Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 19 Bratislava

Programovanie pre dátovú vedu

Vykresľovanie Bézierovej krivky pomocou algoritmu de Casteljau

Dokumentácia projektu

Matej Hrnčiar
Utorok, 18:00

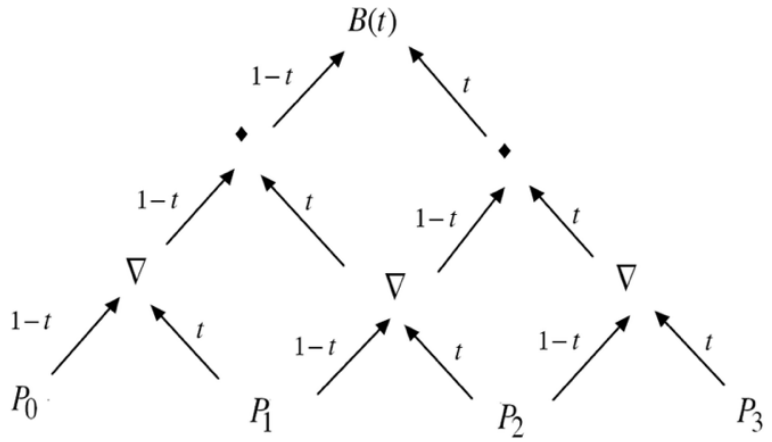
Bézierove krivky sú výrazne používané v počítačovej grafike, animácii a robotike, kedy sa vytvára súvislá krivka pohybu montážnych robotov aby sa predišlo opotrebúvaniu. Matematický základ Bézierovych kriviek bol definovaný už v roku 1912, známy ako Bernsteinove polynómy, avšak efektívny spôsob vykresľovania kriviek bol objavený až v roku 1959 Paul de Casteljauom, ktorý vymyslel výpočtovo jednoduchý a stabilný algoritmus - de Casteljauov algoritmus.

Algoritmus síce výrazne zjednodušil počítanie a tvorbu Bézierovych kriviek, ale napriek tomu môže byť pomerne zložité si ho predstaviť a pochopiť ho. Cieľom projektu je predstaviť jednoduchú animáciu interpolácie a kreslenia Bézierovej krivky použitím algoritmu de Casteljau, pričom v animácii sú vizualizované aj pomocné body a úsečky, s ktorými algoritmus pracuje.

Algoritmus

Krivka je definovaná štyrmi kontrolnými bodmi (ak ide o kubickú krivku, akú vykresľujeme v našom projekte), ktoré definujú tvar Bézierovej krivky. Dá sa použiť aj vyšší počet kontrolných bodov, ale to zvyšuje výpočtovú zložitosť a nie je to ani potrebné, nakoľko všetky krivky je možné vykresliť pomocou kriviek definovaných štyrmi bodmi, ktoré sa spájajú, vytvárajúc zložitejšie tvary. Zvyčajne sa spájanie robí tak, že posledný bod jednej krivky je zároveň prvým bodom nasledujúcej (taktiež známe aj ako C^0 spojitosť - dve krivky sú spojené), ale ak je potrebné aj súvislé zakrivenie, zdieľajú sa dva (C^1 spojitosť - smer a "rýchlosť" pohybu sa nemení skokom) alebo až tri (C^2 spojitosť - krivka nemení "zrýchlenie") kontrolné body medzi dvoma krivkami.

Bézierove krivky však nie sú obmedzené iba na dve dimenzie. Pridaním tretej dimenzie môžeme vytvoriť Bézierove povrchy, nazývané aj Bézierove plochy. Vo vektorovej grafike sa používajú na modelovanie nekonečne škálovateľných kriviek. Používajú sa aj pri tvorbe animácií a vyhladzovaní trajektórie kurzora v rozhraniach ovládaných pohľadom. Plochy sú však iba rozšírením kriviek a najprv je samozrejme potrebné pochopiť, ako vôbec algoritmus funguje a ako sú samotné krivky vykresľované.



Obrázok zobrazuje ako sa zo štyroch kontrolných bodov (označených P_0 až P_3) vypočíta bod Bézierovej krivky. Algoritmus sa dá považovať za rekurzívny. Medzi kontrolnými bodmi sa vytvoria tri úsečky – prvá je medzi prvým a druhým bodom, druhá medzi druhým a tretím

bodom a tretia medzi poslednými dvoma bodmi. Následne sa na každej z týchto úsečiek interpoluje bod podľa zadaného kroku *t* (označené ∇).

So vzniknutými bodmi sa vykoná ďalšia iterácia a rekurzívne sa takto pokračuje, až kým nezostane iba jeden bod, definujúci polohu krivky v danom kroku *t* (označený *B(t)*). Veľkosť *t* určuje hustotu bodov v krivke. Pre malé *t* je krivka „hranatá“ a veľké *t* naopak zvyšuje výpočtovú zložitosť, nakoľko algoritmus sa musí vykonať *t*-krát. Rozumným kompromisom je $t = 100$, čo znamená, že algoritmus sa vykoná 100-krát a krivka sa bude skladať zo 100 bodov. Algoritmus vyžaduje, aby *t* bolo z intervalu $\langle 0, 1 \rangle$, takže momentálny krok sa vydelí celkovým počtom krokov, v tomto prípade 100.

Implementácia

Na interpoláciu stačí použiť jednoduchú lineárnu interpoláciu (tzv. *Lerp*), ktorú je možné použiť aj pri troch dimenziách. Vzorec pre interpoláciu bodu p_t v kroku *t* je nasledovný: $p_t = t * p_0 + (1 - t) * p_1$, kde p_0 je počiatočný a p_1 je konečný bod úsečky. Vzorec jednoducho vraví, že sa nájde bod vo vzdialenosti *t* od bodu p_0 smerom k bodu p_1.

Obrázok naľavo zobrazuje animáciu kreslenia krivky. Kontrolné body sú označené červenou farbou. Prvé tri interpolované body sú spojené úsečkami modrej farby. Na úsečkách sú interpolované ďalšie body spojené úsečkou zelenej farby a samotná Bézierova krivka má červenú farbu. Táto krivka ešte nie je úplne dokreslená, dolu je možné vidieť, že sa nachádza na 70. kroku zo 100. Obrázok napravo zobrazuje rozšírenie algoritmu na vykresľovanie Bézierovej plochy, pričom samotná implementácia je uvedená v notebooku pod algoritmom na vykresľovanie krivky. Základ algoritmu je úplne rovnaký s rozdielom, že je potrebné pridať každému bodu tretiu dimenziu. Keďže plocha má okrem šírky aj dĺžku, samotné *t* už nie je postačujúce a je potrebné pracovať s dvoma parametrami reprezentujúcimi maximálny počet krokov do šírky a dĺžky, konkrétne *u* a *v*. Následne je potrebné pridať funkciu na vykresľovanie plôch, ktorá je do väčšej hĺbky popísaná v notebooku. Pri ploche je možné meniť veľkosť parametrov *u* a *v*, a sledovať, ako ovplyvňujú „hranatosť“ plochy a rýchlosť jej vykresľovania.