Západočeská univerzita
Fakulta Aplikovaných věd
Katedra informatiky a výpočetní techniky

ZPG
semestrální práce

11. prosince 2007

Petr Vondraš
vondras@mesto-domazlice.cz
A06564

I. Zadání

Povinná část

- **Situace -** Aplikace umožňuje procházet terénem na vymyšlené planetě.
- **Implementace**
  - Implementace bude provedena v prostředí .NET s využitím Managed Direct3D.
  - Pro implementaci může být použito rozhraní D3DUT (nutno rozhodnout při prvním odevzdání, není možno se později vrátit do Managed Direct3D). Za implementaci v D3DUT ve Windows je možno získat až 5 bodů, v OS Linux až 10 bodů.
- **povinná část** - nutná podmínka udělení zápočtu terén zadaný polem alespoň 128x128 výšek vykreslený pomocí trojúhelníkové sítě
- **výšková mapa** bude načtena z binárního souboru (ukázka zde), jedna výška je reprezentovaná jedním bytem, výšky jsou uloženy v řádcích od západu k východu, řádky jsou uloženy od severu k jihu
- **interaktivní průchod terénem**
  - pozorovatel chodí po terénu, výška očí pozorovatele viz dále
  - výšku terénu mezi body zadanými výškovou mapou je třeba interpolovat
  - pokud pozorovatel dorazí na konec výškové mapy, nemůže jít dále
  - za žádných okolností nesmí pozorovatel vidět nebo vstoupit "dovnitř" terénu
  - pozorovatel se pohybuje rovnoměrnou rychlostí vzhledem ke skutečnému času (viz níže)i>pozorovatel může pomocí klávesnice chodit vpřed, vzad a dělat úkroky doprava a doleva (nezaměňovat s pohybem na sever, jih, západ a východ!)
  - jednotlivé pohyby vpřed a vzad lze libovolně kombinovat s úkroky do stran, vzniká tak vektor pohybu, který určuje směr, nikoliv rychlost, tzn. hráč se stále pohybuje konstantní rychlostí.
  - pozorovatel se může pomocí myši otáčet, rychlost otáčení je závislá pouze na rychlosti pohybu myši
  - při chůzi dopředu/dozadu/vpravo/vlevo je nutné brát v úvahu natočení pozorovatele vpravo/vlevo
  - natočení pohledu pozorovatele nahoru/dolů nemá na směr chůze vliv
  - při rozhlížení nahoru/dolů se pozorovatel nemůže přetočit, tj. natáčení ve vertikálním směru lze pouze v rozsahu **(−90°; +90°)**, kde 0° je přímo vpřed
  - výchozí pozice pozorovatele je uprostřed mapy
  - veškeré animace a pohyb pozorovatele jsou **závislé na reálném čase, nikoliv na snímkovém kmitočtu**, tj. změna doby nutné na zobrazení jednoho snímku nesmí způsobit změnu rychlosti pohybu či animace v zobrazovaném světě
  - scéna je osvětlena sluncem
    - naše planeta velmi rychle rotuje, jeden astronomický den trvá dvě minuty, přičemž "světlo" je jednu minutu, "tma" také jednu minutu
    - během dne se mění intenzita slunečního svitu, za úsvitu a za soumraku je nejmenší, v poledne největší
    - během noci je terén osvětlený tak, aby bylo něco vidět
    - přechod mezi dnem a nocí musí být plynulý
  - zobrazení počítadla snímků za vteřinu (FPS)
    - výpočet počtu snímků za předchozí vteřinu počínaje aktuálním snímkem, tj. počet předchozích snímků kteréžto se vejdou do okénka o velikosti jedné vteřiny
    - snímková frekvence nesmí klesnout pod 20 fps
    - snímková frekvence je reálné číslo
  - geometrie a údaje >výšková mapa o rozměru 128x128 se zobrazí jako terén 254x254 m, tj. výška terénu je vzorkována po 2 metrech
  - hodnota 0 ve výškové mapě znamená nadmořskou výšku 0, hodnota 255 nadmořskou výšku 25,5 m
  - pozorovatel
    - výška očí: 1,85 m
    - hmotnost: 95 kg
    - průměrná rychlost chůze: 3 m/s
- **ovládání pouze standardní vstupní zařízení :**

vstup	význam
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 'U'	'zapnutí/vypnutí invertace myši pro pohled nahoru/dolů.

Implementované variace

- **grafický vzhled a efekty**
  - **drátěný model**
    Zobrazení drátěného modelu scény s respektováním viditelnosti jednotlivých hran. Čárový mód nesmí ovlivnit zobrazení UI (User Interface) a HUD (Heads Up Display).
  - **model**
    Vytvoření funkce/struktury pro načtení připraveného modelu (X formát) a jeho umístění do terénu. Model musí obsahovat minimálně dva různé materiály.
  - **vegetace**
    Posázení terénu jednoduchou vegetací např. nějakým roštím. Výška, hustota a podobné parametry by se měly v terénu měnit např. podle nadmořské výšky apod.
  - **textura**
    Načtení a aplikace textury s vizuálně nejlepším možným způsobem vzorkování textury dostupným na daném HW. Úloha předpokládá vytvoření autodetekce pro detekováni nejlepšího přístupu jaký je daným HW podporován: nearest-neighbor, linear, anisotropic, nearest-neighbor mip-map, linear mip-map (též trilineární filtrování), atd.
  - **průhlednost**
    Správné zobrazení jednoduchých průhledných objektů (krychlí apod.). Aplikace malířova algoritmu bez sledování částečného překryvu polygonů.
- **hra**
  - **sběr předmětů**
    V terénu jsou rozloženy předměty. Hráč předměty sbírá a počítá se počet sebraných předmětů, který je zobrazen na obrazovce. Předměty poskakují, točí se na místě, popř. provádějí nějaký jiný zajímavý pohyb. Rychlost pohybu předmětů je nezávislá na snímkovém kmitočtu.
  - **teleport**
    V terénu jsou rozmístěny "teleporty" - nějakým způsobem zvýrazněná místa. Teleportace proběhne po vstupu na toto místo někam jinam do terénu. Implementace efektu pro hráče: přechod obrazu do bíla/z bílé do cílové scény.
- **osvětlení**
  - **svítilna I**
    Hráč si v noci může svítit svítilnou. Implementace svítilny bude pomocí vertex/pixel shaderů (předpokládá se využití PixelShaderu 2.0 - HLSL)nebo pomocí standardního kuželového světla s tím, že budou korektně osvětleny i trojúhelníky v hráčově nejbližším okolí.
- **ostatní**
  - **konzole**
    Jednoduchá poloprůhledná konzole obvyklá z počítačových her. Minimálně 3 příkazy s dopadem na scénu. Za vzor je považována konzole ze hry Quake III.

II. Řešení

Povinná část

Program by měl splňovat všechny body povinné části zadání.

Popis řešení pohybu po terénu:

Funkce **pohyb()** podle stisknutých kláves pohybu a směru natočení světa (proměnná yaw) zjisti směr pohybu ve vodorovném směru. Ten poté znormalizuje a vynásobí uplynulým časem od minulého zobrazení a rychlostí pohybu. Poté zkontroluje, zda jsme se nedostali za okraj terénu (kvůli vzhledu 5m od okraje) a případně nás vrátí na okraj. V této nové poloze poté zjistí souřadnice vrcholů trojúhelníka, nad kterým se nacházíme. Z těchto souřadnic vypočteme výšku v novém bodě (pomocí dvou směrových a následně normálového vektoru zjistím rovnici roviny trojúhelníka a poté udělám průnik se svislicí procházející danou polohou) zvednutou o výšku postavy. Rozdílem této výšky a původní výšky získáme třetí (výškovou) souřadnici směru pohybu. Ten opět znormalizujeme, dále vynásobíme hodnotou 1-pohyb.Z (čím prudší kopec tím pomaleji stoupám, resp. rychleji klesám) umocněnou na faktor zpomalení a vynásobíme uplynulým časem a rychlostí pohybu.
Vedlejším efektem tohoto postupu je plynulejší změna svislého směru pohybu ve zlomových místech.

Osvětlení:

Osvětlení terénu je řešeno obíháním slunce kolem terénu. Osvětlení skyboxu analogicky pouze zesvětluje a ztmavuje. Pro osvětlení v noci jsme implementoval baterku ve směru pohledu (implicitně je stále zapnutá - i ve dne - viz ovládání)

Ovládání:

Kromě ovládání dle zadání přidáno:

vstup	význam
klávesa 'B'	zapnutí / vypnutí baterky
klávesa 'Space'	návrat do výchozí pozice
klávesa 'K'	zapnutí / vypnutí konzole
klávesa 'Enter'	potvrzení volby na konzoli
klávesa 'Up' a 'Down'	změna volby na konzoli

Implementované variace

Drátěný model:

Funkce **RenderSitTrojuhelniku()** pokud je zapnuto vykreslování trojúhelníků (bool proměnná *sitTrojuhelniku*) nastaví parametry vykreslování (Material, FillMode, Color) na vykreslení sítě, tu poté vykreslí a vrátí původní parametry.

Model:

Jako X formát načítám skybox, objekty a teleporty. Načtení např. modelu skyboxu zajišťuje funkce **VytvoritSkybox(),** která do proměnné *meshSkyboxu* uloží objekt skyboxu, do pole *materialySkyboxu* materiály a do pole *texturySkyboxu* textury jednotlivých ploch. O vykreslení se pak stará vlastní funkce **Render()** která v cyklu (for (int i = 0; i < materialySkyboxu.Length; i++)) vykreslí všechny Subsety.

Vegetace:
Do terénu jsou náhodně umístěny stromy jako bilboardy. O vytvoření stromů se stará funkce **VytvoritStromy()**, která do ArrayListu stromy postupně ukládá náhodné pozice stromů (do poctu podle konstanty STROMU). Tyto pozice kontroluje, zda nejsou příliš blízko již vytvořeným stromům, nebo objektům, nebo teleportům, nebo příliš vysoko (stromy rostou pouze do 5m). Pokud je pozice v kolizi, vygeneruje náhodně novou, dokud daná pozice nevyhovuje. Dále tato funkce naplní vertexBuffer stromu (Vytvoří obdélník pro texturu). O vykreslení stromů se stará funkce **RenderStromu()**, která stromy nejprve seřadí od nejvzdálenějšího k nejbližšímu, podle polohy pozorovatele, k čemuž využívá třídu RazeniStromu : System.Collections.IComparer. Poté tato funkce natočí obdélník jednotlivých stromů kolmo k pozorovateli. A dále detekuje kolize s jednotlivými stromy. Pokud se dostanu blíže než 1m ke kmeni, jsme vrácen na vzdálenost 1m ve směru od stromu přes pozici pozorovatele. Výsledkem je posun po kruhu (o poloměru 1m) kolem kmene.

Textura:
Pro texturu terénu používám jeden obrázek, a funkce **VytvoritVertexBuffer(object)** poté podle výšky daného vrcholu vypočítá souřadnice (Tu a Tv) vrcholu v dané textuře. K nastavení parametrů vzorkování textur slouží funkce **NastaveniTextur()**, která ověří dostupnost metody vzorkování na daném HW a nastaví vzorkování na první dostupnou v pořadí MipMapLinear, MipMapPoint, Anisotropic, Linear, Point.

Sběr předmětů:
Pro předměty jsem vytvořil model ve formátu .X (jedná se o 3D text "ZPG"). Funkce **VytvoritObjekty()** načte meshObjektu a priřadí mu texturu (objekt.dds s nastavenou průhledností - alpha kanálem). Poté do proměnné polohaObjektu = int[OBJEKTU,3], kde OBJEKTU je konstanta počtu objektů, dosadí do první a druhé souřadnice náhodnou polohu (x a y souřadnici na terénu) a do třetí souřadnice, která značí zda je objekt vidět (1) nebo již byl sebrán (0) dosadí 1. Funkce **RenderObjekty()** poté vykresluje dosud nesebrané objekty (otáčející a pohupující se podle strojového času) a kontroluje, zda jsem se k nim dostal dostatečně blízko. Pokud ano seberu objekt, tj. zvýším sebranoObjektu o 1 a nastavím třetí souřadnici daného objektu v PolohaObjektu na 0. Procházením přes objekty potom pozorovatel získává body.

Teleport:
Teleport je opět načten z .X souboru a je tvořen dvěma modrými válcovými podstavci s mezerou mezi nimi. Funkce **VytvoritTeleporty()** tyto soubory načte a dále vytvoří pole teleportů teleporty = int[5,3], kde první dvě složky jsou souřadnice teleportu v terénu a třetí určuje ke kterému teleportu daný teleport teleportuje. O vykreslení se stará funkce **RenderTeleport()**, která dále kontroluje, zda jsem do nějakého teleportu nevkročil, pokud ano dojde k zastavení pohybu, nastavení počátku teleportace a ke spuštění animace teleportace o kterou se stará funkce **Teleportace().** Tato funkce postupně snižuje průhlednost bílé konzole přes celé okno, až po neprůhlednou bílou. Poté provede posun na pozici cílového teleportu (kousek dál ve směru pohledu, abych nespustil hned další teleportaci) a poté naopak zvyšuje průhlednost až po úplné vypnutí bílé konzole.

Svítilna I:
O implementaci svítilny se stará funkce **NastaveniSvetel()**, která nastaví pozici a směr svícení podle naší pozice a směru pohledu. Aby byly správně osvětleny i objekty v blízkosti pozorovatele, je zdroj posunut kousek zpět proti směru svícení.

Konzole:
Ve funkci **InicializaceGrafiky()** dojde k inicializaci Sprite konzoleSprite a k naplnění pole s texturami pro konzoli třemi obrázky (každý s jednou zvýrazněnou volbou). Pokud je konzole zapnuta (stisk klávesy 'k') funkce **RenderKonzole()** ji vykreslí s danou texturou. O přepínání případně potvrzení voleb se stará funkce **StiskKlavesy(object , KeyEventArgs)** ,která poté mění index textury pro konzoli , případně nastaví proměnnou korespondující s danou volbou

III. Závěr

Při vytváření programu jsem postupně objevoval možnosti 3D programování a zjišťoval sílu předností již objevených konstrukcí (Nejprve jsem například počítal normály vrcholů terénu ručně, poté jsem objevil kouzlo meshe a úvodní načítání dat se mnohonásobně zrychlilo). Získal jsem základní vědomosti o této oblasti, ale zároveň jsem zjistil o jak širokou oblast se jedná.

V programu ještě chybí některá ošetření chybových stavů, která jsem z časových důvodů nestihl dodělat. Jejich implementování by bylo obdobou implementace ošetření neexistujícího souboru s výškovou mapou.

IV. Zdroje

DirectX SDK sample browser
http://www.riemers.net/eng/Tutorials/dxcsharp.php
http://blogs.msdn.com/coding4fun/archive/2006/11/02/938703.aspx