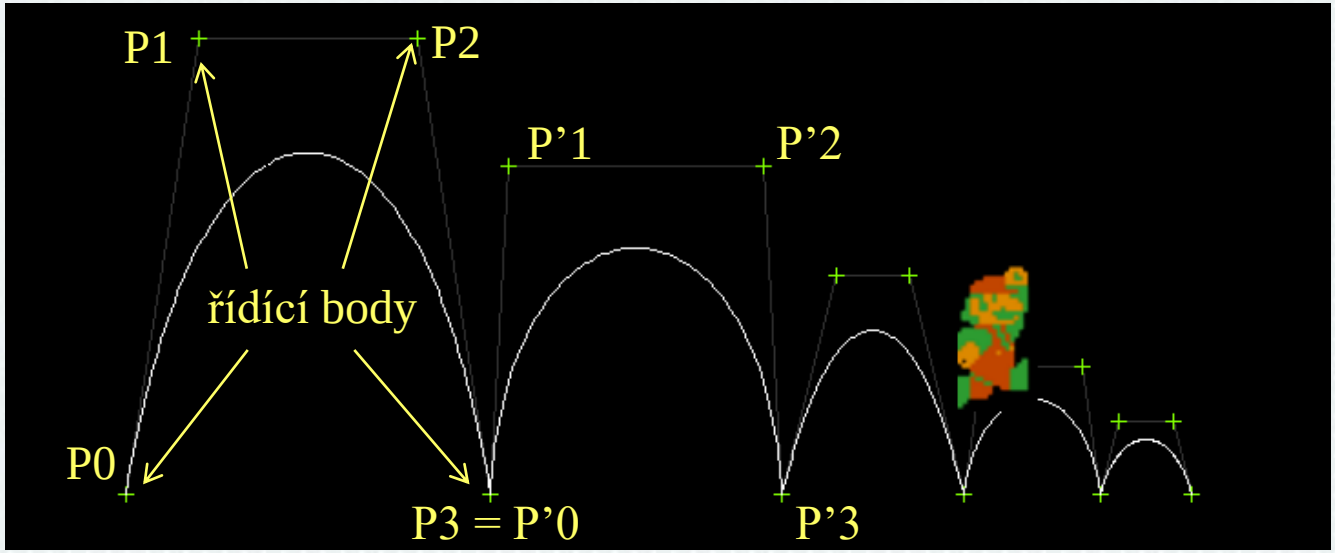
- Soubor student.cpp
- **Úloha 1** (2b) (viz přednáška - slide 31) – Výpočet trajektorie pomocí Beziérových kubik
- **Úloha 2** (1b) (viz přednáška - slidy 18, 19) – Úprava řídících bodů pro C1 spojitost křivky





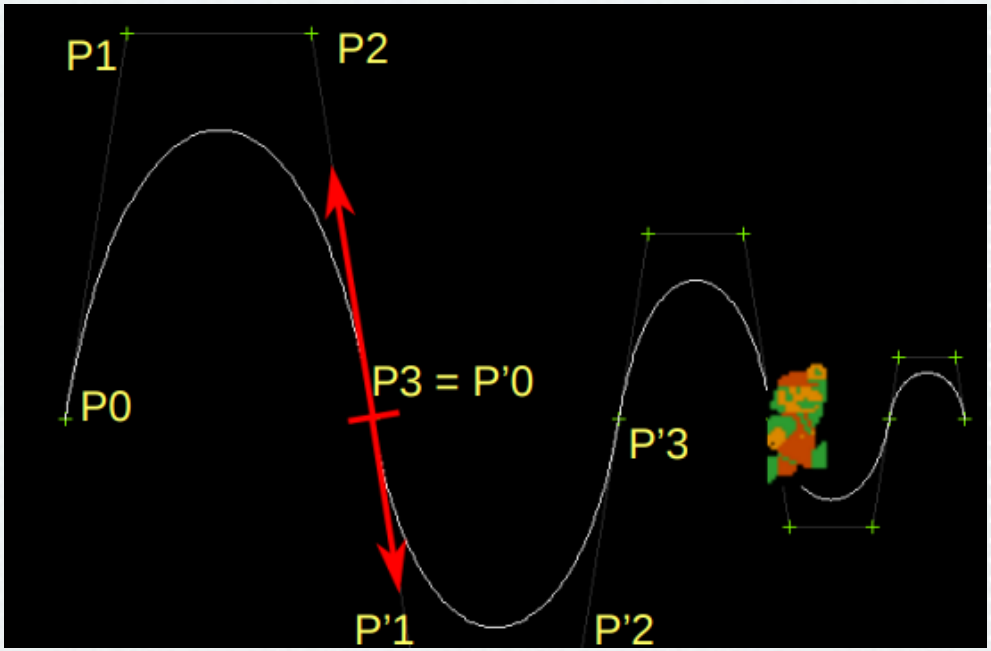
Úloha č.1 – Zakřivení trajektorie

- Ve funkci bezierCubicsTrajectory() získávání řídících bodů a následné volání funkce bezierCubic()
- Ve funkci bezierCubic() výpočet bodů trajektorie za pomocí Bezierové kubiky ze 4 řídících bodů
- Algoritmus de Casteljau
- C0 spojitost – poslední a první bod následujících kubik je stejný



Úloha č.2 – spojitost trajektorie křivky (C_1)

- Potřeba upravit polohu řídících bodů ve funkci initControlPointsDown() tak, aby byla zachována spojitost C1 (tečné přímky v bodech)
- zkopírujte obsah funkce initControlPointsUp() a ručně spočítejte nové souřadnice potřebných bodů



Pomocné funkce – vector.h

- S_Vector (vektor prvků Point2d)
  – struct Point2d { double x, y, weight};
- **point2d_vecGet(pVec, i)**
  – získání i-tého prvku z 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)
  – zruší kompletně celý vektor pVec – už s ním nelze pracovat
- point2d_vecClean(pVec)
  – smaže prvky vektoru – vektor je prázdný o velikosti 0

- Další funkce viz vector.h