Plochy ohraničené Bézierovými křivkami

1 Úvod

Bézierovy křivky jsou parametrické křivky, které jsou základem většiny formátů a aplikací v oblasti vektorové 2D i 3D grafiky. Tato práce se zabývá vykreslováním uzavřených ploch ohraničených sérií navazujících kubických Bézierových křivek (cestou) ve dvourozměrné čtvecové mřížce. V rámci práce vznikla knihovna v jazyce C určená k vykreslování těchto ploch a ukázková aplikace umožňující interaktivní úpravu parametrů plochy ohraničené čtyřmi křivkami.

2 Bézierovy křivky

Bézierovy křivky jsou pojmenovány po francouzském inženýrovi Pierru Bézierovi, který je používal k popisu tvarů automobilových karoserií [3]. Vynalezl je matematik Paul de Casteljau, který také vymyslel numericky stabilní algoritmus na jejich vyhodnocení použitý v této práci [4]. V počítačové grafice se používají proto, že umožňují intuitivní popis hladkých křivek, které je možné vykreslit v libovolném rozlišení.

Práce se omezuje na kubické Bézierovy křivky. Jejich průběh získáme vyhodnocením následujícího vztahu pro všechna $t \in \langle 0, 1 \rangle$.

$$\mathbf{B}(t) = (1-t)^3\mathbf{P}_0 + 3(1-t)^2 t\mathbf{C}_0 + 3(1-t)t^2\mathbf{C}_1 + t^3\mathbf{P}_1 \tag{1}$$

kde P_0 a P_1 jsou koncové body křivky a C_0 a C_1 jsou kontrolní body, které určují tvar křivky mezi P_0 a P_1.

3 Postup vykreslování

Postup vykreslování je do značné míry inspirován vykreslovací částí knihovny freetype [1]. Při vykreslování plochy jsou postupně řádek po řádku vykreslovány úsečky ležící na daném řádku mezi okraji plochy. Pro vykreslení plochy tedy stačí získat seznam průsečíků hranic plochy s jednotlivými řádky obrazu, tyto průsečíky roztřídit na „levé“ a „pravé“ a řádek po řádku naplnit plochu úsečkami.

Pro rozdělení úseků cesty na levé a pravé je třeba zavést pravidlo určující pořadí bodů cesty. Práce počítá s tím, že jsou body cesty uvedené po směru hodinových ručiček. Když je pořadí

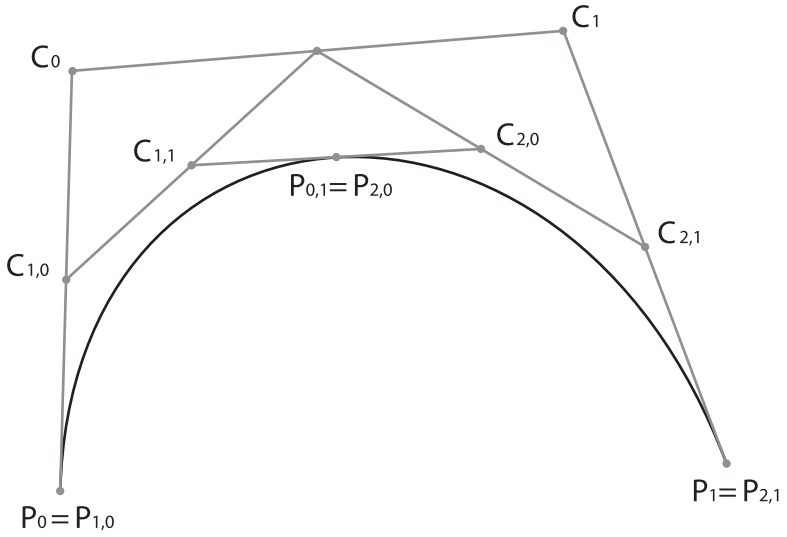
bodů na cestě dáno, je možné rozdělit úseky cesty na levé a pravé podle jejich monotonie ve svislé ose. Při pořadí po směru hodinových ručiček jsou levé úseky rostoucí a pravé klesající. Pro účely práce rozlišujeme čtyři možné typy křivek: rostoucí, klesající, ploché a ty, jejichž monotonii nelze určit. Cílem vykreslování je získat jen úseky cesty, které jsou rostoucí nebo klesající a pro ně následně určit průsečíky s jednotlivými řádky obrazu. Toho je možné dosáhnout postupným půlením křivek na základě de Casteljauova algoritmu.

S použitím naivního přístupu je možné vykreslit Bézierovu křivku postupným dosazováním hodnot t s pevným krokem do polynomu (1). Pohyb po křivce v závislosti na t ale není lineární a volba vhodného kroku tak, abychom získali právě jednu hodnotu pro každý řádek, který křivka protíná, je tedy velmi obtížná. Na základě de Casteljauova algoritmu je ale možné křivku rozdělit na dvě v bodě s libovolnou hodnotou parametru t. Tato práce používá rekurzivní dělení uprostřed křivky, které je možné vypočítat pouze s použitím sčítání a dělení dvěma (to je možné realizovat rychlou operací bitového posunu). Body dvou nových křivek $\mathbf{P}_{1,0}, \mathbf{C}_{1,0}, \mathbf{C}_{1,1}, \mathbf{P}_{1,1}$ a $\mathbf{P}_{2,0}, \mathbf{C}_{2,0}, \mathbf{C}_{2,1}, \mathbf{P}_{2,1}$ získáme z bodů původní křivky $\mathbf{P}_0, \mathbf{C}_0, \mathbf{C}_1, \mathbf{P}_1$ podle následujících vztahů.

$$\mathbf{P}_{1,0} = \mathbf{P}_0, \quad \mathbf{C}_{1,0} = \frac{\mathbf{P}_0 + \mathbf{C}_0}{2}, \quad \mathbf{C}_{1,1} = \frac{\mathbf{C}_{1,0} + \frac{\mathbf{C}_0 + \mathbf{C}_1}{2}}{2}, \quad \mathbf{P}_{1,1} = \frac{\mathbf{C}_{1,1} + \mathbf{C}_{2,0}}{2}$$

$$\mathbf{P}_{2,0} = \frac{\mathbf{C}_{1,1} + \mathbf{C}_{2,0}}{2}, \quad \mathbf{C}_{2,0} = \frac{\frac{\mathbf{C}_0 + \mathbf{C}_1}{2} + \mathbf{C}_{2,1}}{2}, \quad \mathbf{C}_{2,1} = \frac{\mathbf{C}_1 + \mathbf{P}_1}{2}, \quad \mathbf{P}_{2,1} = \mathbf{P}_1$$

Graficky je půlení znárorněno na obrázku 1.



Obrázek 1: půlení křivky na základě de Casteljauova algoritmu

Monotonní úseky křivek (hrany) získáme postupným půlením křivek dokud nejsou jednotlivé úseky rostoucí, nebo klesající. Případné ploché (vodorovné) úseky křívek vypouštíme, protože pro vykreslení nemají smysl. Úseky o jejichž monotonii nelze rozhodnout dále půlíme.

Hrany rastrujeme (jejich průsečíky s řádky obrazu získáme) dalším půlením, dokud jednotlivé úseky křivek neprotínají jen jeden řádek obrazu. Podobný přístup by bylo možné použít i pro vykreslení křivky samotné. Bylo by jen třeba upravit pravidlo pro zastavení půlení.

Získané hrany - nyní již pole vodorovných souřadnic průsečíků a mezí ve svislé souřadnici - spojíme do seznamu. Pro každý řádek obrazu ze seznamu vyjmeme hrany začínající na daném řádku a zařadíme je do seznamu levých, respektive pravých hran (podle monotonie) tak, aby seznamy byly seřazeny podle vodorovné souřadnice hrany na aktuálním řádku. Pak zbývá na daném řádku vykreslit úseky mezi vodorovnými souřadnicemi levých a pravých hran na stejné pozici v levém a pravém seznamu. Podrobnosti udržování seznamů jsou uvedeny v následující části.

4 Implementace v C a dokumentace API

Vykreslování ploch je implementováno jako knihovna v jazyce C. K samotnému vykreslování bodů na obrazovku je použita přenositelná knihovna SDL. Ukázková aplikace byla úspěšně otestována v operačních systémech Linux a Mac OS X. Měla by fungovat bez problému i v systému Windows (s nainstalovanou knihovnou SDL), ale autor neměl možnost funkčnost ověřit.

4.1 Číselný formát souřadnic

Všechny souřadnice bodů křivek jsou uloženy ve formátu s pevnou pozicí desetinné čárky 26,6 (26 bitů na celou část, 6 bitů na neceločíselnou část). Tím získáme pevnou přesnost souřadnic (na jednu čtyřiašedesátinu řádky obrazu) a možnost dělit dvěma bitovým posunem o jedna vpravo. Průsečíky křivky s řádkou se počítají ve vodorovné polovině řádku.

4.2 Výpočet hran plochy

Hranou v rámci knihovny rozumíme pole vodorovných souřadnic průsečíků monotonního úseku křivky s řádky obrazu a informace o svislých mezích a monotonii křivky.

Knihovna udržuje vnitřní stav vykreslování, jehož součástí jsou souřadnice aktuálního bodu.

Při vykreslování plochy uživatel knihovnu instruuje, aby k cestě připojila křivku z aktuálního bodu do nového bodu se dvěma kontrolními body. Křivka je ihned rekurzivně půlena dokud nejsou nejkratší úseky ve svislém směru kratší než řádek obrazu. Toto půlení probíhá na zásobníku křivek, který je také součástí vnitřního stavu vykreslování.

Před zpracováním každého úseku křivky na zásobníku je ověřena jeho monotonie. Je-li úsek plochý, dále se nezpracovává. Když není monotonii možné rozhodnout, je křivka dále rozdělena. Pokud je monotonie rozhodnutelná, vypočítá se počet průsečíků křivky s řádky obrazu (výsledek je 0, 1 nebo *více*). Úsek bez průsečíku s řádkou je ze zásobníku odstraněn. Úsek s více průsečíky je dále půlen.

Při nalezení úseku, který má pouze jeden průsečík s řádkou obrazu (a je navíc ve svislém směru kratší než řádka) se přidá vodorovná souřadnice průsečíku k poslední nalezené hraně (a je-li třeba, upraví se její svislé meze). Seznam hran a jejich průsečíků je také součástí vnitřního stavu vykreslování. Liší-li se nový úsek ve své monotonii od monotonie poslední nalezení hrany, je založena nová hrana.

Výsledkem výpočtu je seznam hran a jejich průsečíků s řádky obrazu. Vzhledem k algoritmu půlení křivek na zásobníku jsou jednotlivé průsečíky hran seřazené proti směru křivky.

4.3 Vykreslení obrazu

Po skončení definice všech křivek ohraničujících plochu uživatel instruuje knihovnu k vykreslení obrazu. Před vykreslením obrazu je třeba seřadit seznam hran získaný v minulém kroku podle vodorovné souřadnice horního průsečíku (prvního u rostoucích a posledního u klesajících hran). Toho se následně využívá při vykreslování jednotlivých řádek. K tomuto účelu knihovna používá algoritmus bubble sort, který byl vybrán pro jednoduchost implementace a předpokladu, že hran budou desítky, maximálně stovky. Toto třídění probíhá jen jednou pro každé vykreslení obrazu. Záměna tohoto algoritmu za rychlejší je možným budoucím vylepšením knihovny.

Po seřazení seznamu hran je možné začít vykreslovat jednotlivé řádky. Před vykreslením každého řádku je třeba odstranit již vykreslené hrany a přemístit hrany, které na daném řádku začínají z původního seznamu do jednoho ze seznamů levých a pravých hran. Protože samotné vykreslení počítá s tím, že jsou levý a pravý seznam seřazeny podle vodorovné souřadnice průsečíku s akutální řádkou obrazu, je třeba nové hrany zařadit do seznamu tak, aby se toto seřazení neporušilo. S využitím předchozího seřazení původních hran je možné toto zařazení

provést na jeden průchod všemi třemi seznamy. Ještě předtím je ale třeba levý a pravý seznam znovu uspořádat, protože mohlo (při křížení dvou hran) dojít k porušení uspořádání. K tomu je opět použit algoritmus bubble sort, zde s předpokladem, že k záměně pořadí ve větším rozsahu než u dvojic dojde jen velmi zřídka (při křížení tří a více levých, nebo pravých hran v jednom bodě) a počet iterací algoritmu tak bude minimální (v praxi většinou jen jedna, která uspořádání ověří). Poté je už možné vykreslit jednotlivé úsečky na daném řádku propojením odpovídajících vodorovných souřadnic hran v levém a pravém seznamu.

4.4 Práce s pamětí

Knihovna využívá pevné množství paměti. Vnitřní stav vykreslování potřebuje udržovat dvoje data proměnlivé velikosti: zásobník křivek pro půlení a data získaných hran.

Zásobník křivek je pevně omezen na hodnotu 32 křivek, ke které se v testovací aplikaci knihovna ani neblíží (tento limit teoreticky umožňuje vykreslení monotonní křivky vysoké 2^{31} pixelů, exponent u obecné křivky bude jen o málo nižší).

Omezení velikost bufferu na vykreslené hrany si volí uživatel při inicializaci knihovny. Jednotlivé hrany jsou ukládány těsně za sebou v tomto bufferu a zřetězeny do spojového seznamu.

Tímto způsobem je možné dosáhnout poměrně nízké hodnoty nutné paměti a knihovnu je tedy možné použít i v malých zařízeních s omezenou pamětí. Knihovna navíc omezuje počet dynamických alokací paměti (což je potenciálně náročná operace) na jednu. Ukázková aplikace používá buffer o velikosti 4 kB.

4.5 Dokumentace API

Protože knihovna používá vlastní formát ukládání souřadnic, je třeba kontrolní body vytvářet funkcí `make_control_point` a obecné body označující souřadnice funkcí `make_point`. Kontrolní body navíc rozlišujeme na body na cestě (koncové body křivek) a mimo cestu (kontrolní body křivek).

Knihovna se inicializuje voláním funkce `rasterizer_init` s jedním číselným parametrem udávajícím omezení velikosti paměti pro data hran.

Samotné instrukce pro kreslení je možné zadávat dvěma způsoby. První možnost je postupné volání funkcí `line_to` a `curve_to` pro kreslení rovných čar (pomocí upraveného Bresenhamova algoritmu [2]) a kubických křívek a `move_to` pro přesun aktuálního bodu bez kreslení. Druhou

možností je předem připravit spojový seznam kontrolních bodů a ten předat funkci `path` (seznam může být cyklický). Všechny funkce jako první parametr přebírají ukazatel na strukturu získanou voláním `rasteizer_init` a v dalších parametrech souřadnice bodů, nebo (v případě `path`) ukazatel na začátek seznamu kontrolních bodů. Na závěr je třeba zavolat funkci `render`, která provede vykreslení obrazu.

4.6 Ukázková aplikace

Ukázková aplikace sestaví ze čtyř křivek cestu jako přibližnou aproximaci kruhu a vykreslí ji. Kromě ní vykreslí i ovládací body křivek, které může uživatel přemisťovat tažením myši a měnit tak tvar cesty.

Aplikace má stejné požadavky k běhu jako knihovna samotná tj. knihovnu SDL. Byla testována na operačních systémech Mac OS X a Linux a měla by tedy bez problému fungovat na unixových operačních systémech s knihovnou SDL. Na těchto systémech je možné aplikaci sestavit příkazem `make` a spustit příkazem `./shape`.

5 Závěr

V rámci práce byl prozkoumán postup vykreslování ploch ohraničených sérií Bézierových křivek. Dále byla vytvořena knihovna, která tento postup implementuje a kterou by bylo možné rozšířit pro použití v širokém spektru aplikací v oblasti počítačové grafiky (např. vykreslování písma, obecná vykreslovací knihovna pro aplikace, vektorový grafický editor, čtečka PDF dokumentů...). Ukázková aplikace, která tuto knihovnu používá pak může sloužit jako ukázka k vysvětlení principu Bézierových křivek.

Reference

[1] Turner, D.: How FreeType's rasterizer work. 2003, dostupný z WWW: <http://svn.dsource.org/projects/bindings/trunk/freetype/doc/raster.txt>.

[2] Wikipedia: Bresenham's line algorithm — Wikipedia, The Free Encyclopedia. 2010, dostupný z WWW: <http://en.wikipedia.org/w/index.php?title=Bresenham's_line_algorithm&oldid=339423145>.

[3] Wikipedia: Bézier curve — Wikipedia, The Free Encyclopedia. 2010, dostupný z WWW: <http://en.wikipedia.org/w/index.php?title=Bézier_curve&oldid=341990447>.

[4] Wikipedia: De Casteljau's algorithm — Wikipedia, The Free Encyclopedia. 2010, dostupný z WWW: <http://en.wikipedia.org/w/index.php?title=De_Casteljau's_algorithm&oldid=335496222>.