Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Princípy počítačovej grafiky a spracovania obrazu

Semestrálny projekt

Vypracovali: Samuel Bubán, Matej Delinčák

Cvičiaci: Ing. Lukáš Hudec, PhD.

Cvičenie: Utorok 16:00

Ak. Rok: 2021/22

Dátové štruktúry

Ako zaujímavú dátovú štruktúru sme použili height mapu pri riešení kolízií ponorky s terénom. Na celý terén sme vygenerovali bmp obrázok, v ktorom máme odtiene sivej farby. Čím je farba tmavšia, tým nižší bod to predstavuje v scéne. Po takejto vygenerovanej mape, sme ju museli namapovať na súradnice v scéne a následne aj výšku jednotlivých bodov. Následne v programe ju teda môžeme využiť pri generovaní objektov alebo už pri spomínanej kolízií. Jednoducho sa pozrieme na súradnice x, z a pomocou prístupu do poľa zistíme súradnicu y (naša scéna má y súradnicu určenú ako výšku). Ide o O(1) prístup, čiže to je rýchle.

V zadaní bolo písane použitie troch zdrojov svetla. V našom projekte používame omnoho viacej zdrojov a tie sú uložené v poli v objekte scény. Tie sa potom vkladajú do shadera v každom objekte. Každé svetlo ma náhodnú silu, dosah a farbu. Osvecuju len svoje okolia a teda neinteragujú s celou scénou.

Výpis zaujímavých algoritmov

Ako jeden z najzaujímavejších algoritmov, ktoré boli použité, sme vybrali výpočet bezierovej krivky. Pri generovaní rýb sme každej priradili kontrolné body tejto krivky. Náš algoritmus nefunguje len na štyroch bodoch ako sme mali robiť na cvičeniach, ale na ľubovoľný počet. Využívame totiž rekurziu pri jej výpočte.

```
glm::vec3 Fish::bezierRec(std::vector<glm::vec3> points, float t) {
      if (points.size() == 2) {
            return getPoint(points.at(0), points.at(1), t);
      }
      std::vector<glm::vec3> new_points;
      for (int i = 0; i < points.size() - 1; i++) {
            new_points.emplace_back(getPoint(points.at(i), points.at(i + 1),
      t));
      }
      return bezierRec(new_points, t);
}
```

Zdrojový kód ku bezierovej krivke

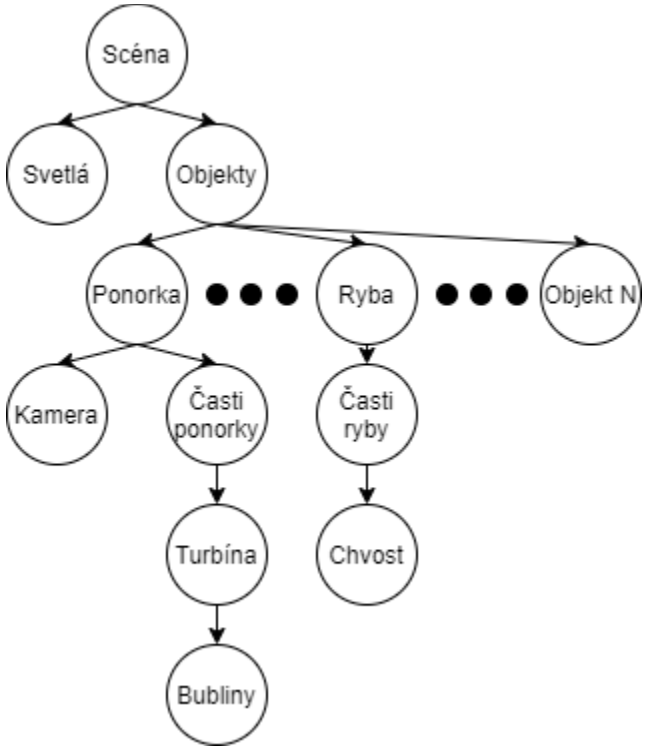
Ako sme spomínali vyššie vygenerovali sme si height mapu a tú používame pri generovaní objektov. Využívame to pri generovaní svetiel (aj v jaskyni aj mimo nej), sopiek (len v jaskyni) a sasaniek. Náhodne si teda zvolíme dve súradnice v scéne x, z a po transformácií si vytiahneme súradnicu y z height mapy. Máme už teda všetky tri súradnice a vložíme objekt do scény.

```
float Scene::getHeight(float x, float z) {
    int j = (x * 10 + 539.7) *  3.7947001;
    int i = (-1* z * 10 + 614.7) * 3.7947001;
    float color = heightFramebuffer[3 * (i * imgWidth + j)] +
                  heightFramebuffer[3 * (i * imgWidth + j) + 1] +
                  heightFramebuffer[3 * (i * imgWidth + j) + 2];
    return ((color) * 0.3310595- 75.6565)/10;
}
```

Zdrojový kód ku získavaniu výšky na základe súradníc

Opis scén pomocou grafu scény, opis priestorových vzťahov

V našej scéne je väčšina objektov na rovnakej úrovni čo sa týka vzťahov medzi sebou. Preto nebudeme rozpisovať celú scénu pomocou grafu len vybrané objekty. Ako prvú si zobrazíme hierarchiu ponorky, scény a kamery.

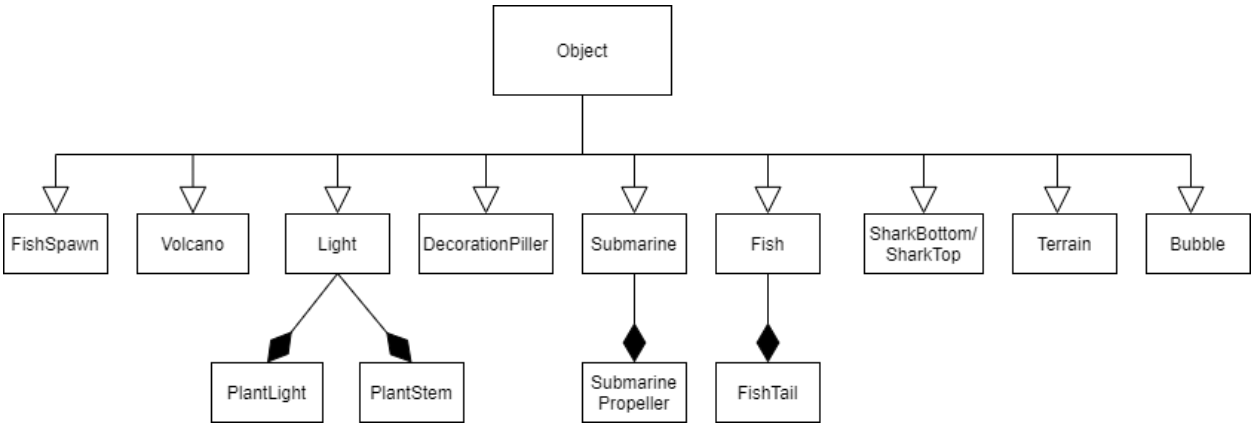


Scéna teda obsahuje svetlá a objekty. Medzi objekty spadá aj objekt ponorky, na ktorú sú naviazané objekty a kamera. V našom deme bude vidieť, že kamera sa drží za ponorkou v určitej vzdialenosti. Posúva sa podľa posunu ponorky a taktiež podľa otočenia ponorky. Translačnú maticu ponorky dostanú aj všetky časti ponorky, ktoré sú jej súčasťou. V našom prípade len jedna turbína, ale je to pripravené aj na viacej častí. Tá sa teda posúva a rotuje na základe ponorky. Pri každom pohybe turbíny je šanca, že sa do priestoru vygenerujú bublinky, ktoré sú ale nie pod správou scény ale samotnej turbíny. Tie tiež dostanú translačnú maticu. A bublinky následne stúpajú podľa aktuálnych vodných prúdov. Medzi priestorové vlastnosti ponorky by sme ešte vložili aj kolízie medzi terénom a dekoračnými stĺpmi. Ponorku scéna nepustí príliš vysoko, ani do strán, či pod terén. Taktiež máme nastavené kolízie so stĺpmi tak, aby cez ne ponorka neprešla, ale zastavila sa pred nimi.

Ako ďalšiu hierarchiu v našej scéne by sme poukázali na hierarchiu v rámci ryby, kde každá ryba vlastní ešte chvost, ktorej posiela svoje otočenie a pozíciu. Chvost tieto informácie spracuje a vybuduje sa model matrix. Rotácia v jednom smere je ale zmenená, lebo sme spravili to, aby sa chvost ešte otáčal do bokov. Taktiež máme pripravenú rybu tak, aby sme jej mohli dať aj viacero objektov, nie len chvost.

Opis objektov v scéne pomocou diagramu tried

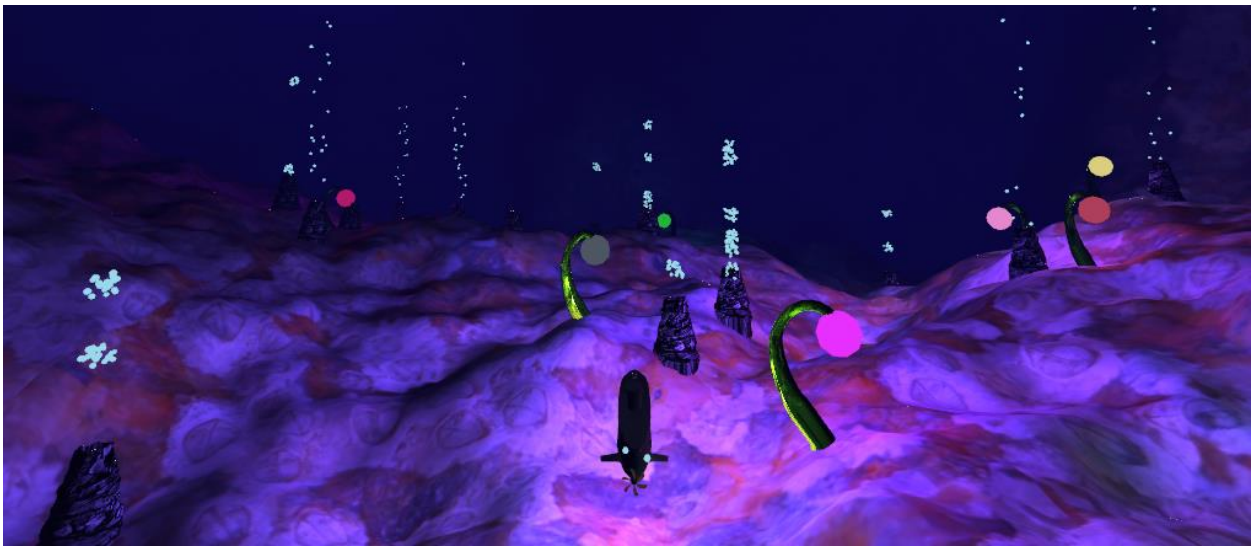
V našom projekte nemáme nejaké zložité diagramy tried. V podstate všetky objekty dedia spoločne vlastnosti od objektu *Object*, ktorý má nejaké základné vlastnosti a metódy. Ponorka sa skladá z hlavnej časti, z turbíny a generovaných bubliniek. Ryba sa skladá z tela a chvostu. Svetlo je rozdelené na časť stonky a časť samotného svetla, ktoré mení farbu. Každý z objektov má vlastný *.obj* súbor s modelom a textúru ako *.bmp*.



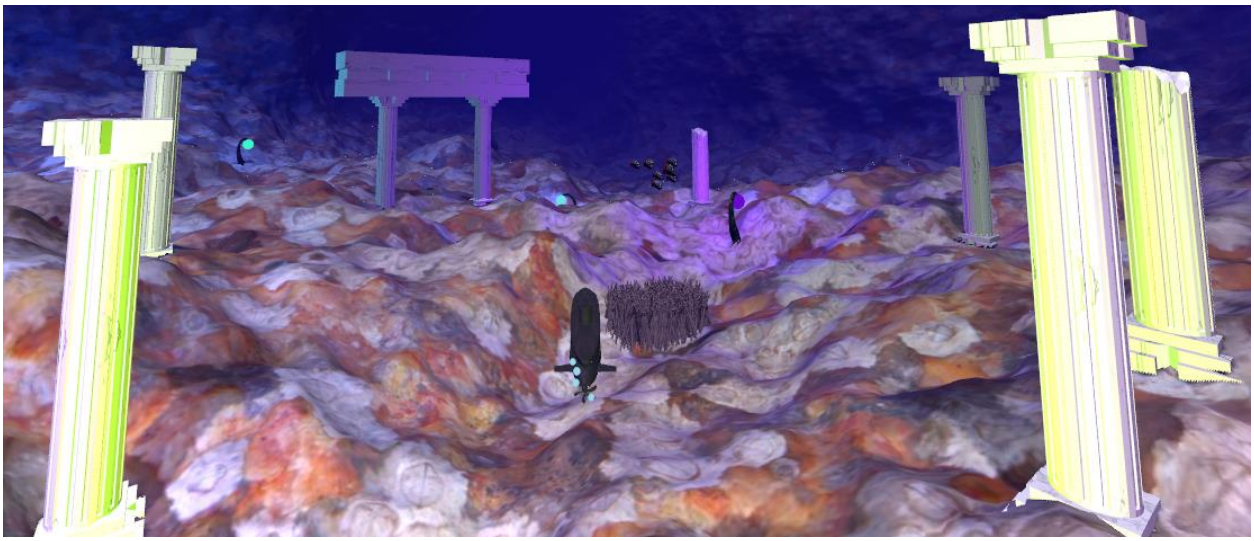
Ďalej tu máme triedy scény a projektovej scény, v ktorej sú základne metódy na vytváranie a renderovanie objektov, získavanie informácie z height mapy a samozrejme aj jej načítanie a ovládanie typu kamery.

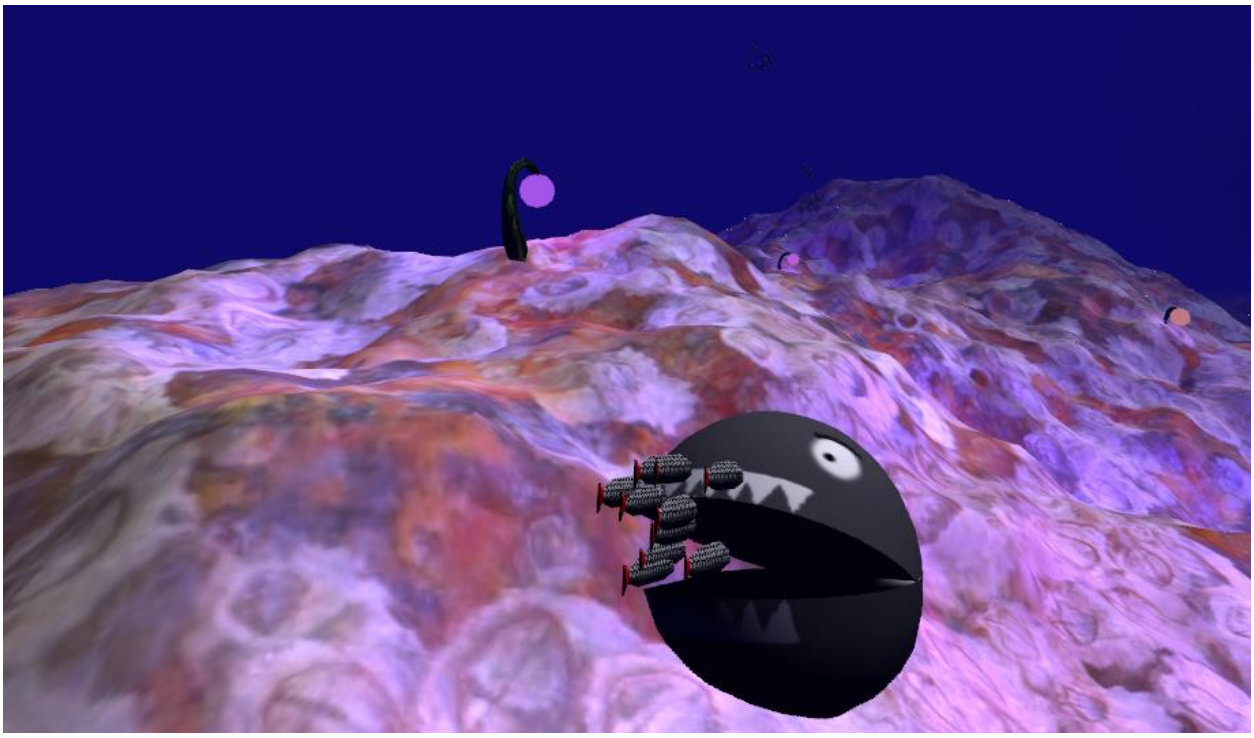
Storyboard

Prvá scéna je v rámci jaskyne – vnútro. Môžeme tu vidieť malé sopky, ktoré chrlia bublinky v dvoch formátoch. Po jednej bublinky neustále a druhý formát ako skupinu bubliniek vychrlenú raz za čas. Pomocou šípiek vieme meniť vodné prúdy a teda bublinky budú unášane podľa týchto prúdov. Po celej scéne sme umiestnili aj rastliny, ktoré vytvárajú osvetlenie pre scénu. Samotná ponorka po osvetlenie tiež.

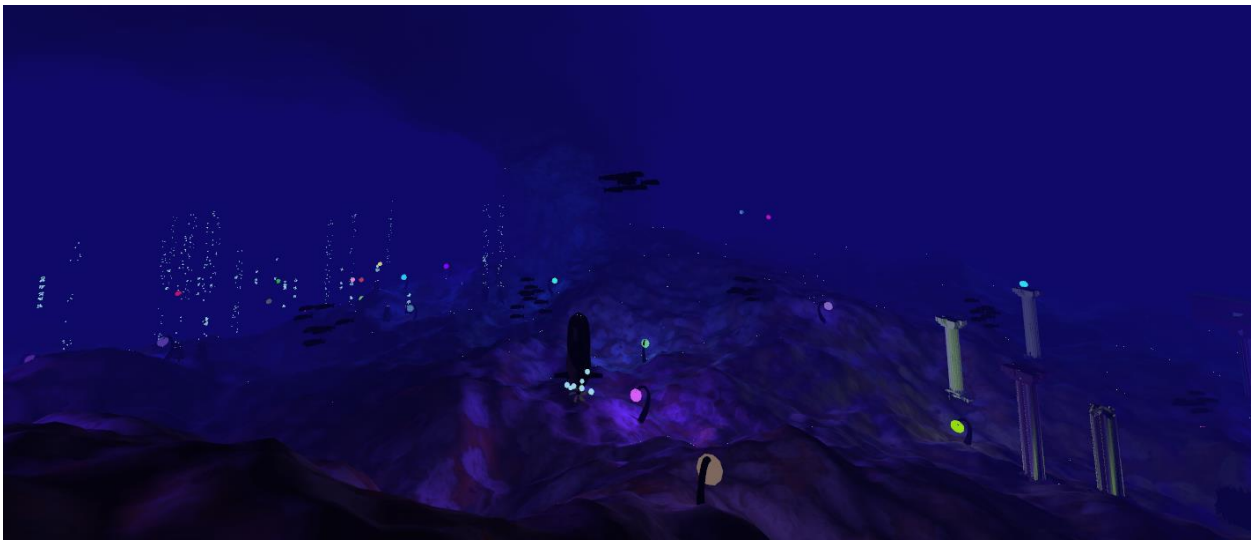


Po vyjdení z jaskyne môžeme po ľavom boku uvidieť skupinu okrasných stĺpov, ktoré sme nahradili za loď, ktorú sme mali v návrhu. Na týchto stĺpoch je vidno každú zložku phongovho osvetľovacieho modelu. Zároveň na tieto stĺpy sú umiestnené kolízie, takže ponorka cez ne nemôže prejsť. Kolízie sú nastavené aj na celý terén, kde máme urobenú aj odozvu. Ak pôjde ponorka dopredu a zároveň aj dole, kolízia túto ponorku nezastaví, ale bude kĺzať po teréne. V strede tejto snímky je vidieť sasanku, v ktorej sa vytvárajú ryby, ktoré putujú ďalej po scéne. Takýchto sasaniek máme viacej.





Na tomto snímku vidíme miesto – žraloka, kde ryby zanikajú. Ten sa pravidelne otvára a zatvára. Pravidelnosť pri spustení programu uvidíme aj v pohybe chvostu rýb. Ktorý sa otáča tiež pravidelne a budí dojem, že ryby naozaj plávajú. V pozadí je vidno ďalšie skupinky rýb.



Ako možnosť pre používateľa sme pridali vypnutie denného svetla, ktoré môžeme rátať aj ako ďalšiu scénu v našom projekte. Pri zapnutí a vypnutí globálneho svetla uvidíme aj tiene, ktoré vrhá terén. Poslednú vec, ktorú chceme spomenúť je, že používateľ sa vie hýbať svetom pomocou ponorky. Ktorej sa rotuje turbína a vytvára pár bubliniek pri pohybe ponorky. Napredku má ponorka umiestnené svetlo, ktorým dokáže svietiť na scenériu.

Ovládanie ponorky a programu

Ovládanie ponorky:

- W — pohyb dopredu (každé stlačenie zvýší rýchlosť vpred)
- S — pohyb vzad (každé stlačenie zvýší rýchlosť vzad, vieme aj cúvať)
- A — zrotovanie ponorky vľavo
- D — zrotovanie ponorky vpravo
- SPACE — zvýšenie ponorky
- LEFT SHIFT — zníženie ponorky
- C — zastavenie ponorky na mieste (rýchlosť na 0)

Ovládanie svetla:

- N — zapnutie/vypnutie globálneho svetla (deň a noc)
- V — zmena farby ponorkového svetla
- F — zapnutie/vypnutie ponorkového svetla

Ovládanie prúdov vody:

- LEFT ARROW — zvýšenie vodného prúdu v osi x
- RIGHT ARROW — zníženie vodného prúdu v osi x
- UP ARROW — zníženie vodného prúdu v osi z
- DOWN ARROW — zvýšenie vodného prúdu v osi z

Ovladanie scény:

- R — restovanie na začiatok
- 1 — prichytenie kamery na ponorku + umožnenie pohybu ponorky
- 3 — statická kamera s pohľadom na dekoračné stĺpy
- 4 — statická kamera s pohľadom na žraloka
- 5 — statická kamera s pohľadom na vnútro jaskyne
- 6,7 — dynamická kamera, ktorá pomaly prechádza cez scénu