OSNOVA PRÁCE

1. **ÚVOD**
   1. **Co je za problém a já je musim resit? Seznameni s tim co budu delat atd.**
   2. **Moje rešení – nastinení ceho se práce týká apod.**
   3. **O čem ta práce je, seznamení, CO BYLO ÚKOLEM**
   4. **Rozvrhnuti prace**
2. **Technologie**
   1. **HTML5**
      1. **Uvodem, co to je apod.**
      2. **Video**
      3. **Canvas**
      4. **svg**
      5. **Další pouzite věci pokud jsou**
   2. **WEBGL:** 
      1. **Uvodem, JS apod**
         1. **jakou má funkci,**
         2. **z čeho vychází,**
         3. **proč vůbec existuje**
         4. **k čemu se nejčastěji pouziva**
      2. **verze WebGL (pouzita)**
      3. **Popis kamery a zobrazení**
         1. **Perspektivní**
         2. **Ortografické apod**
      4. **Trannsformacni matice**
   3. **GLSL**
      1. **Uvodem, k cemu to je**
      2. **Vertex shader**
      3. **Fragment shader**
3. Návrh řešení [SEM PSÁT VÝPOČETY , VZOREČKY A TEORII]
   1. Postup běhu programu, z hlediska navrhu
      1. vypocty geometrie
      2. Vypocet korekce textury
      3. Projekce:
         1. Transformacni matice
            1. Model
            2. View
            3. Perspetive
   2. Jak resit metadata?
   3. Jak resit kompas? Navrh
   4. **JAK RESIT BLENDING**???
      1. **Vertex a fragment shader**
4. Implementace
   1. Uvodem: postup behu programu, Z HLEDISKA TOKU DAT
   2. Tvorba shaderů
   3. Tvorba geometrie
      1. Transformace sférických souřadnic na kartézské
   4. Buffery
      1. Vertex => [x,y,z]
      2. Normal => [x,y,z] (data bez radiusu)
      3. Texture data => [u,v]
      4. Index data => [w,e,r,t,h,m]
   5. Nacteni textur videa
   6. Transformační matice a operace
   7. Ovládání prohlížeče
   8. Kompas
   9. Jak bylo vyřešeno bezešvé prohlížení
      1. Gauss / pyramida, blending obecně
5. Testování
   1. Windows X Linux
   2. Chrome, fireofx, opera, IE apod.
   3. Stěžejní věci skrze kompatibilitu, výsledky
6. Závěr
   1. Zhodnocení dosažených vysledku
7. Literatura
8. Přílohy

Geometrie – Vlastní práce

Aby panoramatický prohlížeč jednotlivých fotek či videí dokázal vyobrazit naše data, je nejprve potřeba vytvořit geometrii. Jedná se o pole souřadnic formátu [x1,y1,z1 , x2,y2,z2,…], které utváří jednotlivé body v prostoru, body pak mezi sebou tvoří prostor, na který budou data mapovaná. Ve Webgl umíme vykreslit jednotlivé body, čáry – tedy spojnice jednotlivých bodů, nebo trojúhelníky. V našem případě bude potřeba vykreslit texturu jako plochu, na kterou se bude vše mapovat. Pro ten to případ se ve webgl používá vykreslení pomocí trojúhelníku, jako základní jednotkou plochy. Způsob vykreslení našich bodu určíme tedy řádkem: gl.drawElements(gl.TRINAGLES, 6, gl.UNSIGNED\_BYTE, 0), kde tedy říkáme že chceme vykreslovat plochu pomocí trojúhelníků, víz výše. (gl.TRINAGLES), druhým parametrem říkáme, kolik hodnot z index bufferu budeme potřebovat vykreslit, třetím parametrem oznamujeme velikost jednoho indexu a posledním parametrem říkáme, odkud začíná program číst naše body. Mohli bychom naše body vykreslovat ve webgl i pomoci příkazu: gl.drawArrays(), ale došlo by pak k situaci, že by se nám již vykreslené body zbytečně vykreslovali znovu. Abychom předešli duplikaci při vykreslování, bude nutné si vytvořit pole indexů, které nám budou ukazovat na jednotlivé vrcholy geometrie. Zde se již mohou indexy opakovat.

…

Geometrie prohlížeče je nastavena na 60 vertikálních spojnic severního polu s a jižním polem. Horizontálních čar (rovnoběžek) budeme potřebovat o něco méně, takže je nastavíme na hodnotu 50. Čím více rovnoběžek a poledníků vytvořili, tím by byla hustota bodů v polích hustší, a mapování textury tedy jemnější. Tato vlastnost se nám bude hodit např. v situaci, kdy bychom měli velký geometrický objekt se spoustou miniaturních sekvenci v geometrii. Větší mezery jsou nadruhou stranu rychlejší na vykreslování a to se stejným výsledkem (v našem konkrétním případě). Geometrii nastavíme výpočtem souřadnic X, Y, Z. Jelikož poledník je rozdělen rovnoběžkami, můžeme tedy 180° úhel rozdělit mezi jednotlivé rovnoběžky, tedy PÍ/Počet rovnoběžek = theta, který bude svírat vertikální čára geometrie s osou Z, tudíž můžeme souřadnici X vyjádřit vztahem X= sin(theta) \* cos(phi).

Vypočet bodů



Výpočet souřadnice **X**:

Sinus je tedy protilehlá strana ***K*** ku přeponě r,

Abychom mohli vytvořit naší geometrii, musíme spočítat body [x,y,z] pro všechny poledníky a rovnoběžky. Každá vertikální čára od severního pólu k jižnímu (poledník) se protíná s našimi rovnoběžkami. Jelikož poledník nabývá od severního polu k jížnímu 0° - 180°, tak podělením tohoto úhlu počtem rovnoběžek geometrie nám vzniká rovnoměrné rozložení bodů po celém poledníku. Takto máme vyřešen jeden poledník a průnik rovnoběžek, dále je potřeba tuto operaci provést pro všechny poledníky, aby byl model kompletní. Součet úhlů všech poledníku nabývá hodnot od 0-360°, tudíž šířka jednoho poledníku a tedy úhel mezi dvěma vertikálními čarami od počátku soustavy souřadnic X,Y,Z je roven podělení 360° všemi poledníky. Tím dojde k aproximaci celé koule, na kterou budeme mapovat naše data.

V praxi tedy potřebujeme body X,Y,Z vypočítat. Využijeme k tomu goniometrické funkce a vztahy v pravoúhlém trojúhelníku. Úhel, který svírají jednotlivé vertikální čáry – poledníky mezi sebou spolu s počátkem soustavy souřadnic, označíme jako úhel: , pro osu x tedy bude platit vztah:

Protože přilehlá strana k úhlu jeprávě osa x, a dle goniometrického vzorce přilehlá/přeponě bude naše přepona označena jako K, tudíž máme vypočten vztah pro osu x v systém XY, neboli ve 2D rovině. Abychom byli shopní souřadnici X spočítat v prostoru, bude nutné vypočítat vztah, rpo přeponu K:

Úhel, který svírá osa Z s jetnolivými rovnoběžkami geometrie označíme jako úhel: . Pro přeponu **K** v rovině **XY** platí goniometrický vztah: protilehlá odvěsna/přepona, odvěsna je v našem případě tedy **K** a přepona **r**, výsledný vztah dosadíme do vzorečku pro výpočet souřadnice **X** a vznikne nám vztah pro výpočet souřadnice X v rovině XYZ:

Obdobným případem budeme postupovat i u osy Y:

Protilehlá strana y k úhlu podělena přeponou K dostáváme vztah pro výpočet souřadnice y v rovině XY, dosazením za K dostáváme výsledný vztah pro výpočet souřadnice Y.

Menší odlišnost tu bude v případě poslední souřadnice Z, kde nám stačí pouze vyjádřit vztah pro úhel: , protože Z nám zde plní funkcí jakési „výšky“



# UVODEM