

Základy Počítačové grafiky

Tomáš Maršálek

27. října 2012

1 Úvod

Pro semestrální práci jsem po osobní dohodě se cvičícím zvolil jazyk Java, poté podle vlastního výběru jeden z nejpoužívanějších wrapperů OpenGL pro Javu - **Lightweight Java Game Library (LWJGL)**.

2 Výpočty

2.1 Terén

Terén je načten ze souboru a v programu používán jako objekt třídy **Terrain**. Terén je $H - 1$ trojúhelníkových pruhů (*GL_TRIANGLE_STRIP*) délky W . Objekt **Terrain** ulehčuje práci pomocí metod umožňujících získání výšky nebo normály v daném bodě této trojúhelníkové sítě.

2.2 Výpočet normál terénu

Normála v každém bodě trojúhelníkové sítě je vypočtena jako průměr normál čtyř okolních myšlených trojúhelníků, popřípadě tří normál na hraně terénu nebo dvou v rozích. Výsledek je pak znormalizován na jednotkovou velikost.

$$\mathbf{n1} = [\mathbf{terrain}(x + 1, z) - \mathbf{terrain}(x, z)] \times [\mathbf{terrain}(x, z + 1) - \mathbf{terrain}(x, z)]$$

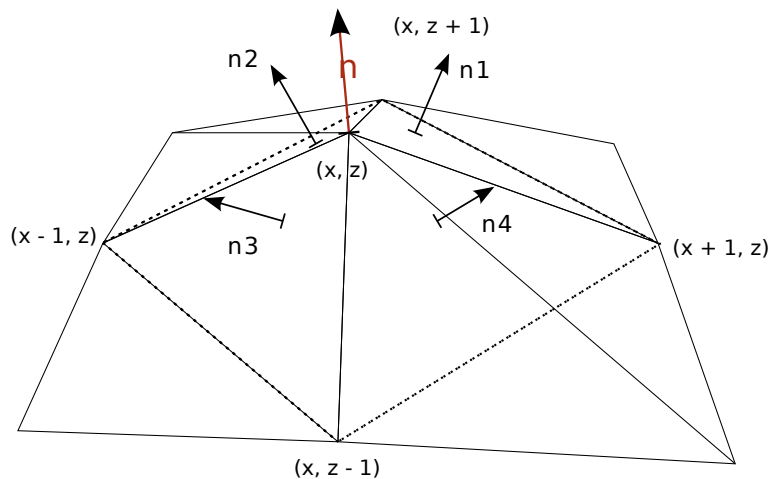
$$\mathbf{n2} = [\mathbf{terrain}(x, z + 1) - \mathbf{terrain}(x, z)] \times [\mathbf{terrain}(x - 1, z) - \mathbf{terrain}(x, z)]$$

$$\mathbf{n3} = [\mathbf{terrain}(x - 1, z) - \mathbf{terrain}(x, z)] \times [\mathbf{terrain}(x, z - 1) - \mathbf{terrain}(x, z)]$$

$$\mathbf{n4} = [\mathbf{terrain}(x, z - 1) - \mathbf{terrain}(x, z)] \times [\mathbf{terrain}(x + 1, z) - \mathbf{terrain}(x, z)]$$

$$\mathbf{n} := \mathbf{n1} + \mathbf{n2} + \mathbf{n3} + \mathbf{n4}$$

$$\mathbf{n} := \frac{\mathbf{n}}{\|\mathbf{n}\|}$$



Obrázek 1: výpočet normály vrcholu

2.3 Detekce kolizí při pohybu po terénu

Pro detekci kolizí slouží metoda *float getY(x, z)*, která vypočte y-souřadnici nad pozicí (x, z) v rovině.

2.3.1 getY

Body x a z se zaokrouhlí dolů, tedy získáme souřadnice nejbližšího rohu čtverce směrem doleva dolů, tzn. bodu **a**.

Jestliže víme, ve kterém čtverci sítě terénu se nacházíme, musíme ještě rozhodnout ze dvou trojúhelníků tohoto čtverce. Skalární součin vektoru s rovinnou normálou vektoru **b – c** určuje v rovině XZ polorovinu, podle znaménka výsledku, čímž zjistíme onen trojúhelník.

$$(\mathbf{c} - \mathbf{b}) \cdot \mathbf{n} = 0$$

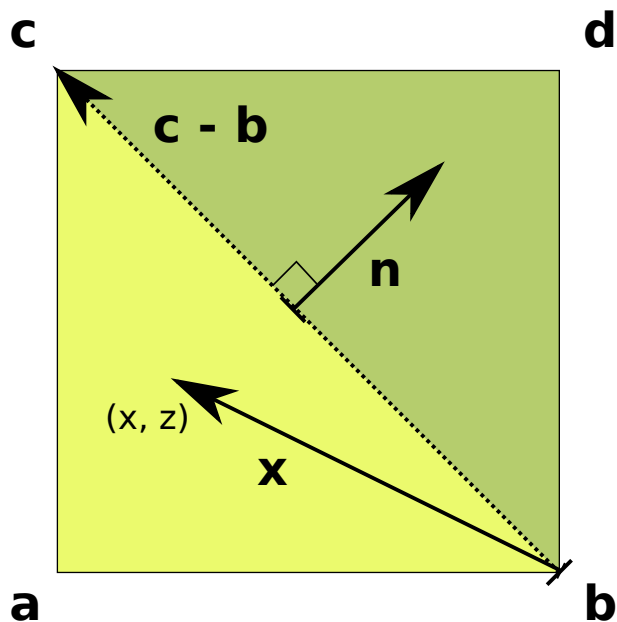
$$\text{sign}(\mathbf{n} \cdot \mathbf{x}) = \text{polorovina}$$

2.3.2 Rychlostní vektor

Poté je spočten směrový vektor v rovině XZ podle momentálně stisknutých kláves WSAD. Výšková složka tohoto vektoru je dopočtena jako rozdíl výšky v terénu nad současnou pozicí a výškou nad novou pozicí určenou směrovým vektorem v rovině. Vektor je pak normalizován a zvětšen rychlostním koeficientem, který zaručuje, že rychlost bude konstantně 3m/s nehlédě na snímkové frekvenci.

Tento rychlostní vektor je pak přičten k současné pozici.

Problém však nastává s výškou pozorovatele nad terénem, která je pevně dána na 1.85m. Použitím relativního vektoru rychlosti dochází ke kumulaci zaokrouhlovacích chyb a po chvíli chození po terénu je výška očí zcela jiná, než 1.85m. Proto je v každém snímku provedena korekce, která zaručuje, že oči budou neustále stejně vysoko.



Obrázek 2: Rozdělení čtverce na poloviny

2.4 Plynulá intenzita světla

Máme-li den, který trvá od času 0 (00:00) do 1 (24:00), očekáváme v čase 0.5 nejvyšší intenzitu světla. Použitá funkce určující intenzitu slunce je Gaussova křivka centrovaná do bodu 0.5.

$$Intenzita = e^{-(C(t-\frac{1}{2}))^2}$$

kde C je magická konstanta, která byla nalezena po chvíli testování tak, aby byl výsledek co nejblíží skutečnosti.

Získaná intenzita se používá přímo jako intenzita hlavního zdroje světla - slunce a jako barva atmosféry.

3 Přeložení a spuštění

Aplikace byla vyvíjena pod GNU/Linux, přiložené knihovny a instrukce pro spuštění aplikace jsou pro Windows.

V kořenovém adresáři je přiložen spustitelný soubor, vytvořený pomocí utility **jarsplice**.

3.1 Z příkazové řádky

spuštění přiloženého archivu:

```
$ java -cp lib/jars/lwjgl.jar:lib/jars/lwjgl_util.jar -jar
bin/ZPG_A10B0632P.jar
```

3.1.1 Přeložení ze zdrojového kódu přímo

přeložení:

```
$ javac -cp .;lib\jars\lwjgl.jar:lib\jars\lwjgl_util.jar; src\Main.java  
src\Terrain.java
```

spuštění:

```
$ java -cp .;lib\jars\lwjgl.jar;lib\jars\lwjgl_util.jar; Main
```

3.2 Použitím Ant

Pokud je nainstalovaný Apache Ant, pak pro přeložení a spuštění stačí použít příkaz z adresáře obsahující skript *build.xml*.

```
$ ant
```

nebo

```
$ ant run
```

3.3 V Eclipse

Získání a integrace **LWJGL** pod eclipse je popsána ve videu [2], ale není třeba. Práce je odevzdána přímo jako projekt eclipse, tedy stačí vytvořit projekt Z

4 Ovládání hry

Kamerou pohneme standardně klávesami WSAD a otočíme pohybem myši. Invertování vertikálního pohledu provedeme pomocí klávesy U a přepnutí mezi drátěným a plným modelem pomocí klávesy C. Hru ukončíme stiskem klávesy ESC.

Reference

- [1] *Bill Jacobs*
OpenGL tutorial 2007 - 2012
<http://www.videotutorialsrock.com/>
- [2] *TheCodingUniverse*
#1 LWJGL Workspace - LWJGL Tutorials
2011
<http://youtu.be/0v56I5UWrYY>
- [3] *Wikipedia contributors*
“Triangle strip,” Wikipedia, The Free Encyclopedia
(accessed October 27, 2012)
http://en.wikipedia.org/wiki/Triangle_strip