

Test

Adamofus

December 21, 2022

Contents

| | | |
|----------|---|-------------|
| 1 | Projekce scény | iii |
| 1.1 | Stínění objektů v 3D | iii |
| 1.2 | Zoom | iii |
| 2 | Rasterizace | iv |
| 2.1 | Úsečka | iv |
| 2.2 | Zrcadlení | vi |
| 2.3 | Kruh | vi |
| 2.4 | Ostatní | vii |
| 2.5 | Bezierova křivka | vii |
| 2.6 | Vyplňování | vii |
| 3 | Analýza | viii |
| 3.1 | Detekce kolize | viii |
| 3.2 | Oblasti průniku dvou objektů a průřez | viii |
| 3.3 | Obsah polygonu | viii |
| 3.4 | Je A v B? | viii |
| 4 | Rotace | ix |
| 4.1 | 2D | ix |
| 4.2 | 3D | ix |
| 4.3 | Posunutí středu a osy otáčení | x |
| 5 | Generování tvarů a fraktály | xi |

Chapter 1

Projekce scény

Projekcí scény je v této práci myšleno perspektivní vidění. Paprsek jdoucí od promítaného bodu do oka pozorovatele se promítá na určenou rovinu v prostoru, tz. tvoří průnik s touto rovinou. Výsledný bod je přenesený na 2D soustavu souřadnou v této rovině.

1.1 Stínění objektů v 3D

1.2 Zoom

Chapter 2

Rasterizace

Rasterizace je proces převodu vektorově definované grafiky do tzv. rastru, tedy mřížky skládající se z bodů (pixelů). Takový rastr je základem obrazového výstupu na digitálních zařízeních. V následujících kapitolách jsou představeny algoritmy rasterizace 2 objektů: úsečky a kruhu. Samostatná kapitola je pak věnovaná rasterizaci Beziérových křivek.

2.1 Úsečka

Převod vstupní

Úsečka je část přímky definovaná dvěma body: b_1 a b_2 . Pro následující algoritmy platí, že souřadnice x bodu b_1 je menší než x bodu b_2 , aby se mohly algoritmy posouvat o jeden dílek doprava, tj $x + 1$. Směrnice úsečky je $a = dy/dx$. Počítá se s vstupem $a \in (-1; 1)$, takže úsečka svírá s osou x úhel $0 - 45^\circ$ a b_2 je v 1. kvadrantu. Opět pro omezení na jedinou podmínku, zda je potřeba y zvětšit či nikoli.

Tyto nároky umožňují zvolit efektivní algoritmus, ale současně vyžadují převod vstupu a výsledných souřadnic.

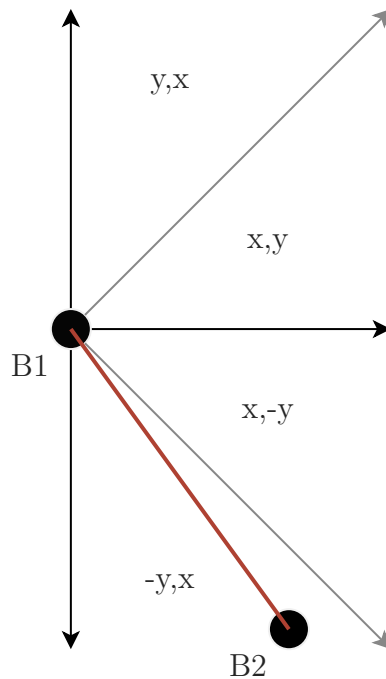


Figure 2.1: Převod souřadnic podle směrnice úsečky.

DDA

DDA využívá zaokrouhlované hodnoty $n * a$ pro výpočet y_{new} , je to prostý přístup. Současně ale zbytečně používá funkci zaokrouhlování či přetypování, kterou lze pro optimalizaci nahradit podmínkou s použitím proměnné, protože jsou pouze 2 možnosti pro y_{next} : $y_{next} = y$, nebo $y_{next} = y + 1$.

Proměnnou vyjadřuje $error(n) = ((n * a) \bmod 1) - 1/2$, kde n je krok iterace, takže $error(0) = -1/2$. Pokud platí zmíněná podmínka $error(n) \geq 0$, platí současně $y_{next} = y + 1$ a $error(n + 1) = error(n) + a - 1$, jinak platí $error(n + 1) = error(n) + a$.

Bresenhamův algoritmus

Další optimalizací je zbavení se desetinné čárky (tzv. float point number). Představme si $error(n) = 2 * dx * error(n)$. Jsou 2 možnosti:

$$error(n + 1) = error(n) + 2 * dy$$

$$error(n + 1) = error(n) + 2 * dy - 2 * dx$$

Nerovnice podmínky se nemění, protože po vynásobení pravé strany $2 * dx$: $error(n + 1) \geq 0 * 2 * dx$ zůstává stejná. Tím jsme se zbavili nutnosti použití desetinné čárky.

$$x \leftarrow x_1$$

$$y \leftarrow y_1$$

```

while  $x \leq x_2$  do
  vykresli bod[x,y]
   $x \leftarrow x + 1$ 
   $error \leftarrow error + 2d_y$ 
  if  $error \geq 0$  then
     $y \leftarrow y + 1$ 
     $error \leftarrow error - 2d_x$ 
  end if
end while

```

2.2 Zrcadlení

Zrcadlení je často využívaná operace, například pro rasterizaci kružnice. Využívá bod B a vektor \vec{v} , přes který B zrcadlíme. Výstupem je zrcadlený bod B_z . Platí, že vektor $\overrightarrow{BB_z}$ je kolmý na \vec{v} a jeho délka je dvojnásobná vzdálenosti B od \vec{v} .

$\frac{1}{2}\overrightarrow{BB_z} = (k * x + P_x; k * y + P_y) - (b_x; b_y)$, skalární součin tedy:

$$(k * x + P_x - b_x; k * y + P_y - b_y) * (x; y) = 0.$$

Úpravou rovnice vyjde $k = -\frac{x(P_x - B_x) + y(P_y - b_y)}{x^2 + y^2}$, bod získám: $B_z = 2(k * \vec{v} + P - B) = (2(k * x + P_x - B_x); 2(k * y + P_y - B_y))$

2.3 Kruh

Pro kružnici rovněž existuje Bresenhamův algoritmus. Algoritmus vykreslí 1/8 kružnice. V podmínce je tedy ze znalosti přímky svírající s osou x 45 stupňů: $x \geq y$. K vykreslení této části víme, že iterativně zvětšujeme y . x dekrementujeme, pokud je hodnota d kladná, tj. zajímají nás pouze 2 pixely. Pro vykreslení celku stačí díky symetrii kružnice tuto část 3x zrcadlit.

```

 $d \leftarrow x^2 + y^2 - r^2$ 
 $x \leftarrow r$ 
 $y \leftarrow 0$ 
 $d \leftarrow 0$ 
while  $x \geq y$  do
  vykresli bod[x,y]
   $y \leftarrow y + 1$ 
   $d \leftarrow d + d_y$ 
  if  $d \geq 0$  then
     $x \leftarrow x - 1$ 
     $d \leftarrow d - d_x$ 
  end if
end while

```

```
end if  
end while
```

Jiný algoritmus se může zakládat na výběru z 3 sousedících pixelů podle toho, který je nejbliž středu. Takové řešení je ale neefektivní, implementací se zde nezabývám.

2.4 Ostatní

2.5 Bezierova křivka

2.6 Vyplňování

Řádkovací metoda

Inverzní vyplňování

Flood fill

Chapter 3

Analýza

3.1 Detekce kolize

3.2 Oblasti průniku dvou objektů a průřez

Sutherland–Hodgman algoritmus

Weiler–Atherton clipping algoritmus

3.3 Obsah polygonu

<https://mathworld.wolfram.com/PolygonArea.html>

3.4 Je A v B?

Chapter 4

Rotace

Otáčet bod vůči středu soustavy souřadné je jako nanášet ho na otočenou soustavu souřadnou, tedy násobit vektory udávající osy x , y , atd. takové soustavy. Tyto vektory mají délku 1. Lze zapsat takto:

$$p_n = \begin{pmatrix} X_1 & Y_1 \\ X_2 & Y_2 \end{pmatrix} \vec{p}$$

4.1 2D

Otočené souřadnice můžeme vyjádřit takto:

$$\begin{aligned} X_1 &= \cos(-\alpha) = \cos(\alpha) \\ X_2 &= \sin(2\pi - \alpha) = -\sin(\alpha) \\ Y_1 &= \cos(\pi/2 - \alpha) = \sin(\alpha) \\ Y_2 &= \sin(\pi/2 - \alpha) = \cos(-\alpha) = \cos(\alpha) \end{aligned}$$

$$\begin{pmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

4.2 3D

Rotace bodu vůči ose Z ve směru hodinových ručiček:

$$\begin{pmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Rotace bodu vůči zvolené ose:

Máme vstup osu o a úhel α .

Mějme 2 soustavy souřadné A a B popsané jednotkovými vektory. Přitom A je naše výchozí, na které vykreslujeme:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Matice B má tvar:

$$\begin{pmatrix} a_x & b_x & o_x \\ a_y & b_y & o_y \\ a_z & b_z & o_z \end{pmatrix}$$

3. řádek B tvoří souřadnice osy o , takže 3. souřadnice p_b je právě vzhledem k o .

Čili je z následující rovnice zaručeno, že získáme přesně takový bod p_b , kde jeho 3. souřadnice je vzhledem k o . To potřebujeme, protože jsme zvolili matici rotace podle osy Z (respektive 3. osy...), kterou chceme bod násobit. Takže tato souřadnice bodu v B po rotaci bude stejná.

Z rovnice $p_a = p_b B$ vyjádříme tedy p_b vynásobením B^{-1} : $p_b = p_a B^{-1}$

Zbývající vektory a a b v B musí být jednotkové a vzájemně kolmé. takový a dostaneme třeba ignorováním z a přehozením souřadnic následovně: $\vec{a} = \{-y, x, 0\}$, kde x a y jsou souřadnice o . b je už jen vektorovým součinem a a b .

To je tedy převod relativních souřadnic ze soustavy A do soustavy B .

Rotace probíhá následovně:

Vstup: osa otáčení o , úhel α

1 převedeme souřadnice z A do B

2 zrotujeme (například přes matici 1.1, tj. vůči 3. ose)

3 vykreslujeme v A , takže převedeme z B do A

4.3 Posunutí středu a osy otáčení

Pro posunutý střed v 2D platí, že vektor \vec{XS} , kde S je střed a X bod, má fakticky souřadnice bodu \vec{p} . K výpočtu stačí potom přičíst S .

Stejně u posunuté osy v 3D, je takový vektor vlastně bod p_a . V praxi, grafickém editoru, můžeme osu určit typicky dvěma body. A ten jeden z nich (třeba pro intuici první určený) je totiž tento střed S . Nakonec stačí opět přičíst S .

Chapter 5

Generování tvarů a fraktály