VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

TVORBA ŘEZŮ Z 3D MODELŮ PRO VYSTŘIHOVÁNKY "SLICEFORMS"

CREATION OF SLICES OF 3D MODELS FOR "SLICEFORMS"

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MICHAL ONDREJÓ**
AUTHOR

VEDOUCÍ PRÁCE — **prof. Ing. ADAM HEROUT, Ph.D.**
SUPERVISOR

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Ondrejó Michal**
Obor: Informační technologie
Téma: **Tvorba řezů z 3D modelů pro vystřihovánky "sliceforms"**
Creation of Slices of 3D Models for "Sliceforms"
Kategorie: Počítačová grafika

Pokyny:

1. Prostudujte a popište vystřihovánku "sliceforms", včetně relevantní matematiky a geometrie pro tvorbu těchto vystřihovánek.
2. Vyberte a popište vhodný nástroj pro návrh a vytvoření vystřihovánek "sliceforms".
3. Navrhněte funkčnost nástroje pro tvorbu vytsřihovánek "sliceforms".
4. Implementujte navržený nástroj, testujte ho na uživatelích a zdokonalujte jej.
5. Vytvořte dostatečnou sadu demonstračních příkladů funkčnosti vytvořeného nástroje a zhodnoťte jeho vlastnosti.
6. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3, značné rozpracování bodu 4.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Herout Adam, prof. Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Cieľom tejto práce bolo navrhnúť a implementovať aplikáciu na tvorbu vystrihovačiek známych ako Sliceforms. Práca sa zaoberá tvorbou prierezov z polygonálneho modelu. Vytvorené riešenie poskytuje možnosť vizualizovať prierezy v 3D modele a umožňuje exportovať prierezy na tlač.

Abstract

The aim of this work was to design and implement an application for the creation of paper models known as Sliceforms. The thesis deals with the creation of cross-sections from a polygonal model. The created solution provides the abilty to visualize the cross-sections in 3D models and allows you to export them to be printed.

Kľúčové slová

Vystrihovačka, Prierezové formy, Geometrické telesá v počítačovej grafike, OBJ, Triangulacia, Triangulator, Metoda rezania uší, Unity, Zárezy

Keywords

Sliceforms, Cross-sectional forms, Geometry objects in computer graphics, OBJ, Triangulation, Triangulator, Ear clipping method, Unity, Notches

Citácia

ONDREJÓ, Michal. *Tvorba řezů z 3D modelů pro vystřihovánky "sliceforms"*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Herout Adam.

Tvorba řezů z 3D modelů pro vystřihovánky "sliceforms"

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána prof. Ing. Adama Herouta Ph.D. Dalšie informácie mi poskytli vo firme Corinth. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Michal Ondrejó
16. mája 2017

Poďakovanie

Týmto by som sa chcel poďakovať vedúcemu práce prof. Ing. Adamovi Heroutovi Ph.D. za odbornú pomoc a nápady pri tvorbe tejto práce.

Obsah

Kapitola 1

Úvod

V dnešnom svete je už možné previesť počítačový model do reálneho sveta vďaka 3D tlači. Táto tlač je ale pre mnohých stále príliš drahá. Modely Sliceforms sú tvorené len z papiera, ktorý nie je taký drahý a pri skladaní jednotlivých plátkov poskytuje tvorcom potešenie a následne hrdosť na poskladané objekty.

Zadaním tejto práce bolo vytvoriť aplikáciu na tvorbu skladačiek známych ako Sliceforms. Táto aplikácia je vytvorená pomocou herného jadra Unity.

Druhá kapitola sa zaoberá históriou modelov Sliceforms a ich skladaním. Tretia kapitola sa zaoberá matematickými a technickými prostriedkami. Kapitola sa začína popisom reprezentácie geometrických telies používaných v počítačovej grafike. Nasledujú geometrické operácie, ktoré sú potrebné na tvorbu prierezov z polygonálneho modela. Ďalej sa kapitola zaoberá herným jadrom Unity, pomocou ktorého je vytvorená výsledná aplikácia. V závere tretej kapitoly je popísaný súborový formát OBJ, ktorý je v vo výslednej aplikácii spracovávaný pri načítavaní polygonálneho modelu. Štvrtá kapitola popisuje postup tvorby plátkov. Kapitola sa začína postupom tvorby prierezov a triangulóciou vzniknutých polygónov. Kapitola pokračuje popisom tvorby zárezov do plátkov a končí sa popisom ukladania finálneho produktu do súboru. Piata kapitola detailne popisuje zvolené užívateľské rozhranie použité vo výslednej aplikácii a kláves používaných na ovládanie aplikácie. Šiesta kapitola popisuje vyhodnotenie vytvoreného riešenia a obsahuje rady k skladaniu. V poslednej siedmej kapitole je popísané možné pokračovanie projektu a o jeho možnom vylepšení.

Kapitola 2

Motivácia pre riešenú úlohu

V dnešnej dobe je niekoľko spôsobov ako vytvoriť skladačky Sliceforms. Jednou z nich je *Slicer for Fusion 360*[1] od firmy Autodesk, ktorá bola nahradená za ukončenú aplikáciu 123D Make na začiatku roka 2017. Táto aplikácia umožňuje vytvoriť prierezy z vybraného objektu a zobraziť ich v rovine. Zároveň umožňuje aj jednotlivé rezy ľubovoľne posúvať a tým zdokonaliť navrhnutý model.

2.1 Prierezový model Sliceform

> *„Obrázky modelov Sliceform neukážu ich celú krásu. Iba vytváraním a fyzickým zaobchádzaním s modelmi je možné plne oceniť ich skutočné dynamické vlastnosti,“* John Sharp [7].

Trojrozmerné objekty nazývané modely Sliceform majú vytvorenú formu z dvoch paralelne pretínajúcich sa sád plátov. Väčšina modelov Sliceform sa neustále deformuje, a to do dvoch plochých tvarov [7].

Olaus Henrici je matematik, ktorý na konci 19. storočia navrhol techniku modelov Sliceform. Modely, ktoré vytvoril, boli tvorené prierezmi štvorcových plôch [8]. Nemecká firma Martin Schilling predávala modely, ktoré môžeme vidieť na obrázku 2.1. Tieto modely navrhol Alexander von Brill v roku 1874 a boli inšpirované modelmi, ktoré Henrici poslal na stretnutie matematikov v Göttingenu. Tieto lepenkové modely neboli také robustné ako drevené alebo plastové ale boli podstatne lacnejšie [6].

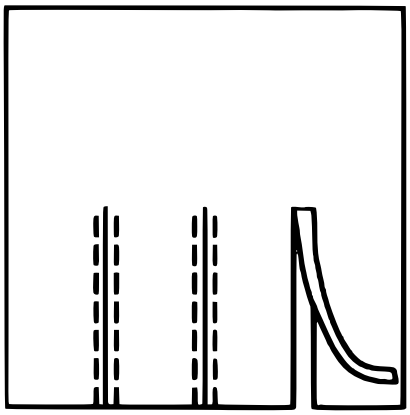
John Sharp neskôr rozšíril modely na rôzne povrchy a polyhedrony a pomenoval ich pod názvom Sliceforms.

[1] https://apps.autodesk.com/FUSION/en/Detail/Index?id=8699194120463301363

Obr. 2.1: Modely predávané firmou Martin Schilling. Zdroj: [2]

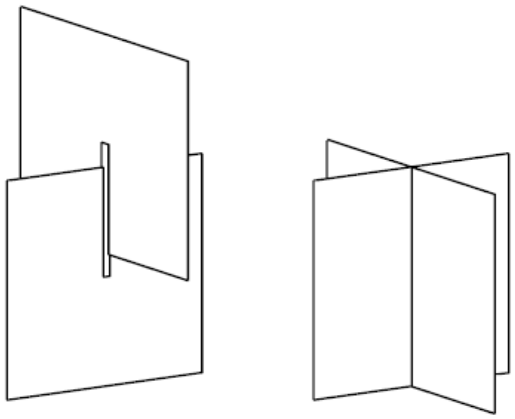
2.1.1 Skladanie modelov Sliceforms

Pred začatím skladania je najprv treba vystrihnúť všetky zárezy. Tieto zárezy je nutné vystrihnúť zastrihnutím pri vyznačenom zárezе z oboch strán. Tým vznikne tenký vlások, ako môžeme vidieť na obrázku 2.2, ktorý sa dá pomocou nechtov vybrať a vytvoriť tak dostatočne široký otvor na vkladanie kolmých plátkov. Tento otvor ale nemôže byť príliš široký, lebo model nebude pevne držať a rozpadne sa [8].



Obr. 2.2: Odporúčané zastrihovanie do prierezov. Zdroj: [8]

Na poskladanie skladačky Sliceforms je potrebné si zoradiť jednotlivé prierezy podľa numerickej postupnosti – najmenšie číslo dole a navyššie hore. Skladanie dvoch kolmých plátov do seba je zobrazené na obrázku 2.3.

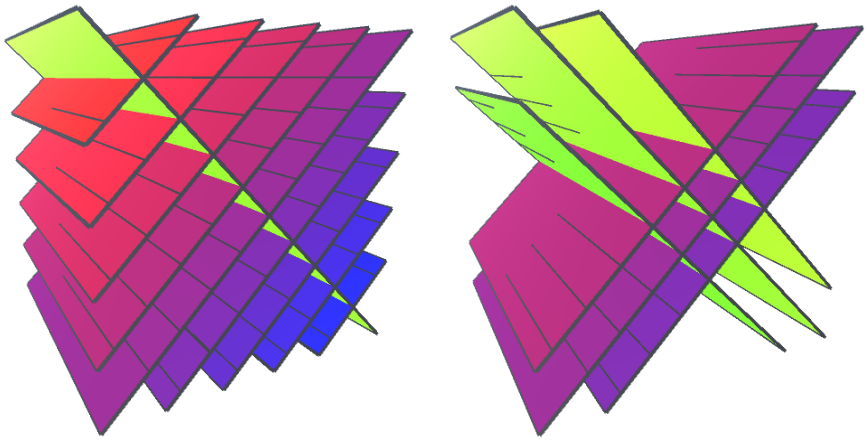


Obr. 2.3: Spojenie dvoch kolmých rezov. Zdroj: [8]

Je viacero možností ako poskladať model Sliceform. Inšpirovať sa je možné niektorov z následujúcich metód [7]:

Metóda 1: Najprv vložíme do stredného plátku X všetky Y-ové plátky a potom prevrátime celý model a postupne od vloženého stredného plátku vkladáme ďalšie X-ové plátky.

Metóda 2: Najprv si spojíme prostredné plátky z osí X a Y a potom postupne striedavo vkladáme ďalšie plátky z jednotlivých osí a prevraciame model. Táto metóda je jednoduchšia, keďže pri vkladaní vnútornejších plátkov je ešte menší počet kolmých plátkov. A pri vkladaní posledných plátkov je už model dostatočne stabilný.



Obr. 2.4: Dve varianty skladania. **Vľavo** je metóda vloženia všetkých plátkov z jednej osi do stredného plátku kolmej osi a **vpravo** je metóda postupného striedavého vkladania plátkov.

Kapitola 3

Použité matematické a technické prostriedky

3.1 Reprezentacia geometrických telies v počítačovej grafike

Veľa počítačových objektov, ktoré sa nachádzajú v trojrozmernom priestore majú charakter telesa. Tieto objekty pripomínajú reálne hmotné telesá, ktoré prirodzene nadobúdajú určitý objem. Každé teleso sa skladá z množiny vnútorných bodov a množiny hraničných bodov. Pri vnútornom bode sa nachádza aspoň jeden vnútorný a prípadne aj hraničný bod. Pri každom hraničnom bode sa nachádza aspoň jeden vnútorný, vonkajší a ďalší hraničný bod. Všetky tieto tri množiny sú navzájom disjunktné. Z uvedeného vyplýva, že sem nepatria objekty ako sú úsečky a krivky v priestore, časti rovín a všeobecné plochy, keďže tieto objekty neobsahujú žiadne vnútorné body [11].

Hlavným stavebným prvkom v počítačovej grafike je trojuholník, a to práve vďaka svojim dobrým vlastnostiam. Je vždy konvexný a vždy leží celý v jednej rovine, narozdiel od všeobecných mnohouholníkov. Pre trojuholníky existujú veľmi rýchle algoritmy pre vypĺňovanie a jeho zobrazovanie je podporované grafikým procesorom.

Množinu trojuholníkov, ktorá zdieľa svoje hrany, nazývame sieťou (triangle mesh). Dátovú štruktúru, ktorá popisuje sieť, môžeme rozdeliť do dvoch častí a to:

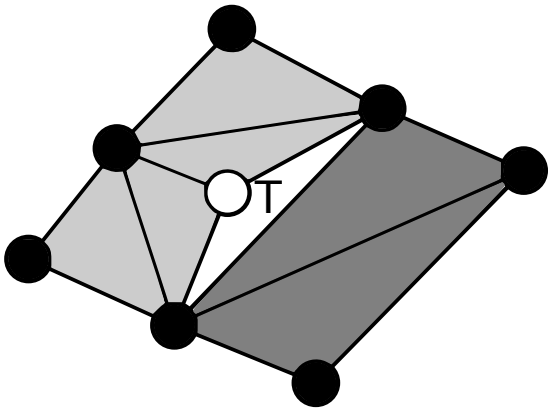
geometrická časť: Zaznamenáva súradnice vrcholov trojuholníka.

topologická časť: Uchováva údaje o vrcholoch tvoriace trojuholník.

Toto rozdelenie umožňuje vykonávať geometrické transformácie so zmenou súradnic vrcholov len so zmenou geometrickej časti a nie je potrebná zmena topologickej časti [11].

3.1.1 T-vrcholy

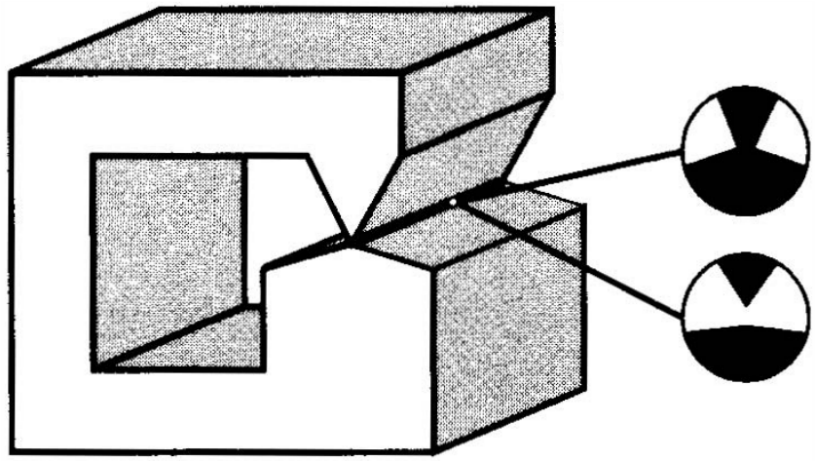
Sieť trojuholníkov má aj svoju nevýhodu, a tou je geometrický alias, ktorý sa prejavuje pri zmene mierky. Najčastejší jav ktorý to spôsobuje sa nazýva T-vchol. Tento jav je možné vidieť na obrázku 3.1. Pri pozícii vrchola nastáva numerická chyba, ktorá je dôsledkom toho, že pamäť počítača nie je nekonečná. Táto malá vzdialenosť zapríčiňuje, že zo vzdialenosti je vidieť objekt spojený, ale po priblížení je vidieť v objekte dieru. T-vrcholy často vznikajú pri priebežnej zmene úrovne detailov alebo pri spojení dvoch sietí trojuholníkov [11].



Obr. 3.1: Na obrázku vidieť roztrhnutie siete pri T-vrchole, ktoré vytvára v objekte dieru.

3.1.2 Manifoldy

Ako bolo spomenuté vyššie, objekt je tvorený vnútornými a hraničnými bodmi. Táto reprezentácia ale nie je úplne vhodná, pretože popisuje aj také objekty, ktoré sa nedajú v skutočnom svete vytvoriť. Pojem manifold sa zaviedol práve pre také prípady, kedy je možné objekt vytvoriť aj v skutočnom svete. Objekt, ktorý sa nedá vytvoriť sa nazýva nonmanifold. Príklad takéhoto objektu s možným riešením prevodu na manifold môžeme vidieť na obrázku 3.2. Vo všeobecnosti môžeme povedať, že manifold je objekt, ktorého každá hrana spája práve dve plochy [11].



Obr. 3.2: Nonmanifold a dve možnosti jeho prevodu na manifold. Zdroj: [11]

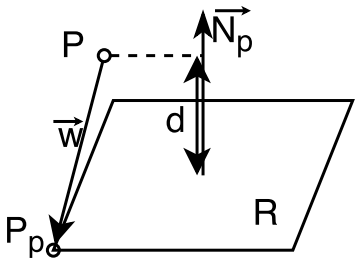
3.2 Potrebné geometrické operácie

Na tvorbu plátkov na vystrihovačky Sliceforms bolo potrebné využiť niektoré geometrické operácie s úsečkou a plochou. Tieto operácie budú následovne popísané v tejto sekcii.

3.2.1 Vzdialenosť bodu od roviny

Vzdialenosť bodu od roviny sa vypočíta pomocou vzorca (3.1), kde sa premietne vektor $\overrightarrow{w}$ na normálu plochy $\overrightarrow{N_p}$. Ak vzdialenosť bodov P_1 a P_2 úsečky U je od roviny R rovná nule ($d = 0$), dá sa povedať, že úsečka U leží na rovine R.

$$d = \overrightarrow{w} \cdot \overrightarrow{N_p} \tag{3.1}$$



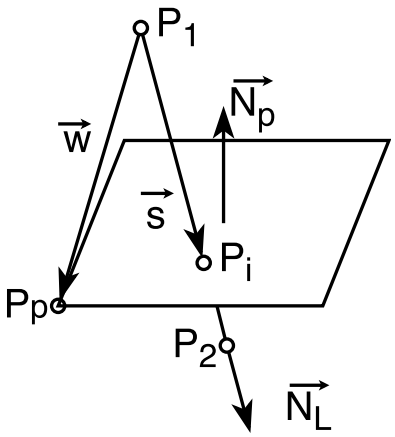
Obr. 3.3: Vzdialenosť bodu od roviny.

3.2.2 Priesečník úsečky a roviny

Bod pretnutia úsečky a roviny sa vypočíta pomocou vzorca (3.3). Pomocou tohto vzorca sa dostane vzdialenosť $|s|$, ktorá keď sa vynásobí normálou úsečky vznikne vektor $\overrightarrow{s}$, ktorý sa pripočíta k bodu P_1. Takto sa získa výsledná pozícia bodu pretnutia. Keďže tento výpočet zistuje bod pretnutia aj mimo úsečky $[P_1, P_2]$, je potrebné zistiť, či sa získaný bod aj skutočne nachádza medzi bodmi P_1 a P_2. Ak je skalárny súčin vektorov $\overrightarrow{N_L}$ a $\overrightarrow{N_p}$ rovný nule znamená, že je úsečka rovnobežná s rovinou.

$$|s| = \frac{\overrightarrow{w} \cdot \overrightarrow{N_p}}{\overrightarrow{N_L} \cdot \overrightarrow{N_p}} \tag{3.2}$$

$$P_i = P_1 + |s| \times \overrightarrow{N_L} \tag{3.3}$$



Obr. 3.4: Priesečník úsečky a roviny.

3.3 Herný engine Unity

Herný engine (herné jadro) je špecializovaný softvér na vytváranie nielen hier, ale aj rôznych aplikácií. Kedysi boli herné jadrá veľmi drahé a komplikované, a preto boli dostupné iba pre veľké štúdia. V dnešnej dobe sú ale jednoduchšie a prístupné aj pre jednotlivcov. Jeden z najrozšírenejších herných jadier je Unity. Unity je používaný rozmanitou komunitou vývojárov ako sú študenti a fanatici, ale aj mnohými organizáciami, ktoré používajú Unity na vývoj hier [9].

Herný engine Unity pracuje so skriptami, ktoré môžu byť napísané v jazyku Javascript alebo v jazyku C#. Pomocou týchto skriptov sa ovládajú jednotlivé herné objekty. Tieto skripty sú spolu s materiálmi a modelmi uložené v priečinku Assets.

Výsledná aplikácia v Unity je multiplatformová, stačí iba pri vytváraní aplikácie zvoliť cielenú platformu.

Prostredie Unity je rozdelené na [9]:

Scene (Scéna): Používa sa na výber herných objektov a navigáciu.

Game (Hra): Táto časť je na testovanie aplikácie a umožňuje rovnaký pohľad na aplikáciu ako bude mať užívateľ.

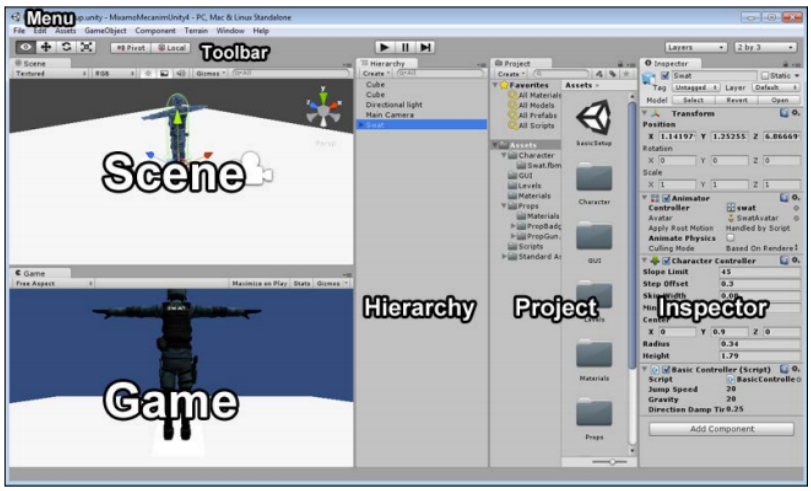
Hierarchy (Hierarchia): Zobrazuje herné objekty ako Herné objekty sú zobrazené v stromovej štruktúre.

Project (Projekt): Táto časť obsahuje assety ako sú skripty, textúry a modely.

Inspector (Inšpektor): Pomocou inšpektora je možné nastavovať herné objekty a komponenty, ktoré sú k nemu pripojené.

Toolbar (Panel nástrojov): Obsahuje transformačné nástroje a nástroje na spúšťanie aplikácie.

Menu (Menu): Umožňuje prístup k rôznym príkazom, ktoré zahrňujú vytvorenie herného objektu, komponenty alebo zobrazenie manuálu.



Obr. 3.5: Prostredie herného jadra Unity. Zdroj: [9]

3.4 Formát OBJ

Keďže herný engine Unity neumožnuje natívne nahratie modelu počas behu aplikácie, bolo nutné sa zaoberať aj touto záležitosťou a vytvoriť aspoň jednoduchý spôsob pre import 3D modelu. Rozhodol som sa pre súborový formát `OBJ`, kvôli jeho dostupnosti a zároveň jednoduchosti pri spracovaní.

OBJ je geometrický súborový formát, ktorý bol vytvorený spoločnosťou Wavefront Technologies pre ich animačný nástroj Advanced Visualizer. Tento súborový formát je v súčastnosti verejne prístupný a bol prevzatý aj ďalšími firmami, ktoré sa zaoberajú tvorbou 3D grafických aplikácií a stal sa jedným z najpopulárnejších formátov pre popis 3D modelov.

Tento súborový formát je jednoduchý dátový formát, ktorý reprezentuje 3D polygonálne modely a je uložený ako prostý text. 3D polygonálne modely sú definované pomocou pozície jednotlivých vrcholov, súradnice textúry na vrchole, normály vertexu, a plochou (face), ktorá tvorí jednotlivé polygóny, a je definovaná ako list bodov, textúr a normál.

Názov objektu: (object name)

```
o názov
```

V jednom súbore môže byť uložených aj viac objektov. Toho sa dá docieliť pomocou tohto elementu. Tento element združuje prvky, ktoré sú po ňom následne definované.

Komentár: (comment)

```
# komentár
```

Riadok začínajúci sa znakom # sa vždy ignoruje. Ide o riadok s komentárom, ktorý zvyčajne obsahuje informácie o programe, pomocou ktorého bol súbor vytvorený. Obvykle tiež obsahuje aj počet vrcholov alebo plôch, ktoré sú v objekte použité.

Vrchol: (vertex)

```
v x y z
```

Vrcholy objektu sú označované znakom `v`. Špecifukujú geometrické vrcholy a ich x, y a z koordináty. Tieto koordináty sa zapisujú ako čísla s plávajúcou desatinnou čiarkou.

Plocha: (face)

```
f v1[/[vt1][/vn1]] v2[/[vt2][/vn2]] v3[/[vt3][/vn3]] ...
```

`f` označuje plochu. Plochy sa skladajú z troch alebo viac trojíc čísel špecifikujúce číslo geometrického vrchola `v`, textúry vrchola `vt` a normály vrchola `vn`. Textúra vrchola a normála vrchola sú nepovinné parametre. Táto trojica čísiel je oddelená pomocou znaku `/`.

Ďalšie možnosti, ktoré formát `OBJ` ponúka sú zhrnuté v manuále [10].

Ukážka jednoduchej polygonálnej kocky vo formáte OBJ vytvorená iba pomocou vertexov a plôch:

```
# kocka.obj
o kocka
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 1.000000 1.000000 1.000000
v -1.000000 1.000000 1.000000
v -1.000000 1.000000 -1.000000
f 5 1 4
f 5 4 8
f 3 7 8
f 3 8 4
f 2 6 3
f 6 7 3
f 1 5 2
f 5 6 2
f 5 8 6
f 8 7 6
f 1 2 3
f 1 3 4
```

Kapitola 4

Navrhnuté riešenie generovania prierezov

Pred začatím tvorby prierezov je potrebné načítaný objekt normalizovať, teda zmeniť jeho veľkosť tak, že najväčšia vzdialenosť v niektorej osi bude rovná jedna. Tento objekt sa ale nemusí nachádzať na správnej pozícii, preto je treba jeho stred umiestniť na bod $[0, 0, 0]$.

Generovanie prierezov prebieha pre každú os zvlášť a to v dvoch fázach. Generovanie sa začína od stredu a postupným pripočítavaním vektoru posunu k reznej ploche sa generujú ďalšie prierezy. Tento proces trvá až pokiaľ nedôjde k bodu, kde rez nepretne plochu objektu. V tomto bode sa generovanie presunie na zápornú časť objektu, a tam sa znova začnú generovať prierezy ale tentoraz sa už vektor posunu odpočítava.

4.1 Postup generovania prierezov

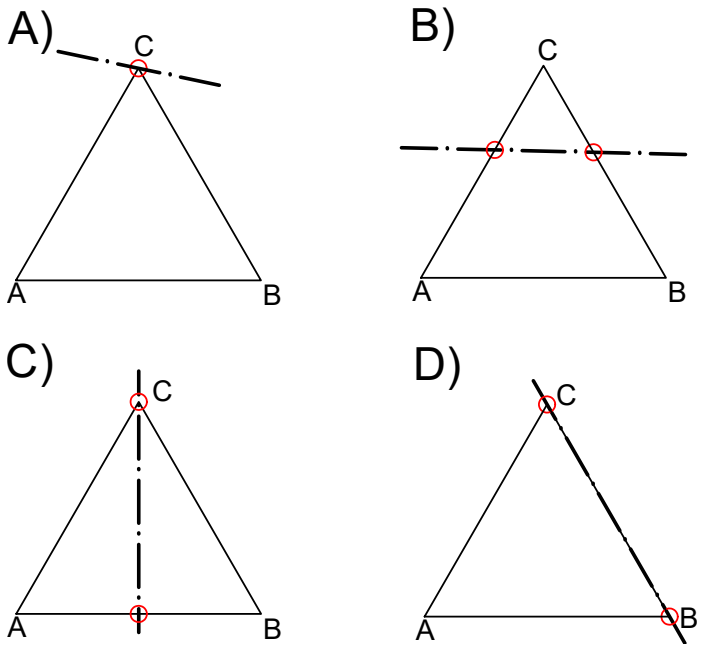
Prierezy generujeme tak, že prechádzame všetky trojuholníky v polygonálnom modele a testujeme na pretnutie reznou rovinou. Ak je trojuholník pretnutý, zaznamenáme úsečku, v ktorej sme ho pretli. To sa deje na základe algoritmu 1. Tento algoritmus najprv zistí pozície bodov trojuholníku a následne zistuje, ktoré úsečky medzi bodmi tojuholníka sa pretínajú s reznou rovinou. Sú 4 možnosti ako môže rovina prechádzať cez trojuholník. Tieto možnosti sú zobrazené na obrázku 4.1.

Vo variante A a variante B pretína rovina úsečky $[A, C]$ a $[B, C]$. V možnosti A sa priesečník týchto dvoch úsečiek s rovinou nachádza práve v bode C. Tieto dve varianty berem ako rovnocenné, keďže nám pridanie jedného bodu viacnásobne výsledný tvar nijak neovplivňuje a po nájdení všetkých bodov a následnom spojení sa zredukujú body, ktoré sa nachádzajú na približne rovnakej súradnici. Pri variante C prechádza rovina cez všetky tri úsečky. Úsečky $[A, C]$ a $[B, C]$ pretína práve v jednom bode C a úsečku $[A, B]$ pretína niekde medzi bodmi A a B.Vo variante D rovina leží na úsečke $[B, C]$. Je pravda, že prechádza tiež cez úsečky $[A, B]$ a $[C, A]$ ale kedže túto variantu – ako je vidno v algoritme 1 – testujem ako prvú, tak sa zaznamenajú iba body B a C. Tento proces sa opakuje pre každý trojuholník v polygonálnom modely.

Po nájdení všetkých úsečiek v trojuholníkoch je potrebné úsečky pospájať podľa postupnosti a tak vytvoriť výsledný obrys mnohouholníka. Sieť trojuholníkov ale môže obsahovať aj T-vrcholy, ktorým sme sa venovali v časti 3.1.1, a preto je nutné pri porovnávaní bodov u nájdených úsečiek počítať aj s odchylkou.

Algorithm 1 Algoritmus pre nájdenie bodov, v ktorých sa pretína model s rovinou

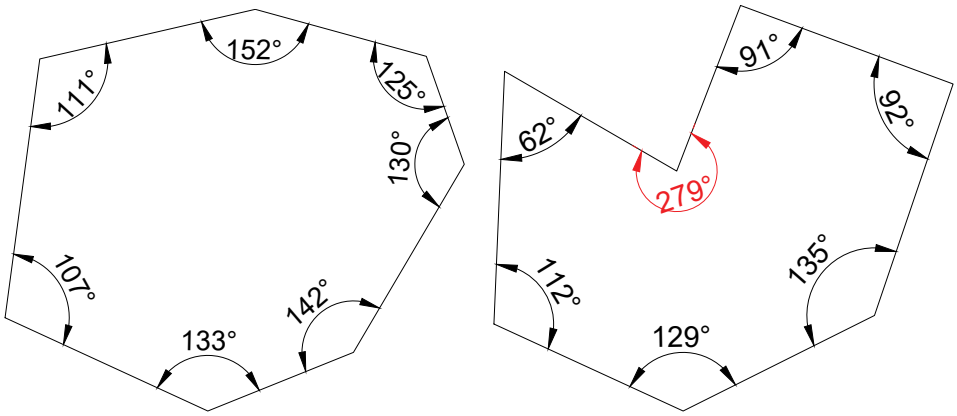
```
1: function NÁJDI PRETNUTIE(Rezacia rovina, Sieť objektu)
2:     for Všetky trojhuholníky do
3:         Bod1 = Prvý bod trojuholníka
4:         Bod2 = Druhý bod trojuholníka
5:         Bod3 = Tretí bod trojuholníka
       // Ak rovina prechádza po úsečke
6:         if Prechádza po úsečke(Bod1, Bod2) then
7:             Zaznamenaj(Bod1,Bod2)
8:         if Prechádza po úsečke(Bod2, Bod3) then
9:             Zaznamenaj(Bod2,Bod3)
10:        if Prechádza po úsečke(Bod3, Bod1) then
11:            Zaznamenaj(Bod3,Bod1)
12:        if Prechádza po niektorej úsečke then
13:            Choď na ďalší trojuholník
       //Ak prechádza cez všetky 3 úsečky znamená, že prechádza cez niektorý vrchol
14:        if Prechádza cez všetky tri úsečky then
15:            if Bod pretnutia 1 == Bod pretnutia 2 then
16:                Zaznamenaj(Bod pretnutia 3,Bod pretnutia 1))
17:            if Bod pretnutia 2 == Bod pretnutia 3 then
18:                Zaznamenaj(Bod pretnutia 1,Bod pretnutia 2))
19:            if Bod pretnutia 3 == Bod pretnutia 1 then
20:                Zaznamenaj(Bod pretnutia 2,Bod pretnutia 3))
21:        if Prechádza cez dve úsečky then
22:            if Prvý & Druhý then
23:                Zaznamenaj(Bod pretnutia 1, Bod pretnutia 2));
24:            if Tretí & Prvý then
25:                Zaznamenaj(Bod pretnutia 3, Bod pretnutia 1));
26:            if Druhý & Tretí then
27:                Zaznamenaj(Bod pretnutia 2, Bod pretnutia 3));
```



Obr. 4.1: Štyri možnosti pretnutia trojuholníka s rovinou.

4.1.1 Polygonálna triangulácia

Aby sme mohli vidieť vytvorené prierezy v 3D priestore, je potrebné vytvoriť sieť (mesh), ktorá je tvorená pomocou trojuholníkov. Triangulátor je nástroj, pomocou ktorého nájdeme možnosť, ako pospájať všetky body mnohouholníka do množiny trojuholníkov tak, aby sa trojuholníky neprekrývali a tým vyplniť vnútornú plochu mnohouholníka. Mnohouholníky sa rozdeľujú do dvoch skupín a to na konvexné a konkávne 4.2.



Obr. 4.2: Konvexný a konkávny polygón. **vľavo:** konvexný polygón má všetky z vnútorných uhlov $\leq 180°$. **vpravo:** konkávny polygón má jeden alebo viac z vnútorných uhlov $> 180°$.

Triangulácia konvexného mnohouholníka je veľmi jednoduchá. Stačí vybrať jeden bod mnohouholníka a spojiť ho s ostatnými bodmi. Pri konkávnom mnohouholníku je ale situácia rozdielna. Ak by sme postupovali rovnako, ako u konvexného mnohouholníka, mohli by sme zahrnúť do plochy aj časť, ktorá nepatrí do vnútornej časti polygónu. Preto musíme použiť niektorú z triangulačných metód. Pri tvorbe tejto práce som sa rozhodol využiť

triangulačnú metódu rezania uší (Ear Clipping method). Táto metóda má však jednu veľkú nevýhodu, a tou je, že nedokáže pracovať s polygónmi, ktoré obsahujú diery.

Ear Clipping method (Metóda rezania uší)

Metóda rezania uší sa zakladá na *teoréme dvoch uší*, ktorý sa často pripisuje Gary H. Meistersovi, ktorý ho objavil v roku 1975 [4]. Teorém dvoch uší však bol dokázaný už skôr Maxom Dehnom, ako časť dôkazu teorému Jordánových polygónov [3]. Jordánov polygón je jednoduchý uzavretý rovinný polygón s konečným počtom bodov. Pre Jordánove polygóny s bodmi $P = V_1V_1V_1 \dots V_n$ pre $n \geq 4$ platia nasledovné dve vyhlásenia [4]:

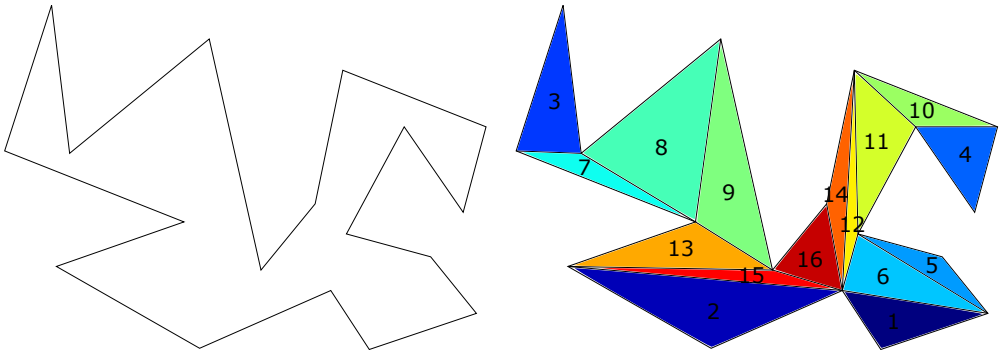
- Polygón P má aspoň dve neprekrývajúce sa uši.
- Polygón P umožňuje trianguláciu bez pridania nových vrcholov.

Ucho polygónu je definované: „Vrchol P_i jednoduchého polygónu P sa nazýva ucho v prípade, že uhlopriečka (P_{i-1}, P_{i+1}) leží celá v polygóne P. Hovoríme, že obe uši P_i, a P_j sú neprekrývajúce, ak vnútro trojuholníka (P_{i-1}, P_i, P_{i+1}) nepretína vnútro trojuholníka (P_{j-1}, P_j, P_{j+1}).“[5]

„Každý jednoduchý mnohouholník P s počtom bodov n po triangulácii pozostáva presne z $n - 2$ trojuholníkov.“[1] Ak máme mnohouholník s počtom bodov $n = 3$ vieme, že nám po triangulácii vznikne práve jeden trojuholník. Mnohouholník P, ktorý máme vyznačený na obrázku 4.3, má osemnásť vrcholov. Podľa vzorca $n - 2$ mnohouholník po polygonálnej triangulácii obsahuje práve šestnásť trojuholníkov.

Pred začatím triangulácie musíme najprv zistiť či polygón, ktorý sme premietli do 2D priestoru je pravotočivý alebo ľavotočivý. Natočenie polygónu zistíme pomocou sčítaní súčinu vektorov (P_{j-1}, P_j) a (P_{j-1}, P_{j+1}) na všetkých bodoch polygónu. Ak je tento súčet záporný, znamená to, že je polygón pravotočivý a ak je kladný, tak je polygón ľavotočivý. Aby sme docielili iba pravotočivých objektov, musíme pri ľavotočivých objektoch otočiť poradie bodov P.

Metóda rezania uší je vidieť na algoritme 2. V tejto metóde najprv zoberieme prvý bod P_i, kde $i = 1$ a testujeme, či sa v trojuholníku (P_{i-1}, P_i, P_{i+1}) nenachádza iný bod a zároveň či je trojuholník pravotočivý. Ak takýto trojuholník nájdeme, znamená že sme našli ucho mnohouholníka a teda ho môžeme odrezať. Tento algoritmus sa opakuje pre všetky body P, až pokým sa nezaznamená aj posledný trojuholník.



Obr. 4.3: Triangulácia polygónu metódou rezania uší. **Vľavo** sa nachádza spracovávaný mnohouholník a **vpravo** vytvorená sieť trojuholníkov.

Algorithm 2 Triangluačná metóda rezania uší

```
1: function TRIANGLUATE(P)
2:     WatchDog=PocetBodov
3:     while OstavajuciPocetBodov > 2 do
4:         WatchDog=WatchDog-1
5:         PosunNaDalšieBody
6:         if ŽiadnyBodVTrojuholníku && JePravotočivý then
7:             ZaznamenajBody
8:             OdrežUcho
9:             WatchDog=OstávajúciPočetBodov
10:        if WatchDog==0 then
11:            break
```

Do tohto algoritmu som pridal aj premennú watchdog, ktorá sa postupne dekrementuje až pokým sa nenájde ucho, ktoré by sa dalo odrezať. Pre túto variantu som sa rozhodol kvôli tomu, že sa často stretávam s nedokonalými modelmi, ktoré obsahujú prekrývajúce sa úsečky a teda sa algoritmus zacykluje. Tieto nepresnosti boli často spôsobené spojením dvoch objektov v niektorom z 3D modelovacích nástrojov.

Unity zobrazuje iba trojuholníky, ktoré sú v smere hodinových ručičiek. My ale chceme vidieť vytvorené plátky z oboch strán. Preto je nutné, pridať k vytvoreným trojuholníkom aj ich opačnú verziu pomocou zmeneny poradia bodov.

4.1.2 Vzdialenosť medzi prierezmi

V rannej dobe tvorby projektu som nechával na užívateľovi, aby nastavil vzdialenosť medzi jednotlivými prierezmi. Tento proces ale nebol dostatočne intuitívny, a preto som sa rozhodol umožniť užívateľovi definovať, aký počet rezov má byť použitý pre tvorbu prierezov. Pri generovaní som ale musel tento počet následne pretransformovať na vzdialenosť d_{span}.

Najprv si zistíme body P_1 a P_2, ktoré sa pretínajú najďalej od stredu S s rezom R, ktorý má rovnaké natočenie ako budú mať vytvorené prierezy. Pomocou zistených vzdialeností som získal celkovú vzdialenosť a vydelil ju potrebným počtom medzier medzi rezmi N. Kedže ale nechceme, aby sa prierezy generovali hneď od okraja objektu, tak je treba zinkrementovať počet medzier medzi rezmi o jeden a tým zabezpečiť, aby sa na okraji objektu vynechal priestor rovný polovici vzdialenosti medzi prierezmi $\frac{d_{span}}{2}$.

$$d_{span} = \frac{d_{P_1P_2}}{N+1} \tag{4.1}$$

4.1.3 Posuv rezov

Keďže použitý algoritmus generuje prierezy od stredu objektu, tak pri párnom počte by neboli prierezy rozložené rovnomerne. Kvôli tomu je nutné posunúť počiatočný prierez o polovicu vzdialenosti medzi rezmi a zabezpečiť tak rovnomernosť vygenerovaných prierezov.

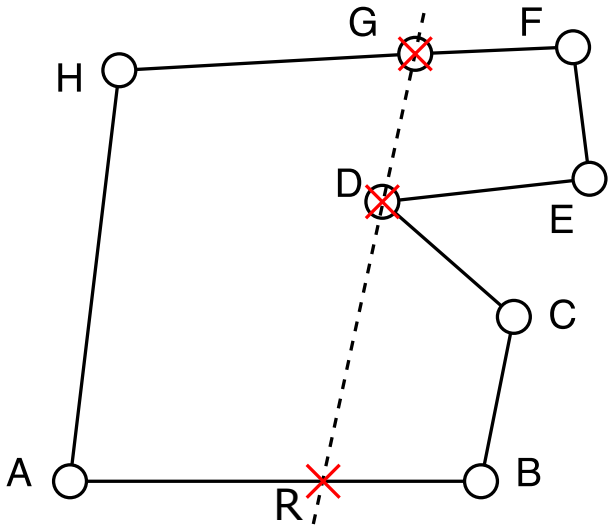
4.2 Tvorba zárezov do jednotlivých prierezov

Na to, aby sa dali skladačky Sliceforms skladať, musíme do jednotlivých plátkov urobiť zárezy v mieste, kde sa plátky stretávajú s priečnymi plátkami. Tieto zárezy musíme vytvoriť tak, aby boli do polovice celej pretínacej vzdialenosti.

Najprv musíme nájsť všetky body, v ktorých sa pretína polygón s kolmím plátkom. Na obrázku 4.4 máme mnohouholník $P = \{A, B, C, D, E, F, G, H\}$, ktorý pretína rovina R. Táto rovina pretína polygón cez päť úsečiek. Spracovanie týchto úsečiek si rozdelíme na tri časti.

- Úsečku $[A, B]$ pretína rovina medzi bodmi A a B. Zaznamenáme si tento bod pretnutia.
- Úsečku $[C, D]$ a úsečku $[D, E]$ pretína rovina práve v jednom bode, a tým je bod D. Keďže body C a E sa nachádzajú na rovnakej strane roviny, zaznamenáme bod D dvakrát, ako keby rovina pretínala úsečky niekde medzi bodmi.
- Posledná tretia varianta spracovania je pre úsečky, ktoré rovina pretína práve v jednom bode a zároveň sú ďalšie doby úsečiek na rôznych stranách roviny. V prípade mnohouholníka P sú to úsečky $[F, G]$ a $[G, H]$. Rovina prechádza práve cez bod G, a body F a H sú na rôznych stranách roviny. V tejto variante zaznamenávame spoločný bod G iba raz.

To, či sú body na rovnakej alebo rozličnej strane, zisťujeme pomocou vypočítania vzdialenosti bodu od roviny, ktorej sme sa venovali v časti 3.2.1. Ak majú vypočítané vzdialenosti rovnaké znamienka, spracovávame ich podľa druhej varianty. Ak majú rôzne znamienka, spracovávame ich podľa tretej varianty.



Obr. 4.4: Tento obrázok zobrazuje pretnutie mnohouholníka rovinou. Rovina pretína mnohouholník práve v troch bodoch, a to na úsečke [AB], v bode D a v bode G.

Ako prvé čo musíme urobiť je, že si nájdené body musíme zoradiť, kedže pri väčšine mnohouholníkov nevieme, ktorou úsečkou sa hľadanie začalo a nájdené body môžu byť v zlom poradí. Keď už máme body nájdené a zoradené, môžeme začať vyznačovať zárezy. Zárezy v X-ovej osi musia byť opačne ako zárezy v Y-ovej osi. Pri vyznačovaní zárezov

berieme body po dvoch. Postupne spájame jeden z týchto bodov podľa toho, v ktorej osi zárezy robíme, s bodom, ktorý sa nachádza práve v strede medzi týmito dvoma bodmi. S týmto postupom docielime, že sa nám správne vyznačia zárezy, aby sme mohli výsledný objekt správne vystrihnúť a poskladať.

4.3 Finálny produkt

Aby sa vytvorený prierezový model dal použiť ako vystrihovačka, musel som vymyslieť ako ho uložiť, aby sa dal vytlačiť a vystrihnúť. Navrhol som päť možností, ktoré sa rozdeľujú do troch tried. Sú to:

Modelové: Model sa uloží do súboru OBJ, a každý prierez sa uloží ako samostatný objekt. Túto možnosť som zvolil kvôli tomu, aby si užívateľ mohol danú vystrihovačku ešte pozrieť a použiť ako prípadnú pomôcku pri skladaní.

Vektorové: Vo formáte SVG alebo PDF. Pri SVG sa ukladá každý prierez do samostatného súboru. Pri PDF sa uložia všetky prierezy do jedného súboru, kde sa poukladajú podľa užívateľom zvoleného rozloženia prierezov na daný rozmer strany. Možnosť ukladania do PDF je preferovaná, keďže sa dá bez ďalších úprav hneď vytlačiť pomocou tlačiarne.

Rastrové: Pri tejto možnosti sa nám najprv vytvoria textúry, na ktoré sa vykreslia plátky. Každá z jednotlivých textúr sa uloží do samostatného súboru a to vo formáte JPG alebo PNG. Pre jednoduchšiu manipuláciu s vytvorenými súbormi som zvolil jednotný rozmer tvorených textúr.

4.3.1 Ukladanie do PDF

Pri formáte PDF bolo nutné zvoliť si nástroj na tvorenie súborov. Našiel som niekoľko nástrojov kde väčšina pracovala s .NET 4,0 ale keďže herný engine používa iba .NET 3,5 tak som ich nemohol použiť. Nakoniec som sa dostal k dvom nástrojom a to SharpPdf[1] a iTextSharp[2].

SharpPdf: Tento nástroj som používal ako prvý. Umožňuje vytvárať veľmi primitívne dokumenty. Hlavným prvkom je objekt pdfDokument, na ktorý sa postupne pridávajú objekty pdfPage, na ktoré sa dá písať text alebo vykreslovať jednoduché tvary ako čiary, obdĺžniky a kruhy.

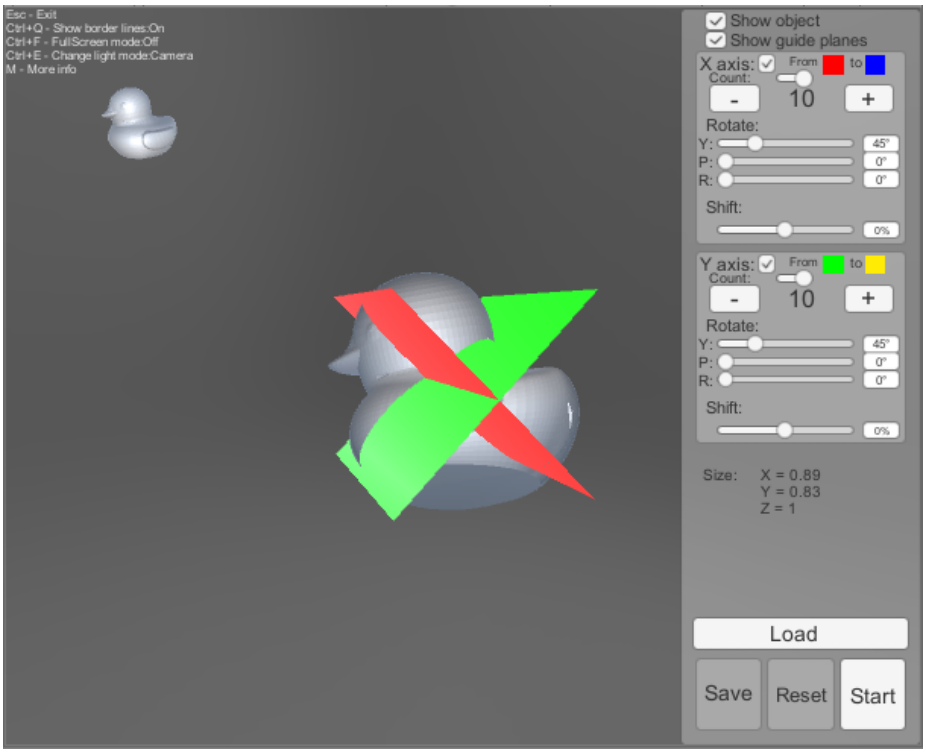
iTextSharp: iTextSharp umožňuje oproti SharpPdf vytvárať komplexnejšie dokumenty. Hlavnou výhodou pre mňa bolo, že dokáže vykreslovať mnohouholníky, ktoré sa dajú následne vyplniť zvolenou farbou.

[1] http://sharppdf.sourceforge.net/
[2] http://itextpdf.com/

Kapitola 5

Návrh užívateľského rozhrania

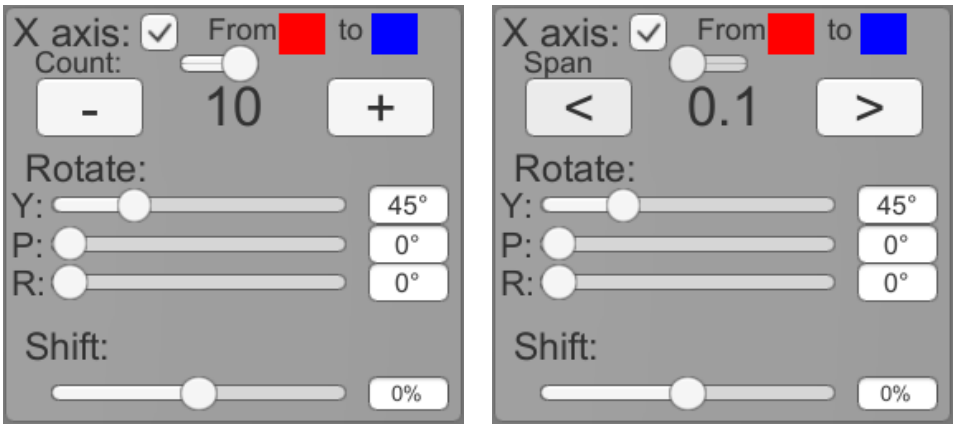
Každá aplikácia potrebuje komunikovať s užívateľom. Na to slúži užívateľské rozhranie, pomocou ktorého si užívateľ definuje v akom natočení majú byť prierezy robené a počet rezov. Užívateľské rozhranie bolo implementované v spolupráci s užívateľmi. Každá časť užívateľského rozhrania bola otestovaná na užívateľoch.



Obr. 5.1: Na obrázku sa nachádza okno aplikácie, na ktorom je vidno použité užívateľské rozhranie.

Výzor užívateľského rozhrania môžeme vidieť na obrázku 5.1. V strede obrazovky sa nachádzajú dve rezné plochy - jedna pre každú os. Po načítaní objektu sa na tomto mieste zobrazí zvolený objekt, u ktorého je možné vidieť iba vnútornú stranu objektu. Aby si užívateľ vedel daný objekt lepšie predstaviť, vytvoril som náhlad, ktorý sa nachádza v ľavej hornej časti okna.

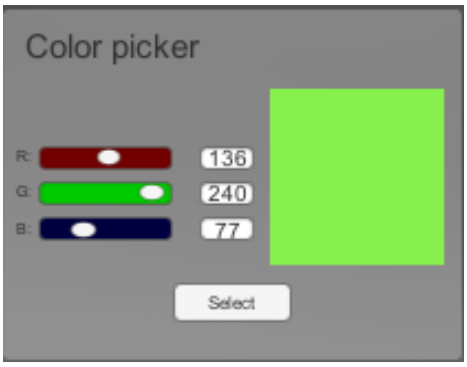
Na pravej strane okna sa nachádza panel nastavení. Tento panel obsahuje dva panely, pomocou ktorých je možné nastaviť počet tvorených prierezov alebo rozmer, ktorý bude použitý na tvorbu prierezov. Tento panel môžeme vidieť aj na obrázku 5.2. Pomocou tohto panela si môže užívateľ zvoliť natočenie reznej plochy a aj prípadný posuv. Ďalej si v tomto panely môže zmeniť farbu vytváraných prierezov a pomocou prepínača môže vytvorené prierezy v danej osi skryť.



Obr. 5.2: Panel pre nastavenie tvorby rezov. **Vľavo** je pre nastavenie počtu tvorených rezov a **vpravo** je pre nastavenie rozmeru medzi tvorenými plátkami.

Na spodnej časti panela nastavení sa nachádzajú tlačidlá pre zobrazenie panela na zvolenie súboru a panel na uloženie vytvorených plátkov. Taktiež sa tu nachádza aj tlačidlo pre tvorbu prierezov a tlačidlo, pomocou ktorého sa vytvorené plátky zmažú. Tlačidlo pre ukladanie sa sprístupní užívateľovi až po vytvorení prierezov a tlačidlo pre tvorbu prierezov sa sprístupní až po zvolení objektu na spracovanie. Pri sprístupnení sa zmení farba tlačidla. Toto opatrenie som zvolil kvôli zjednodušeniu užívateľského rozhrania, kde užívateľ stále vie, aké operácie sú prístupné.

Užívateľovi tiež umožňujem vybrať aj v akej farbe sa plátky zobrazia. Na tento účel som vytvoril jednoduchý panel na výber farby (color picker), ktorý môžeme vidieť na obrázku 5.3. Na tomto panely sa nachádzajú tri posuvníky (slider), pomocou ktorých si užívateľ môže zvoliť veľkosť červenej, zelenej a modrej zložky farby.

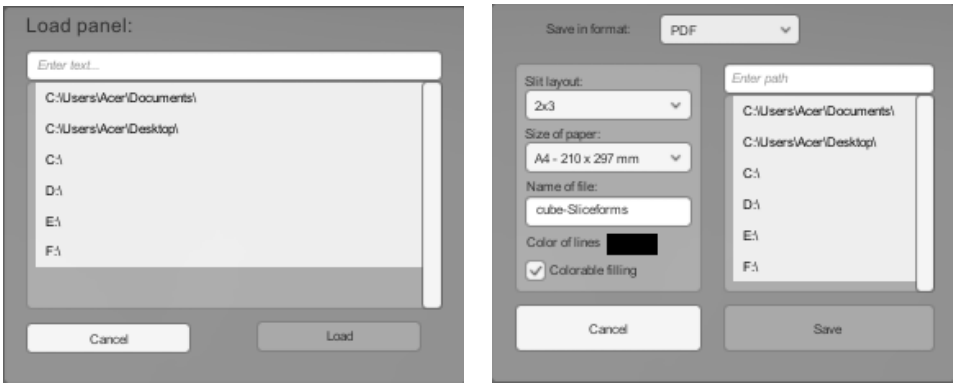


Obr. 5.3: Panel pre zvolenie farby

Vytvoril som aj jednoduchý prehliadač súborov (file explorer), ktorý môžeme vidieť na obrázku 5.4. Pomocou tohto prehliadača môže užívateľ prechádzať súbory a vybrať

súbor vo formáte OBJ, ktorý sa bude spracovávať, prípadne zvoliť si cestu pre uloženie vytvorených plátkov do súboru.

V panely ukladania môže užívateľ vybrať v akom formáte chce vytvorené plátky uložiť. Tieto možnosti sme si prešli v časti 4.3. Pri súborovom formáte PDF si môže užívateľ zvoliť veľkosť papiera (A5, A4, A3) a rozloženie v akom budú plátky uložené na papieri. Umožňujem sedem možností rozloženia, a to 1×1, 1×2, 2×3, 3×4, 4×5, 4×6 a 5×7. Ďalej si môže užívateľ zvoliť obvodovú farbu, a to či sa plátky vyplnia vybranou farbou, ktorou boli vytvorené alebo či ostanú bez výplne. Ukladanie sa spracováva pomocou nástroja na vytváranie pdf súborov, ktorému sme sa venovali v časti 4.3.1.



Obr. 5.4: **Vľavo** je panel pre zvolenie súboru vo formáte OBJ a **vpravo** je panel pre uloženie vytvorenej vystrihovačky.

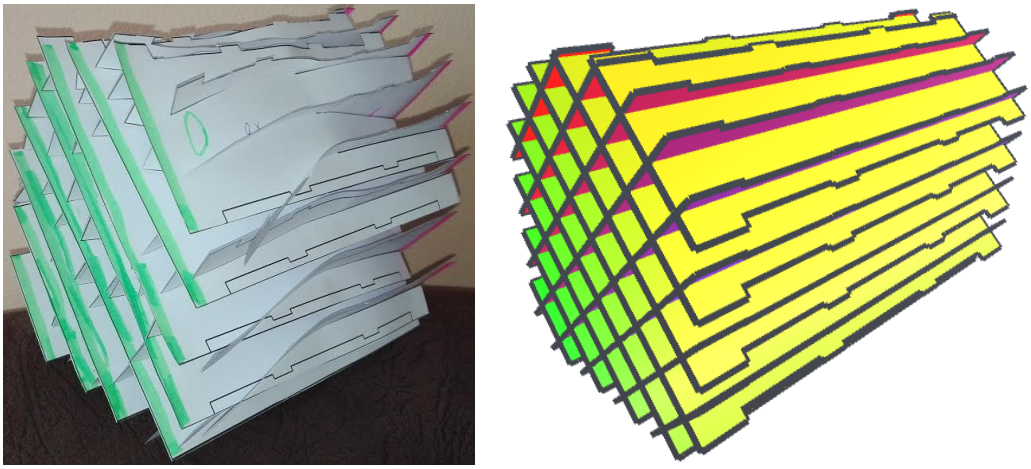
Aplikácia sa dá ovládať aj prostredníctvom klávesnice a myšky. Aplikácia umožňuje užívateľovi zobraziť obvodovú čiaru okolo vytvorených plátkov. Pomocou tejto obvodovej čiary je možné lepšie vidieť jednotivé plátky a pri skrytí niektorej z osí je možné vidieť aj vytvorené zárezy. Táto funkcia sa zapína klávesmi `Ctrl` a `Q`. V aplikácii je možné prepínať medzi zobrazením v okne alebo zobrazením na celú obrazovku pomocou kláves `Ctrl` a `F`. Ďalšie nastavenie, ktoré aplikácia umožňuje, je zmena osvetlovania načítaného objektu pomocou kláves `Ctrl` a `E`. Osvetlenie objektu sa prepína medzi svetlom z kamery, svetlom krúžiacim okolo objektu a statickým svetlom. Kameru je možné ovládať myšou, šípkami alebo numerickou klávesnicou. Aplikáciu je možné vypnúť pomocou klávesy `Esc`.

Tieto možnosti ovládania s klávesami sú užívateľovi zdelené v ľavom hornom rohu, kde sa pomocou klávesy `M` zobrazia aj klávesy použiteľné k ovládaniu kamery.

Kapitola 6

Experimentálne vyhodnotenie vytvoreného riešenia

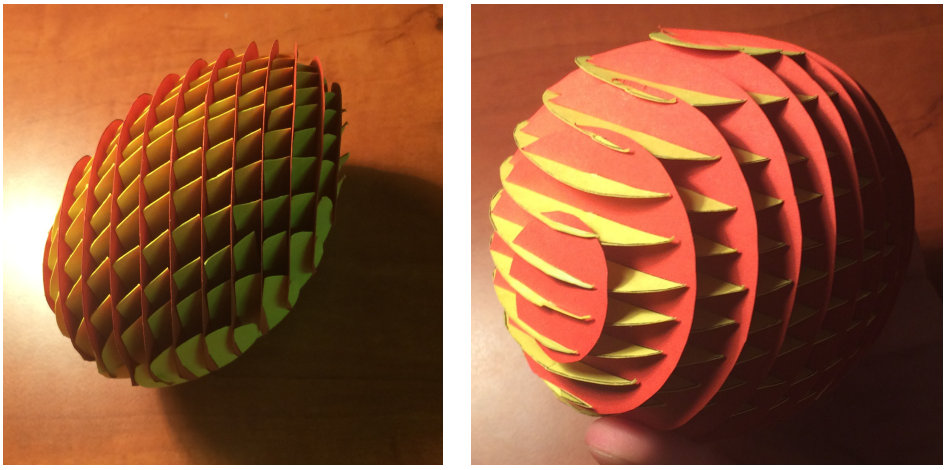
Ako prvé som sa rozhodol skladať iba jednoduché objekty. Ako prvý objekt som poskladal truhličku 6.1. Táto skladačka bola vytvorená iba z obyčajného kancelárskeho papiera (80g), ktorý je veľmi mäkký, a preto vytvorený model je veľmi labilný. Bolo ťažké skladať túto skladačku aj keď je tvorená iba z ôsmich rezov a bola vytlačená vo veľkosti formátu A5.



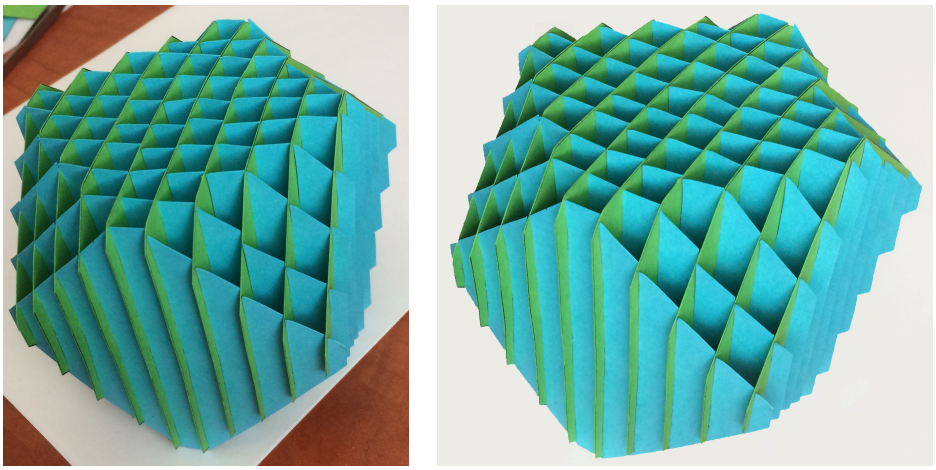
Obr. 6.1: Truhlička. **Vľavo** je v skutočnosti a **vpravo** v programe.

Ako ďalší objekt som poskladal guľu 6.2 a cubotahedron 6.3, čo je v podstate kocka s odrezanými vrcholmi. Tieto objekty sú poskladané z farebného papiera s gramážou 225g. Pri guli boli zárezy tvorené jedným zastrihnutím do prierezu v strede vyznačenej plochy. Táto skladačka sa veľmi ťažko skladala, pretože takto vytvorené zárezy sú veľmi úzke a teda priestor v zároze nebol dostatočný pre vkladané prierezy. Tento objekt sa po zložení stále zmršťoval a bolo treba jemné tlačenie na hrany, inak by nepripomínal guľu ale skôr vajce. Pri cubotahedrone bolo do každého zárezu zastrihnuté dvakrát, ako je uvedené na obrázku 2.2. Bolo oveľa jednoduchšie skladať túto skladačku a výsledný objekt si držal aj svoj pôvodný tvar.

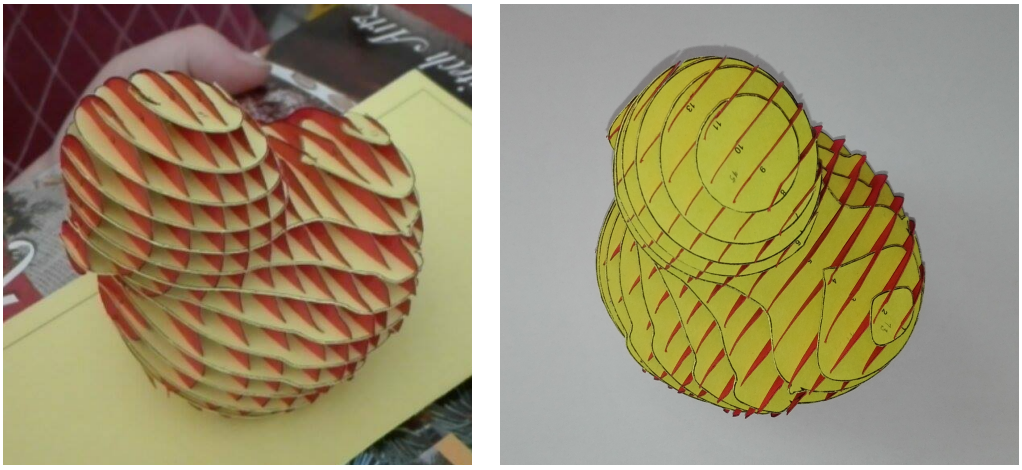
Po vytvorení týchto jednoduchých objektoch som sa rozhodol pre vytvorenie zložitejšieho modelu a to modelu gumennej kačičky, ktorý môžeme vidieť na obrázku 6.4.



Obr. 6.2: Guľa



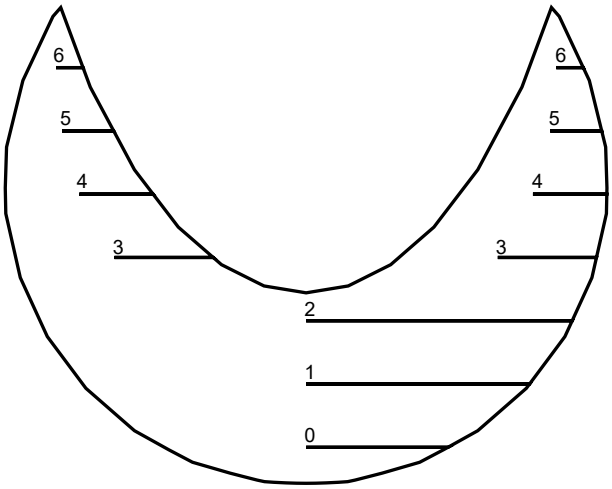
Obr. 6.3: Cubotahedron



Obr. 6.4: Gumená kačička

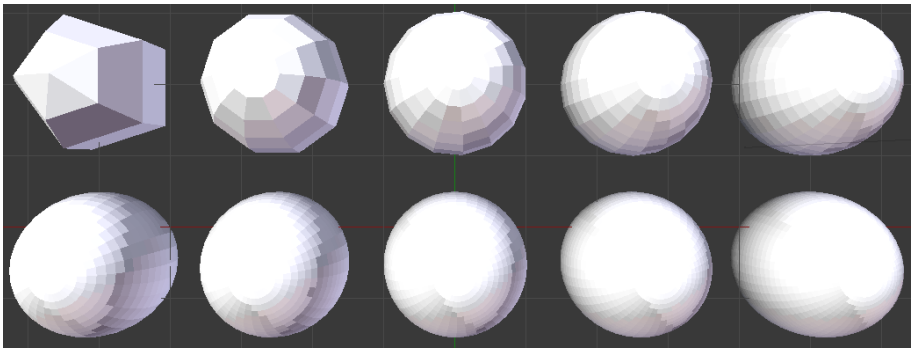
6.1 Označenie zárezov

Pri skladaní prvých objektov som zistil, že je veľmi náročné pre užívateľa začať skladať keď nevie, do ktorého zárezu má ísť ktorý plátok. Preto som sa rozhodol pridať pomôcku pre skladajúceho a označiť jednotlivé zárezy indexom prierezu. Výslednú ukážku je možné vidieť na obrázku 6.5.



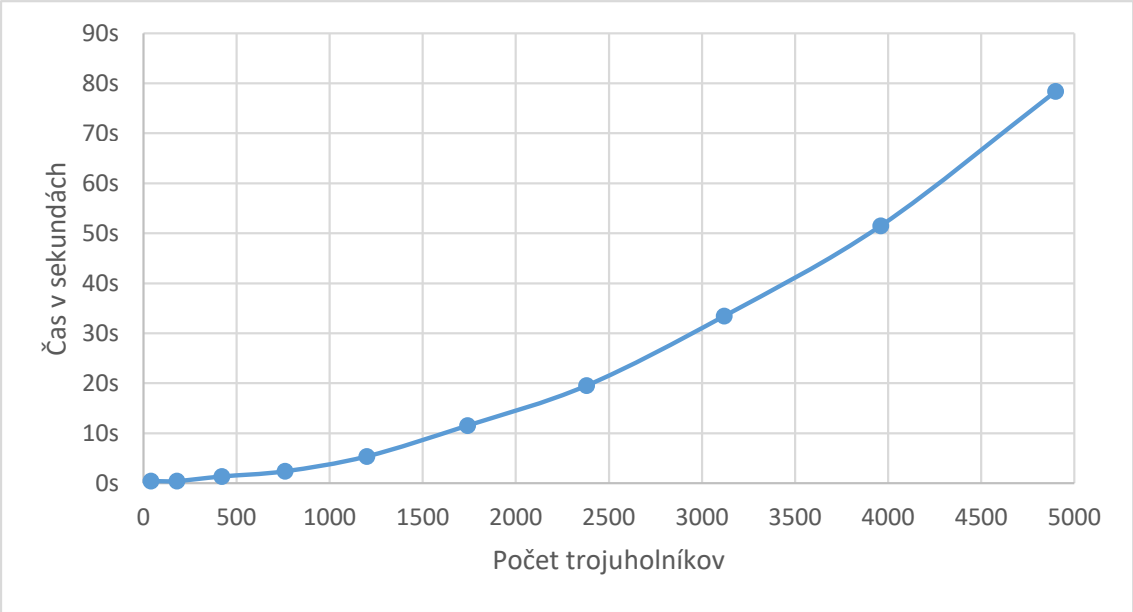
Obr. 6.5: Ukážka výsledného rezu s označením zárezov v pdf

6.2 Výkonový test



Obr. 6.6: Objekt gule použitý na test

Na zaznamenanie doby výpočtu bol použitý objekt gule, v ktorej bol navyšovaný počet plôch v stĺpcoch a riadkoch. Tieto modely môžeme vidieť na obrázku 6.6. Guľu som si vybral preto, že je vždy konvexná a teda je doba generovania viac zameraná na tvorbu prierezov a zárezov do nich a nie na použitý triangulátor. K testovanému modelu bolo postupne pridávaných 5 riadkov a 5 stĺpcov. Výsledok tohto testu je zobrazený na grafe 6.7.



Obr. 6.7: Doba generovania skladačky Sliceform

Kapitola 7

Záver

V tejto práci som popísal vystrihovačky Sliceforms aj s matematikou a geometriou na ich tvorbu. Tento nástroj na tvorbu skladačiek som v práci navrhol a následne aj implementoval v hernom jadre Unity.

Táto aplikácia vytvorí prierezy podľa užívateľom definovaného natočenia a posuvu zo zadaného polygonálneho objektu. Do týchto vytvorených prierezov vytvorí zárezy v mieste, kde sa plátky pretínajú s kolmými plátkami. Tieto vytvorené plátky si môže užívateľ pozrieť v 3D priestore a následne uložiť do niektorého z dostupných súborových formátov. Vytvorený nástroj aj s užívateľským rozhraním bol konzultovaný a testovaný na užívateľoch. Ako som na konci práce ukázal, po vytlačení na papier a následnom vystrihnutí si môže užívateľ vytvorenú skladačku aj poskladať.

Je niekoľko možností ako aplikáciu vylepšiť. Napríklad by si aplikácia sama zvolila optimálne rozloženie prierezov tak, aby bolo viac prierezov v užších miestach a v širších miestach menej. Poprípade by umožnila užívateľovi upraviť rozloženie prierezov tak, aby si mohol ľubovoľne zmeniť jednotlivé rozostupy medzi vybranými rezmi.

Literatúra

[1] Berg, M. d.; Cheong, O.; Kreveld, M. v.; aj.: *Computational Geometry: Algorithms and Applications*. Santa Clara, CA, USA: Springer-Verlag TELOS, třetí vydání, 2008, ISBN 3540779736, 9783540779735.

[2] Cue, K.: 3 Ellipsoid Collipsoid. apr 2013.
URL https://mathspig.wordpress.com/2013/04/09/3-ellipsoid-collipsoid/

[3] Guggenheimer, H.: The Jordan Curve Theorem and an Unpublished Manuscript by Max Dehn. *Archive for History of Exact Sciences*, ročník 17, č. 2, 1977: s. 193–200, ISSN 00039519, 14320657.
URL http://www.jstor.org/stable/41133486

[4] Meisters, G. H.: Polygons Have Ears. *The American Mathematical Monthly*, ročník 82, č. 6, Červen 1975: s. 648–651.

[5] Muhammad, R. B.: The Two-Ears Theorem.
URL http://www.personal.kent.edu/~rmuhamma/Compgeometry/MyCG/TwoEar/two-ear.htm

[6] O'Connor J. J., R. E. F.: Alexander Wilhelm von Brill. 2014.
URL http://www-history.mcs.st-andrews.ac.uk/Biographies/Brill.html

[7] Sharp, J.: *Sliceforms: Mathematical Models from Paper Sections*. Tarquin Publications, 1995, ISBN 9781899618064.
URL https://books.google.sk/books?id=c6wLAAAACAAJ

[8] Sharp, J.: How to make and assemble the models. nov 2010.
URL https://sliceforms.wordpress.com/2010/11/21/how-to-make-and-assemble-the-models/

[9] Smith, M.: *Unity 4.x Cookbook*. Packt Publishing, Limited, 2013, ISBN 9781849690430.
URL https://books.google.cz/books?id=AhVbTW9w_sMC

[10] Wavefront Technologies, 530 East Montecito Street, Santa Barbara, CA 93103: *B1. Object Files (.obj)*.
URL http://www.cs.utah.edu/~boulos/cs3505/obj_spec.pdf

[11] Žára, J.; Beneš, B.; Sochor, J.; aj.: *Moderní počítačová grafika*. Computer Press, 2004, ISBN 9788025104545.
URL https://books.google.cz/books?id=USQnAAAACAAJ

Prílohy

Príloha A

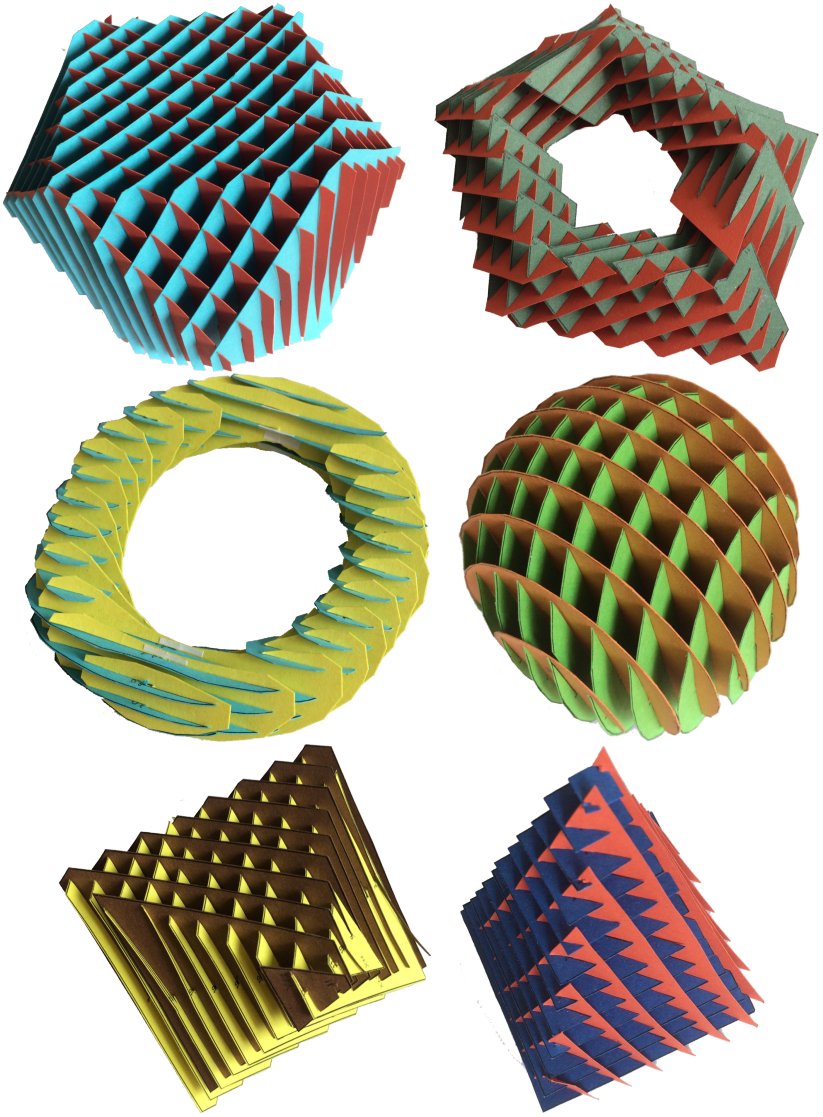
Obsah CD

Na priloženom disku sa nachádza:

- **3D modely** – Jednoduché 3D modely vo formáte OBJ
- **Aplikácia** – V tomto priečinku sa nachádza spustiteľná aplikácia
- **Dokumentácia Latex** – Zdrojový kód dokumentácie v Latex
- **Unity projekt** – Tento priečinok obsahuje Unity projekt zahrňujúci zdrojové súbory
- **Videá** – Videá zobrazujúce vytvorenú aplikáciu
- **Bakalárska práca.pdf** – Dokumentácia k práci
- **Plagát.pdf** – Vytvorený plagát
- **Readme.txt** – Obsahuje informácie o práci

Príloha B

Ďalšie vytvorené objekty

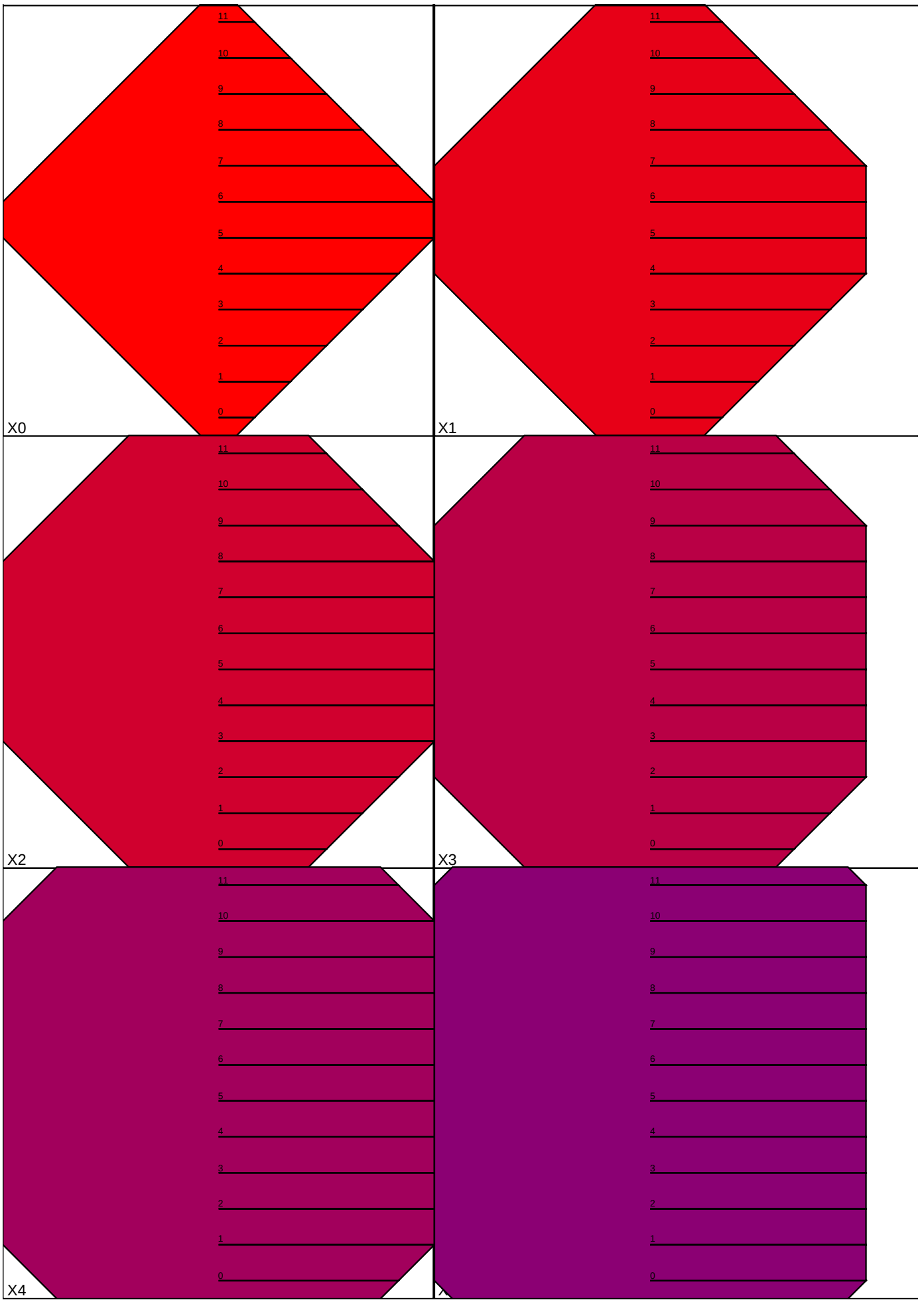


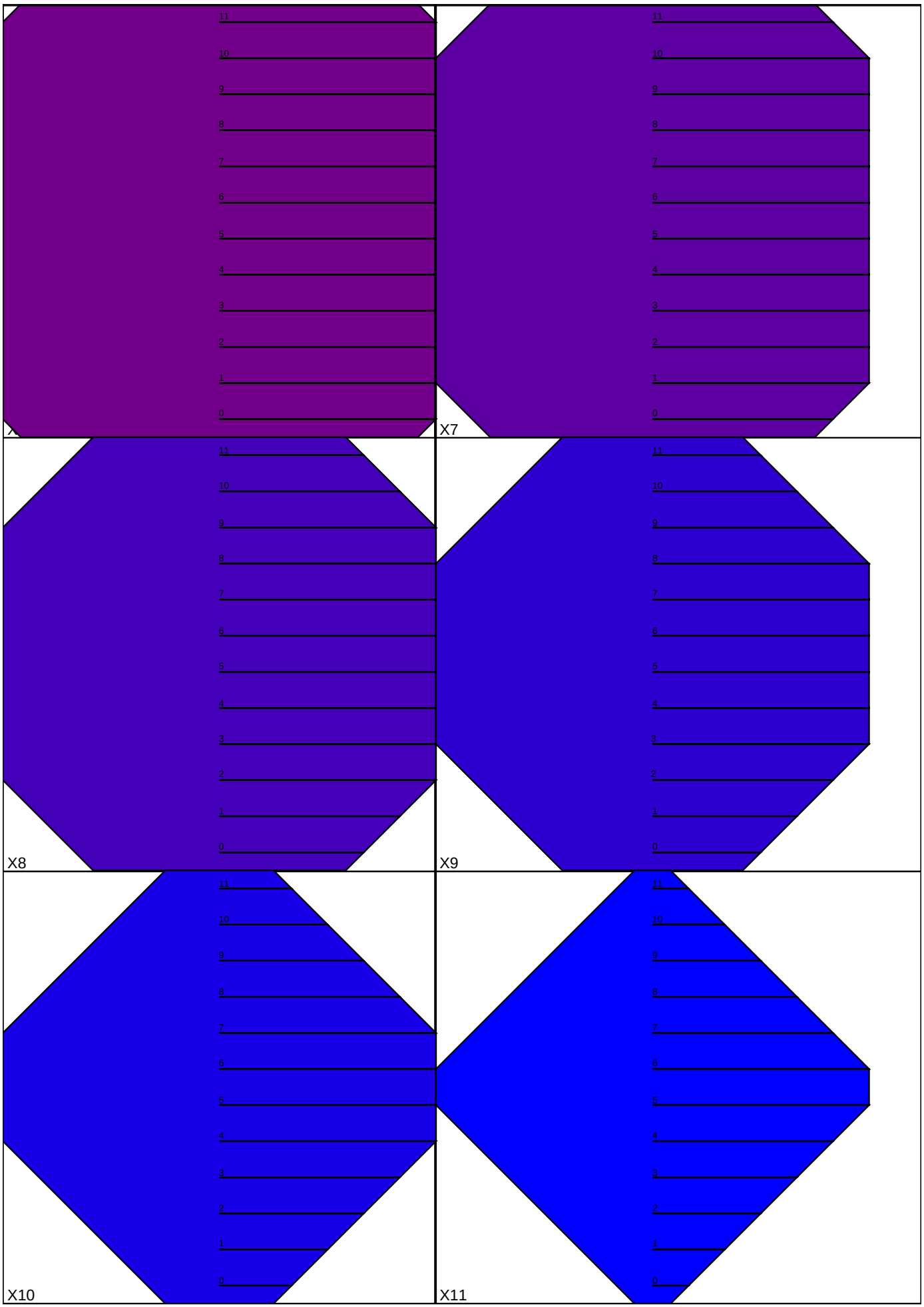
Obr. B.1: Na obrázkoch je vidieť ďalšie vytvorené objekty. Cubotaherdon, torusy, guľa, octahedron a ihlan.

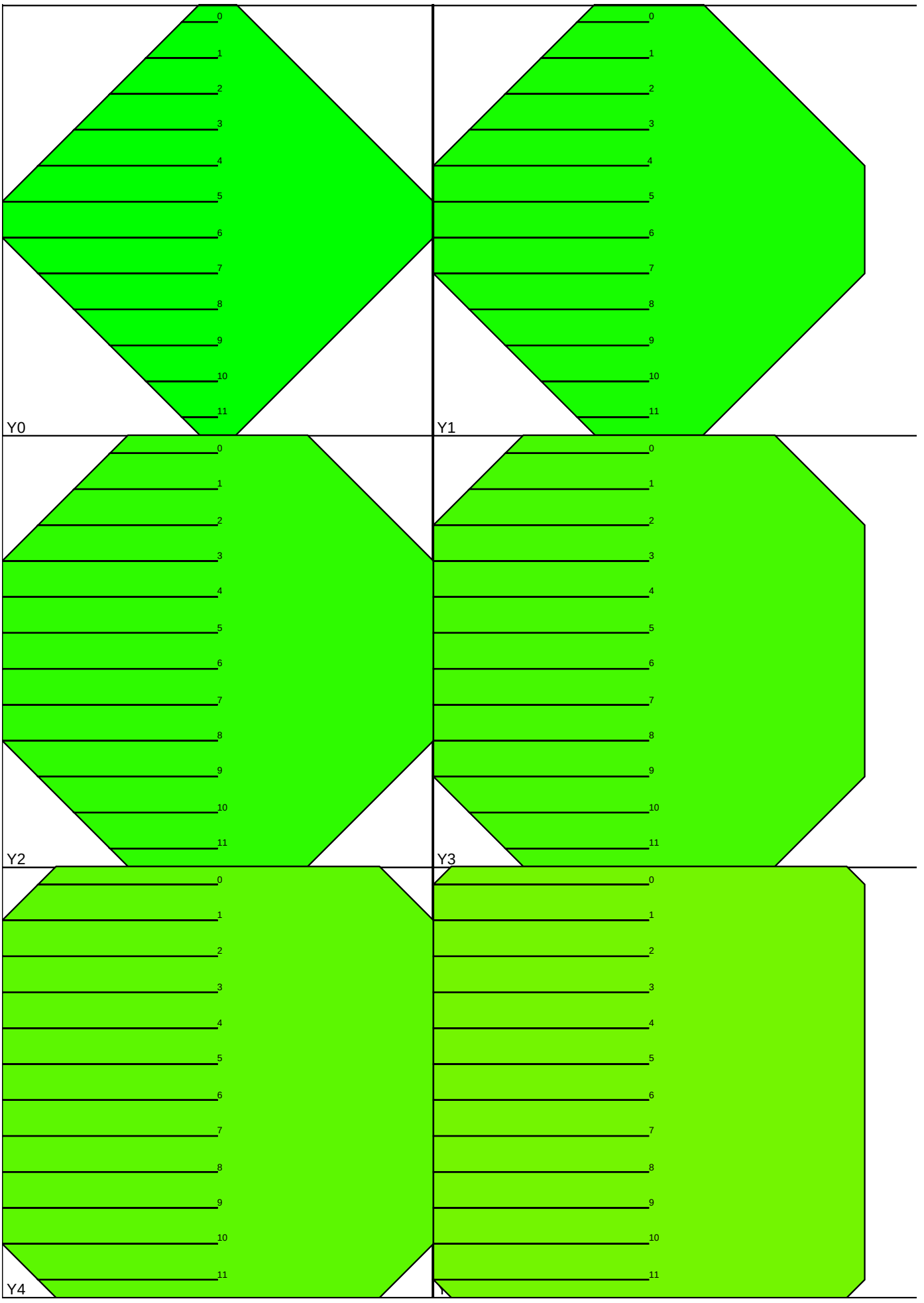
Príloha C

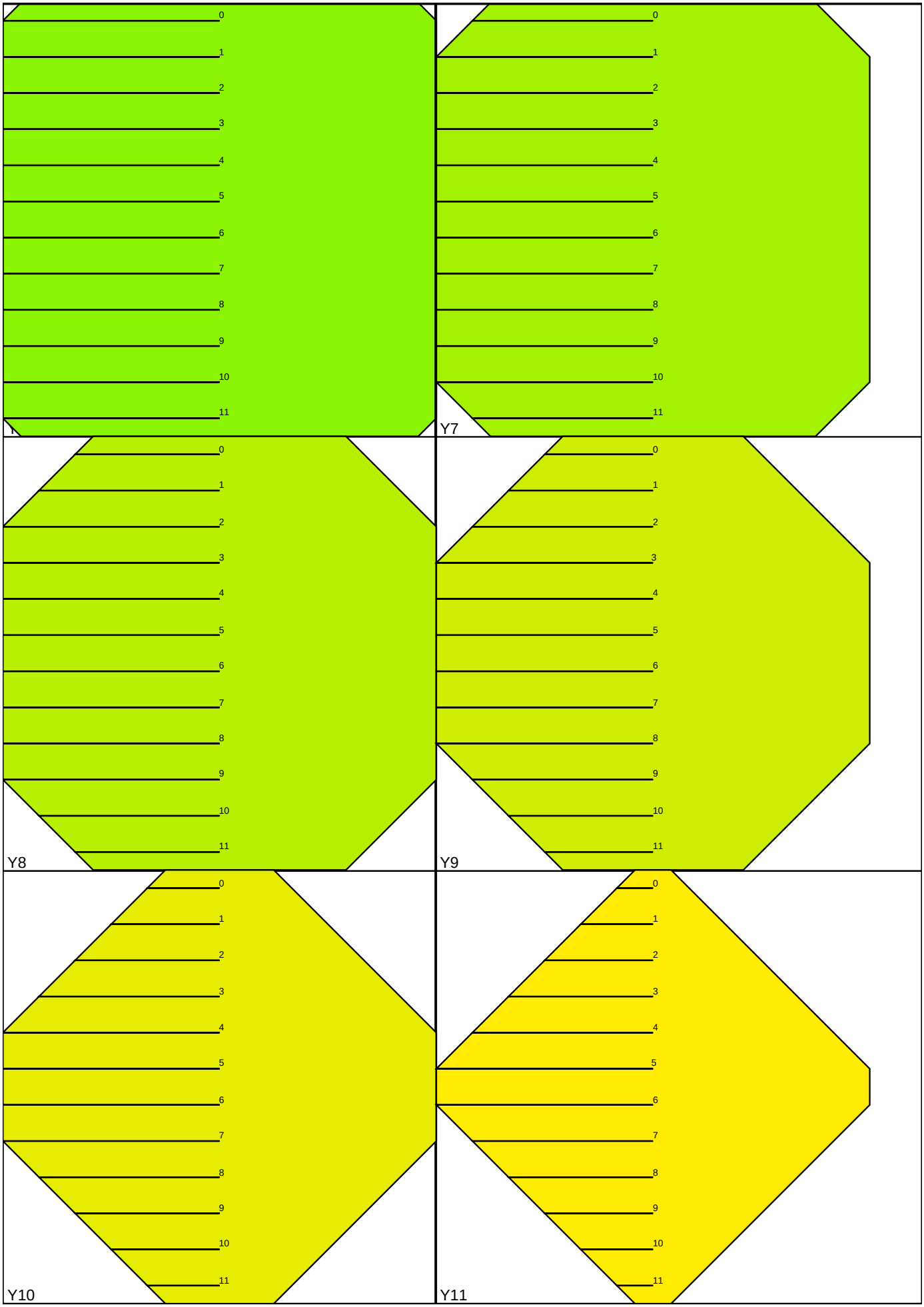
Vytvorená skladačka

Na nasledujúcich stranách je zobrazená skladačka cubotahedronu. Táto skladačka reprezentuje export z vytvoreného programu do formátu PDF s farebným vyplnením plátkov. Pomocou tejto vystrihovačky sa dá poskladať rovnaký objekt ako je vidieť na obrázku 6.3.


X0
X1
X2
X3
X4


X7
X8
X9
X10
X11


0
1
2
3
4
5
6
7
8
9
10
11
Y0
Y1
Y2
Y3
Y4


0
1
2
3
4
5
6
7
8
9
10
11
Y7
Y8
Y9
Y10
Y11