Programátorská Encyklopedie

KORESPONDENČNÍHO SEMINÁŘE Z PROGRAMOVÁNÍ

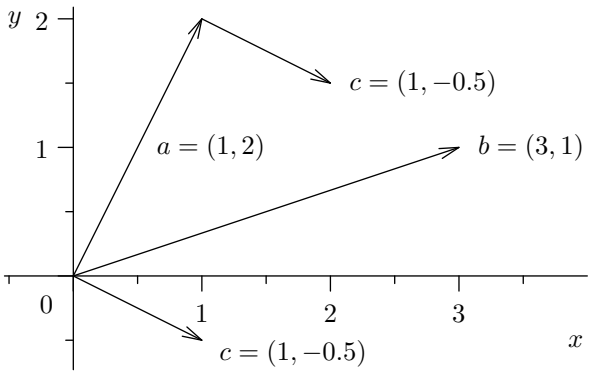
Vektory

Tento studijní text vznikl pro účely seriálu o počítačové grafice a shaderech, proto se na ně místy odkazuje.

Vektor je objekt z lineární algebry, který může být spoustou různých věcí. Pro naše účely je vektor uspořádaná n-tice desetinných čísel. Budeme pracovat jen s vektory o dvou nebo o třech číslech (či komponentách) a zaměříme se na jejich geometrický význam.

Dvourozměrný vektor obsahuje dvě čísla (či komponenty), typicky se jim říká x a y podle os souřadnic, které běžně reprezentují. Trojrozměrný k nim přidává komponentu z, čtyřrozměrný w. S vícerozměrnými vektory se u shaderů nesetkáme.

Co může takový 2D vektor popisovat? Jistě do něj můžeme uložit třeba souřadnice nějakého bodu v rovině jako dvě čísla, potom vektor tento bod popisuje. Lze se na něj dívat ale také z jiného úhlu: takový vektor totiž zároveň popisuje *rozdíl* polohy mezi počátkem souřadnic a daným bodem nebo také *směr* z počátku do nějakého bodu.



V obrázku jsou dva vektory c. Oba jsou stejné $(1, -0.5)$, jen jsou zakreslené s různými počátky. Dolní může reprezentovat souřadnice bodu na $(1, -0.5)$, horní je něco jako „směr“ z bodu $(1, 2)$.

Systém souřadnic

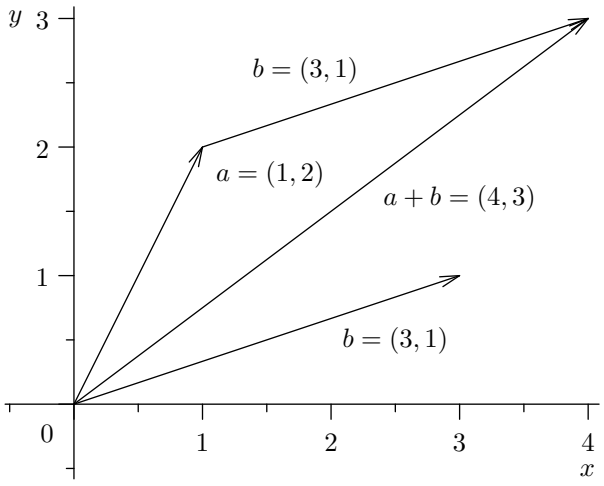
Když se bavíme o souřadnicích, měli bychom si pořádně definovat co znamenají. V celém seriálu bude platit, že x roste směrem „doprava“, y roste „nahoru“ a z roste „dozadu“. Tedy pokud je váš monitor v rovině x a y, tak z bude růst směrem ven z monitoru k vám.

Operace s vektory

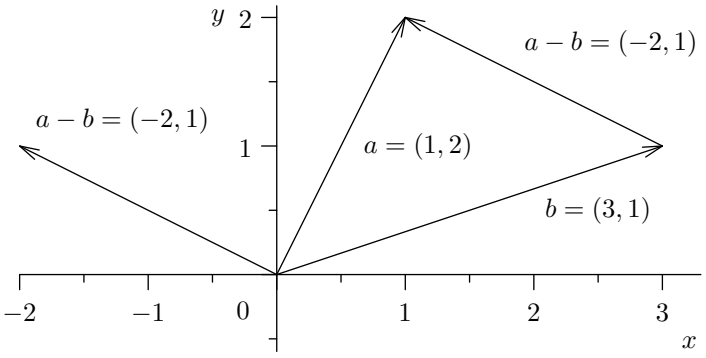
S vektory můžeme provádět několik různých operací. Tyto operace vždy provádíme na vektorech o stejném počtu komponent.

Můžeme je *sčítat:* součet dvou vektorů je vektor, jehož každá komponenta je součet odpovídajících komponent vstupních vektorů, tedy například $(1, 2) + (3, 1) = (4, 3)$. Vlastně jsme aplikovali klasickou operaci součtu dvou čísel na každou komponentu zvlášť.

Geometrický význam součtu vektorů $a + b$ je takový, že nejdříve se z počátku souřadnic přesuneme podle vektoru a a poté se z tohoto místa přesuneme podle vektoru b.



Stejným způsobem můžeme vektory i *odečíst*. Důležitou vlastností rozdílu vektorů $a - b$ je, že výsledný vektor odpovídá směru z místa b do místa a. Toto je dobré si pamatovat.

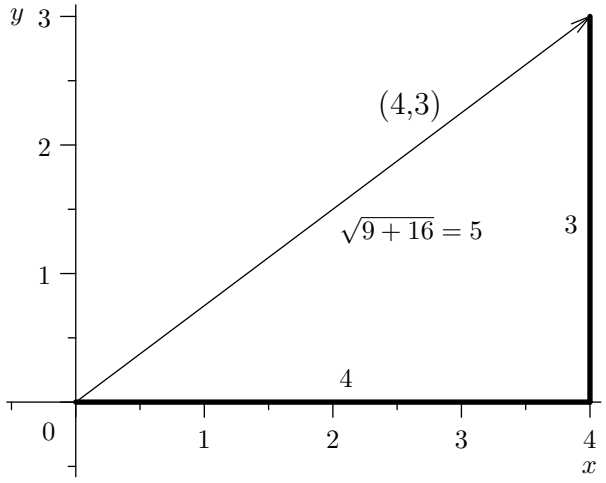


Též můžeme *po komponentách* i *násobit* a *dělit*, což vektor nějak natáhne nebo zkrátí, a to i různě v různých komponentách. Důležité je uvědomit si, že vynásobení vektorů po komponentách je jiná operace než skalární a vektorový součin, který popíšeme později. V matematice se násobení vektorů po komponentách příliš nepoužívá, v shaderech je ale velmi užitečné.

Také je občas užitečné provést nějakou operaci, třeba vynásobení, s vektorem a *skalárem* (což je obyčejné číslo, tedy „jednokomponentový“ vektor). Operaci stále provedeme zvlášť po složkách. Chová se to tedy stejně, jako bychom například násobili vektor vektorem, kde je každá komponenta nastavena na hodnotu tohoto skaláru, například $(2, 3) \cdot 4 = (8, 12)$ nebo $(1, 2) + 3 = (4, 5)$. Násobení a dělení skalárem se v matematice používá běžně, sčítání a odčítání je spíše syntaktická zkratka v shaderech.

Délka vektoru a normalizace

Délku vektoru, neboli jeho *velikost,* snadno spočítáme pomocí Pythagorovy věty. Osy x a y jsou na sebe kolmé, tudíž pokud si 2D vektor představíme jako nejdříve posunutí po x, poté po y, vznikne nám pravoúhlý trojúhelník, u něhož známe délky odvěsen (hodnoty komponent vektoru), a délka vektoru je rovna délce přepony. Stejný princip lze snadno přenést i do tří rozměrů.



Často chceme vědět jak jsou od sebe dva body a a b vzdálené. Nyní už známe vše potřebné k tomu, abychom to spočítali: nejdříve nalezneme vektor z a do b, což je $b-a$, poté spočítáme jeho délku (stejně dobře můžeme vzít vektor z b do a, tedy $a-b$).

Když přemýšlíme o nějakém vektoru jako o směru, typicky nás jeho délka nezajímá a spíše nám překáží (jak uvidíme u skalárního součinu). Proto se hodí zařídit, aby měl tento vektor délku právě jedna. Tato operace se nazývá *normalizace*. Provádí se tak, že spočteme délku daného vektoru a touto délkou ho vydělíme.

V GLSL je pro spočítání délky vektoru zabudovaná funkce `length` a pro normalizaci `normalize`.

Skalární součin

Skalární součin, anglicky *scalar product* či *dot product* (protože se typicky značí tečkou), spočítáme pro dva vektory tak, že nejdříve vynásobíme složky zvlášť (jako u komponentového násobení) a poté všechny výsledné složky sečteme. Výsledkem tedy není vektor, ale *skalár*). V *GLSL* je pro skalární součin zabudovaná funkce `dot`.

Spočítáme-li `dot(vec2(1, 2), vec2(3, 4))`, dostaneme

$$1 \cdot 3 + 2 \cdot 4 = 11.$$

Nejdůležitější vlastností skalárního součinu je, že pokud se na vstupní vektory díváme jako na směry, výsledek operace je roven kosinu úhlu mezi těmito vektory vynásobený jejich velikostmi. Pokud úhel mezi vektory a, b označíme s a velikost vektoru budeme značit jako $|a|$, platí:

$$\mathrm{dot}(a,b) = |a| \cdot |b| \cdot \cos s.$$

Pokud oba vstupní vektory nejdříve znormalizujeme, platí:

$$\mathrm{dot}(a,b) = |a| \cdot |b| \cdot \cos s = 1 \cdot 1 \cdot \cos s = \cos s.$$

Skalární součin tedy lze použít k výpočtu úhlu mezi dvěma vektory či jeho kosinu.

Vektorový součin

Vektorový součin, anglicky *cross product* (značí se typicky křížkem), je operace definovaná pouze pro trojrozměrné vektory. Vstupem jsou dva vektory a výstupem je opět vektor. Tento výstupní vektor je vždy kolmý na rovinu tvořenou vstupními vektory (stále o vektorech přemýšlíme jako o „směrech“ ukazujících z počátku souřadnic). Navíc velikost výsledného vektoru je rovna:

$$|a \times b| = |a| \cdot |b| \cdot \sin s.$$

Výsledek je kolmý na rovinu vstupních vektorů, jenže k rovině existují kolmé vektory dva (když nás zajímá jen směr a ne délka), pro vodorovnou rovinu by to byl vektor „nahoru“ a vektor „dolů“. Který z nich vektorový součin vrací, si lze zapamatovat pomocí pravidla pravé ruky: nastavte svou pravou ruku následujícím způsobem:

- Ukazováček ukazuje dopředu a symbolizuje první vstup operace.
- Prostředníček ukazuje doleva a symbolizuje druhý vstup operace.
- Palec ukazuje nahoru a symbolizuje výsledek.

Dejte ovšem pozor na to, že toto nemusí platit v každém souřadnicovém systému. Kdybychom nadefinovali, že z je *dopředu,* pravidlo by fungovalo obráceně (museli bychom použít levou ruku místo pravé).

V grafice se využívá především toho, že vektorový součin dokáže najít vektor kolmý na rovinu nějakých dvou jiných vektorů.

Barva jako vektor

V shaderech často reprezentujeme barvu jako trojrozměrný vektor. Komponenta x odpovídá červené, y zelené a z modré. Hodnoty mezi 0 a 1 odpovídají různému rozsvícení pixelu na monitoru.

Jak se barvy chovají vzhledem ke komponentovým operacím na vektorech?

Součet a je něco mezi zesvětlením a smícháním barev:

$$(0.2, 0.0, 0.0) + (0.5) = (0.7, 0.5, 0.5).$$

Násobení a dělení je zesvětlení či ztmavení při zachování odstínu a sytosti barvy, aspoň dokud hodnoty zůstávají v zobrazitelném rozsahu $0 \ldots 1$:

$$(0.2, 0.8, 0.2) \cdot 0.5 = (0.1, 0.4, 0.1).$$

Pokud hodnoty přesáhnou 1, tak se stále zobrazí jako 1: barva $(0.2, 0.8, 0.2) \cdot 4 = (0.8, 3.2, 0.8)$ se zobrazí stejně jako $(0.8, 1.0, 0.8)$

Komplexní čísla

V prvním díle grafického seriálu se i krátce setkáme s komplexními čísly, takže v rychlosti shrneme, co to vlastně je.

Co kdyby šlo odmocnit záporná čísla? To je základní myšlenka za komplexními čísly. Odmocňování záporných čísel realizujeme tak, že definujeme výsledek odmocnění -1 jako i, nazývané *imaginární jednotka*.

$$\sqrt{-1} = \mathrm{i}.$$

$$\sqrt{-4} = \sqrt{4} \cdot \sqrt{-1} = 2\mathrm{i}.$$

Co se stane, když k násobku i přičteme nebo odečteme reálné číslo? Samozřejmě se s i nemůže „zkombinovat“, takže dostaneme výraz typu $a + b\mathrm{i}$, kde a, b jsou reálná čísla. Složku a nazýváme *reálná část, b imaginární část.* Číslům, které mohou mít imaginární část, říkáme *komplexní čísla.*

Komplexní číslo v tomto tvaru se chová skoro jako dvojrozměrný vektor: a a b jsou jeho komponenty. Často komplexní čísla vizualizujeme v rovině stejně jako vektory. Sčítání a odčítání komplexních čísel se chová úplně stejně jako u dvojrozměrných vektorů, tedy $(a + b\mathrm{i}) + (c + d\mathrm{i}) = ((a + b) + (c + d)\mathrm{i})$.

Podobně se také chová absolutní hodnota komplexních čísel – je totiž definována stejně jako délka vektoru.

Násobení a dělení komplexních čísel jinými komplexnímy čísly se ale chová úplně jinak. My si ukážeme jen násobení, které lze snadno odvodit:

$$(a + b\mathrm{i}) \cdot (c + d\mathrm{i}) = a \cdot c + ad\mathrm{i} + bc\mathrm{i} + bd\mathrm{i}^2 =$$
$$= ac + (ad + bc)\mathrm{i} - bd.$$

Násobení a dělení komplexního čísla reálným číslem se chová stejně jako u vektorů: $(a + b\mathrm{i}) \cdot c = ac + bc\mathrm{i}$

Článek pro vás sepsal
Kuba Pelc