ZÁPADOČESKÁ
UNIVERZITA
V PLZNI

Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky
Centrum počítačové grafiky a vizualizace dat
Základy počítačové grafiky KIV/ZPG

Průchod terénem (Passage of terrain)

Zadání:

Zadáním semestrální práce je vytvoření aplikace pro průchod terénem na smyšlené "planetě". Aplikace bude kompletně provedena v prostředí .NET s využitím rozhraní OpenGL.
Terén bude zadaný polem alespoň 128x128 výšek a vykreslený pomocí trojúhelníkové sítě. Výšková mapa terénu bude načtena z binárního souboru, jedna výška je zde reprezentována jedním bytem, výšky jsou uloženy v řádcích od západu k východu, řádky jsou uloženy od severu k jihu.
Průchod pozorovatele terénem je ve výšce očí (1,85 m), výšku terénu mezi body zadanými výškovou mapou je třeba interpolovat. Pozorovatel nesmí vylézt z konce výškové mapy a za žádných okolností nesmí vidět nebo vstoupit do terénu. Pozorovatel se pohybuje konstantní rychlostí (3 m/s). Pozorovatel se může pohybovat pomocí klávesnice a rozhlížet se pomocí myši (nesmí se však přetočit při pohybech nahoru/dolů, rozsah pouze (-90° ; +90°)). Výchozí pozice pozorovatele je uprostřed mapy.
Naše "planeta" je osvícena sluncem, jehož doba oběhu je 2 minuty (tj. "světlo" je jednu minutu, "tma" také jednu minutu). Přechod mezi dnem a nocí musí být plynulý. Během dne se mění intenzita slunečního svitu a uzpůsobuje se i barva atmosféry.

Prohlášení:

Prohlašuji, že všechny použité zdroje jsou řádně citovány a předkládaná práce byla vytvořena zcela mnou, pokud není uvedeno jinak.

e-mail studenta:	jmoulis@students.zcu.cz	Datum:	13. 12. 2011
Kód studenta:	A09B0359P		
Jméno studenta:	Jan Moulis	Hodnocení (počet bodů):	
Datum odevzdání:	13. 12. 2011	Jméno cvičícího:	Ing. Oldřich Petřík
Podpis studenta:		Podpis cvičícího:	

Dne: 2011-12-13

1. Úvod

1.1. Problematika

Při průchodu výškovou mapou je nutné řešit kolize pozorovatele s terénem (pozorovatel musí chodit ve výšce 1,85 m nad terénem a nesmí vstoupit do terénu), kolize s načtenými modely (kolem načtených modelů je prostor, do kterého nemůže pozorovatel vstoupit) a zajistit, aby nemohl pozorovatel opustit terén.

Načítání objektů modelů bylo prováděno z externích souborů Wavefront OBJ pro popis geometrie objektů.

Dalším problémem je rychlost pohybu pozorovatele, která musí být přesně 3 m/s a doba oběhu slunce (2 min = 3° /s, 1 min den, 1 min noc). Oba tyto pohyby musí být závislé na reálném čase a ne na snímkovací frekvenci.

Jako další problematiku bych uvedl změnu intenzity slunečního svitu a změnu barvy atmosféry. Oba tyto problémy jsou závislé na aktuální pozici slunce, respektive na jeho úhlu.

1.2. Řešení

Řešením kolize pozorovatele s terénem je Bilineární interpolace, pomocí které dokážeme určit přibližnou hodnotu výšky terénu v daném bodě. Kolize pozorovatele s vykreslenými objekty je řešena pomocí hranic (obdélníků), do kterých nemůže pozorovatel vstoupit.

Rychlost pohybu pozorovatele a dobu oběhu slunce musíme počítat v závislosti na době, která uběhne mezi jednotlivým rendrováním a dopočítávat z této doby posun pozorovatele a pozici slunce.

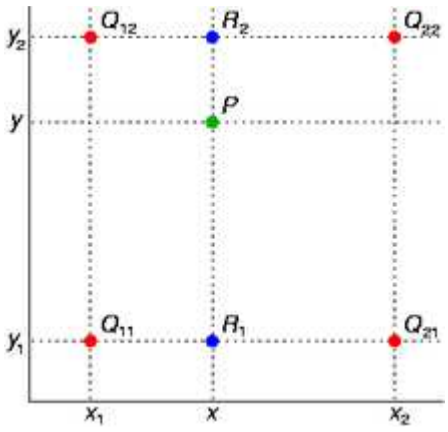
Sluneční svit i změna barvy atmosféry je řešena vesměs stejným způsobem, založeným na úhlu, ve kterém se nachází slunce.

2. Známé metody

2.1. Výpočet výšky a Bilineární interpolace

Pomocí Bilineární interpolace [1] jsme schopni vypočítat přibližnou výšku terénu v kterémkoliv místě a následně upravit pozici pozorovatele (nebo pozici jiných objektů) do správné výšky nad terénem.

Bilineární interpolace je rozšíření lineární interpolace pro interpolaci funkce dvou proměnných na pravidelnou prostorovou mřížku. Klíčová myšlenka je provést lineární interpolaci nejprve v jednom směru a pak i ve druhém směru. Tím dostaneme odhad f(x, y).



Obr. 1: Body pro Bilineární interpolaci

Červené body Q11, Q21, Q12 a Q22 na Obr. 1: Body pro Bilineární interpolaci jsou výškové body načtené z binárního souboru. X a Y je pozice bodu P, ve kterém chceme spočítat přibližnou hodnotu funkce, v našem případě výšku v tomto bodě.

$$f(x,y) \approx \frac{f(Q_{11})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y_2-y) + \frac{f(Q_{21})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y_2-y) + \frac{f(Q_{12})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y-y_1) + \frac{f(Q_{22})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y-y_1).$$

Obr. 2: Vzorec Bilineární interpolace

Podle vzorce na Obr. 2: Vzorec Bilineární interpolace vypočteme přibližnou hodnotu v bodě P a následně k této výšce přičteme výšku postavy pozorovatele 1,85 metru a do této výsledné výšky přesuneme pozorovatele.

Při reálném výpočtu v programu je nejprve nutné zjistit pozici pozorovatele, a jelikož je výškový terén vzorkován po 2 metrech, je nutné přepočítávat tyto souřadnice, abychom byli schopni získat z pole výšek výšky čtyř vrcholů čtverce. Vzorec Bilineární interpolace se zjednoduší, jelikož víme, že rozměr jednoho čtverce je vždy 2x2 metry.

Výpočet výšky v programu:

```
private float getSurfaceHeight(float x, float z)
{
        // zjisteni souradnic leveho dolniho rohu ctverce, ve kterem se nachazi zadany bod, ve vyskove
        // mape
        int x2 = getPositionInField(x);
        int z2 = getPositionInField(z);
        // nacteni vysek v krajnich bodech ctverce
        float xz = groundHeight[z2][x2];
        float x1z = groundHeight[z2][x2 + 1];
        float xz1 = groundHeight[z2 + 1][x2];
        float x1z1 = groundHeight[z2 + 1][x2 + 1];
        // zjisteni presne pozice bodu v danem ctverci
        float xPos = -x - 2 * x2;
        float zPos = -z - 2 * z2;
        // vypocteni vysky v danem bodu pomoci bilinearni interpolace
        float surfaceHeight = xz / 4 * (2 - xPos) * (2 - zPos)
            + x1z / 4 * (xPos) * (2 - zPos)
            + xz1 / 4 * (2 - xPos) * (zPos)
            + x1z1 / 4 * (xPos) * (zPos);

        return surfaceHeight;
}
```

2.2. Rychlost pozorovatele a doba oběhu slunce

Jak již bylo napsáno v úvodu, rychlost pohybu pozorovatele a dobu oběhu slunce, musí být stále stejná a nesmí záviset na snímkovací frekvenci. Proto je nutné zjišťovat si přesný čas před každým rendrováním a následně spočítat dobu od posledního rendrování. Když známe dobu mezi rendrováním, můžeme následně posouvat pozorovatele o přesně danou dráhu, kterou ušel za tuto dobu rychlostí 3 m/s.

Stejný postup je u slunce, jen s tím rozdílem, že rychlost slunce je dána dobou oběhu planety. Tato doba je 2 minuty. Z této doby jsme schopni dopočítat, že za 1 sekundu se slunce posune o 3°. Jeho obvodová rychlost je tedy 3° /s.

2.3. Sluneční svit a změna barvy atmosféry

Pro výpočet slunečního svitu a změny barvy atmosféry je zde použita funkce cosinus, která nám z úhlu, ve kterém se nachází slunce, spočítá hodnotu svitu a hodnotu modré složky atmosféry.

Výpočet slunečního svitu a modré složky atmosféry v programu:

```
light = Math.Abs((float)Math.Cos(sunAngle * rad));      // vypocteni intenzity
amb = 0.5f * light;                                     // hodnota ambientni slozky osvetleni
dif = 1.5f * light;                                     // hodnota difusni slozky osvetleni
LightAmbient1 = new float[] { amb, amb, amb, 1.0f };    // nastaveni ambientního svetla
LightDiffuse1 = new float[] { dif, dif, dif, 1.0f };    // nastaveni difusního svetla
gl.ClearColor(0, 0, light, 0);                          // nastaveni barvy pozadi
```

2.4. Načítání objektů z OBJ souboru

Nejprve bych chtěl lehce popsat formát Wavefront OBJ souboru, následně říci, jak jsem OBJ soubory získal a poté popsat řešení načítání těchto souborů.

2.4.1. Formát Wavefront OBJ soubor

Wavefront OBJ [2] je otevřený souborový formát pro geometrický popis těles vyvinutý institutem Wavefront Technologies. OBJ formát je jednoduchý datový formát reprezentující 3D geometrii objektu – jméno, pozice každého vrcholu, UV texturovací souřadnice pro každý vrchol, normálové vektory a plochy tvořené vrcholy.

Ukázka struktury OBJ souboru pro objekt krychle:

```
# Blender v2.56 (sub 0) OBJ File: 'untitled.blend'
# www.blender.org
mtllib box.mtl
o Cube
v 1.000000 -1.000000 -1.000000
v 1.000000 -1.000000 1.000000
v -1.000000 -1.000000 1.000000
v -1.000000 -1.000000 -1.000000
v 1.000000 1.000000 -1.000000
v 0.999999 1.000000 1.000001
v -1.000000 1.000000 1.000000
v -1.000000 1.000000 -1.000000
vn -0.000000 -1.000000 0.000000
vn 0.000000 1.000000 -0.000000
vn 1.000000 0.000000 0.000000
vn -0.000000 -0.000000 1.000000
vn -1.000000 -0.000000 -0.000000
vn 0.000000 0.000000 -1.000000
g Cube_Cube_Material
usemtl Material
s off
f 1//1 2//1 3//1 4//1
f 5//2 8//2 7//2 6//2
f 1//3 5//3 6//3 2//3
f 2//4 6//4 7//4 3//4
f 3//5 7//5 8//5 4//5
f 5//6 1//6 4//6 8//6
```

Řádky začínající znakem křížku # jsou komentáře.

```
# www.blender.org
```

Řádky začínající řetězcem mtllib udává jméno externího souboru MTL, který obsahuje definici jednoho nebo více pojmenovaných materiálů, které popisují vizuální stránku vykreslovaných mnohoúhelníků. Tento MTL soubor je používán pro všechny následující objekty, dokud se neobjeví tag stejného typu.

```
mtllib box.mtl
```

Řádky začínající písmenem o slouží k pojmenování jednoho objektu uloženého v souboru. Toto jméno platí pro všechny následující vrcholy, texturovací souřadnice, normálové vrcholy a stěny, dokud se neobjeví tag stejného typu.

```
o Cube
```

Řádky začínající písmenem v slouží k popisu souřadnic vrcholu (x y z [w], w je volitelné).

```
v 1.000000 -1.000000 -1.000000
```

Řádky začínající řetězcem vt slouží k popisu texturovacích souřadnic (u v [w], w je volitelné).

```
vt 0.230930 0.909480
```

Řádky začínající řetězcem vn slouží k popisu normálových vektorů (x y z).

```
vn -0.000000 -1.000000 0.000000
```

Řádky začínající písmenem g slouží k pojmenování skupiny mnohoúhelníků. Toto jméno platí pro všechny následující stěny, dokud se neobjeví tag stejného typu.

```
g Cube_Cube_Material
```

Řádky začínající písmenem usemtl slouží k pojmenování materiálu použitého na dané plochy. Jméno materiálu musí odpovídat jménu materiálu v externím MTL souboru, aby ho bylo možné použít. Toto jméno platí pro všechny následující stěny, dokud se neobjeví tag stejného typu.

```
usemtl Material
```

Řádek začínající písmenem f slouží k popsání jedné stěny, tvořené vrcholy. Tyto vrcholy jsou zapsané pomocí indexů. Mimo indexů vrcholů, zde můžou být uvedené i indexy texturovacích souřadnic a indexy normálových vektorů.

```
f 1 2 3 4                       jen indexy vrcholů
f 1/1 2/1 3/1 4/1               indexy vrcholů / texturovací souřadnice
f 1//1 2//1 3//1 4//1           indexy vrcholů // normálové vektory
f 1/1/1 2/2/1 3/3/1 4/4/1       indexy vrcholů / texturovací souřadnice / normálové vrcholy
```

Indexování vrcholů, texturovacích souřadnic a normálových vektorů je možné dvěma způsoby. Prvním z nich je absolutní indexace podle jejich pozice, ve které jsou vypsány. Nebo relativně pomocí záporných indexů a odpočítávání. Ne všechny softwary ale podporují druhou formu zápisu a na druhou stranu, některé využívají pouze této formy a nepoužívají první formu. To vede v některých případech k nekompatibilitě.

Ukázka struktury MTL souboru pro objekt krychle:

```
# Blender MTL File: 'untitled.blend'            hlavička souboru
# Material Count: 1
newmtl Material                                 definice nového materiálu
Ns 96.078431                                    váha použitá pro odrazovou složku barvy
Ka 0.000000 0.000000 0.000000                   ambientní složka barvy
Kd 0.640000 0.640000 0.640000                   difusní složka barvy
Ks 0.500000 0.500000 0.500000                   odrazová složka barvy
Ni 1.000000
d 1.000000                                      průhlednost
illum 2                                         nasvícení modelu
map_Kd sun_20070204_color.jpg                   difusní texturovací mapa
```

2.4.2 Získání OBJ a MTL souboru

Soubory OBJ a MTL jsem exportoval z hotových vymodelovaných objektů v programu Blender a v programu Autodesk 3ds max.

Kvůli relativní indexaci vrcholů, texturovacích souřadnic a normálových vektorů při exportu z programu 3ds max je nutné provést převod na absolutní indexaci, aby bylo zajištěno bezproblémové načtení souboru. Převod je možné udělat například importováním OBJ souboru do programu Blender a následně vyexportovat nový OBJ soubor.

2.4.3 Načítání OBJ a MTL souboru

Načítání souboru je prováděno řádek po řádku. Podle tagů na začátku řádku jsou prováděny ihned operace s daty (jména, názvy souborů, souřadnice, indexy, …).

Pro každý načítaný OBJ soubor je v programu vytvořena jedna instance třídy OBJObject, která má v sobě uloženo pole jednotlivých objektů a pole indexů použitých textur. Jednotlivé objekty jsou reprezentovány třídou SingleObject, která obsahuje informace o vrcholech a stěnách, které jsou důležité pro vykreslování v OpenGL.

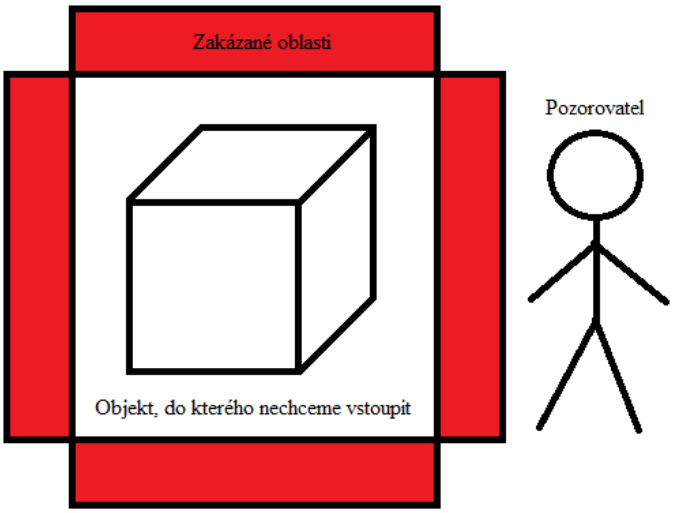
MTL soubor je otevřen, pokud je na něj v OBJ souboru odkázáno. Při změně materiálu objektu, je tento materiál vyhledán v MTL souboru a nastaven objektu. Pokud program MTL soubor nenalezne, nebo v něm daný materiál není, nastaví se objektu náhodný materiál.

Při nastavování materiálu objektu program používá jen difusní složku barvy, pokud je na objekt mapovaná difusní textura, je tato textura použita. Jiné složky barvy a jiné mapovací textury program zanedbává.

Pro správný chod načítání objektu je nutné mít model, který obsahuje stěny tvořené minimálně 2 a maximálně 4 vrcholy. Pokud bude stěna obsahovat více vrcholů, program ji nezobrazí správně.

2.5. Kolize pozorovatele s objekty

Kolize pozorovatele s načtenými objekty je řešena podle následujícího obrázku Obr. 3: Kolize pozorovatele s objektem



Obr. 3: Kolize pozorovatele s objektem

Při každém pohybu pozorovatele se kontroluje, zda není jeho pozice v zakázané oblasti. Pokud ano je jeho pozice nastavena na okraj zakázané oblasti, čímž docílíme toho, že pozorovatel nemůže vkročit dovnitř objektu.

3. Závěr

Základní funkcionalita programu je plně funkční podle rozsahu zadání.

Výpočet výšky pozorovatele je v pořádku a nedochází k poskakování při průchodu terénem.

Řešení kolize pozorovatele s objekty není úplně optimální. Pokud program běží s malou snímkovací frekvencí (FPS < 6), je možné projít zakázanou oblastí a kolidovat s objektem.

Načítání souborů OBJ a MTL funguje v pořádku, ale bylo by možné udělat více vylepšení, které by dovolovali načítat i jiné složky barev a mapovací textury.

Povrch terénu je otexturován texturou pouště, dále je přidán kruh jako slunce a měsíc, aby měl pozorovatel možnost zjistit jaká je doba dne nebo noci a mohl se zorientovat, kde je jaký směr. Slunce i měsíc je taktéž otexturováno. Dále je v programu možnost navrátit se do výchozí pozice (středu mapy), přepínat mezi zobrazením ploch a zobrazením hran a (viz. Ovládání programu) a zapnutí létání pro rychlejší přesuny v prostoru.

Některé problémy jsem řešil a konzultoval s ostatními studenty:
A09B0411P Václav Rajtmajer
A09B0415P Kamil Rendl
A09B0417P Jakub Rinkes

4. Literatura

Bilineární interpolace (online publikace):

[1] Bilineární interpolace. In Wikipedia : the free encyclopedia[online]. St. Petersburg (Florida) : Wikipedia Foundation, 23. 11. 2007, 11:34, last modified on 19. 4. 2011, 20:26 [cit. 2011-12-13]. Dostupné z WWW: http://cs.wikipedia.org/wiki/Biline%C3%A1rn%C3%AD_interpolace

Wavefront OBJ formát (online publikace):

[2] Wavefront .obj file. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 20:32, 8 April 2005, last modified on 13:46, 11 August 2011 [cit. 2011-12-13]. Dostupné z WWW: http://en.wikipedia.org/wiki/Wavefront_.obj_file

5. Ovládání programu

myš	otáčení vlevo/vpravo, rozhlížení nahoru/dolů (pohyb myši vpřed = nahoru)
klávesa "W"	pohyb kupředu ve směru pohledu
klávesa "S"	pohyb vzad vzhledem ke směru pohledu
klávesa "A"	pohyb vlevo vzhledem ke směru pohledu (nikoliv otáčení)
klávesa "D"	pohyb vpravo vzhledem ke směru pohledu (nikoliv otáčení)
klávesa "R"	navrácení se do výchozí pozice
klávesa "U"	zapnutí/vypnutí invertování myši pro pohled nahoru/dolů.
klávesa "SPACE"	zapnutí/vypnutí volného pohybu (létání)
klávesa "+"	pohyb vzhůru (jen při letu)
klávesa "-"	pohyb dolů (jen při letu)
klávesa "F"	přepnutí mezi zobrazením ploch a hran

Chůze rychlostí 3 m/s, let 30 m/s.