Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

3D tisk didaktických pomůcek pro výuku informatiky

2024

David Tureček

Vedoucí práce:
Mgr. Markéta Trnečková, Ph.D.

Studijní program: Informační technologie,
kombinovaná forma

Bibliografické údaje

Autor:	David Tureček
Název práce:	3D tisk didaktických pomůcek pro výuku informatiky
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2024
Studijní program:	Informační technologie, kombinovaná forma
Vedoucí práce:	Mgr. Markéta Trnečková, Ph.D.
Počet stran:	54
Přílohy:	Elektronická data v úložišti katedry informatiky
Jazyk práce:	český

Bibliographic info

Author:	David Tureček
Title:	3D printed aids for teaching coding
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2024
Study program:	Information Technologies, combined form
Supervisor:	Mgr. Markéta Trnečková, Ph.D.
Page count:	54
Supplements:	Electronic data in the storage of department of computer science
Thesis language:	Czech

Anotace

Tato bakalářská práce se zabývá tiskem 3D didaktických pomůcek pro výuku programování. Součástí práce je naprogramovaná knihovna obsahující pět základních bloků pro tvorbu kódu, které jsou inspirované vzhledem jazyka Scratch. Práce ukazuje, jakým způsobem vytvořit vlastní bloky v jazyce OpenSCAD, který je použit pro programování bloků. Dále práce obsahuje ukázkové lekce věnované základům vizuálního kódování.

Synopsis

This bachelor's thesis addresses the topic of 3D printing of didactic aids for teaching programming. It comprises a programmed library comprising five basic code blocks, inspired by the appearance of the Scratch language. It demonstrates the creation of custom blocks in the OpenSCAD language, which is used to program the blocks. Additionally, it contains sample lessons dedicated to the basics of visual coding.

Klíčová slova: 3D tisk; 3D model; vizuální programovací jazyk; OpenSCAD

Keywords: 3D printing; 3D model; visual programming language; OpenSCAD

Chtěl bych především poděkovat své vedoucí Mgr. Markétě Trnečkové, Ph.D. za trpělivost, cenné připomínky a odborné vedení mé bakalářské práce.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam zdrojových kódů

1 Úvod

3D tisk je technologie, která vznikla v 80. letech 20. století, a je využívána v různých oblastech. Setkat se s ní můžeme ve všech stupních vzdělávání, ve zdravotnictví, strojírenském průmyslu, ale také v domácím prostředí. Rozmach 3D tisku podpořil projekt „Průša pro školy“, který za zvýhodněnou cenu nabízí nejen 3D tiskárny, ale také tiskový a didaktický materiál[1]. 3D tisku se mohou věnovat i uživatelé v domácím prostředí a tisknout si výrobky dle svých zájmů a potřeb. Mezi oblasti profesionálního využití 3D tisku můžeme zařadit například zdravotnictví, kde je možné vytvářet různé chirurgické šablony, protézy a další pomůcky, které slouží pro zefektivnění práce zdravotníků a zlepšení péče o pacienty. Další využití je ve strojírenství, kde dochází ke snížení nákladů možností otestovat si prototyp před jeho zavedením do velkovýroby, ve stavebnictví a v dalších odvětvích. Výhodou 3D tisku je možnost tisknout z různých materiálů s různými vlastnostmi. Nejpoužívanějším materiálem je termoplast. V průmyslových odvětvích se lze setkat s kovem.

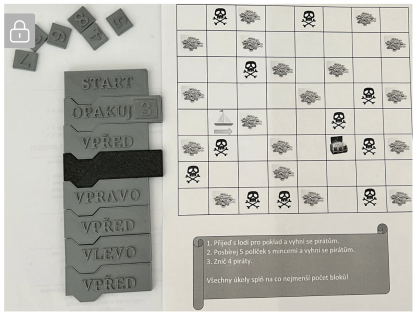
Tato bakalářská práce se zabývá využitím 3D tisku pro tisk didaktických pomůcek k výuce kódování. Základní myšlenkou, která vedla ke vzniku knihovny 3D modelů, bylo umožnit fyzicky si vyzkoušet vizuální kódování. V případě využití knihovny 3D bloků je možné představit uživatelům i možnost samotného 3D tisku a fungování 3D tiskáren. Knihovna je programovaná v jazyce OpenSCAD a obsahuje základní bloky, kterými jsou `DefaultBlock`, `StartBlock`, `ForBlock`, `IfBlock` a `ConnectionBlock`. Bloky lze exportovat do formátu STL a vytisknout na 3D tiskárně. Podoba bloků vychází z jazyka Scratch. Blokem se v této práci označuje model programovaný v jazyce OpenSCAD v programu OpenSCAD.

Druhá kapitola práce je věnována problematice 3D tisku, fázím procesu 3D tisku a především fázi slicování. Dále jsou představeny software určené pro vytváření modelů. Ve třetí kapitole je popsán software a programovací jazyk OpenSCAD a způsob, jakým se vytváří objekty. Čtvrtá kapitola se zabývá vizuálním programováním, je zde představen programovací jazyk a aplikace Scratch. V páté kapitole práce je uveden autorský postup vytváření jednotlivých bloků. Poslední kapitola práce představuje ukázky výukových lekcí pro zařazení tématu kódování a práce s 3D bloky do výuky.

1.1 Existující řešení

K řešením, která jsou na trhu dostupná, patří bloky Hands On Coding viz obrázek 1 a modely dostupné na `printables.com`. Vytištěné bloky se využívají při výuce, lze je skládat za sebe a tím vytvářet algoritmus bez použití elektronických zařízení nebo internetového připojení. Použitím této pomůcky lze prezentovat digitální prostředí ve fyzickém světě. Zároveň se jedná o interaktivní a atraktivní způsob výuky programování. Prostřednictvím konkrétní zkušenosti dochází k hlubšímu porozumění problematiky. Bloky Hands On Coding jsou vhodné pro výuku dětí od předškolního věku. Nevýhodou tohoto řešení je, že lze zakoupit pouze celé sady bloků, nikoli jednotlivé kusy. Bloky nelze dále upravovat, například není možné měnit jejich rozměry nebo na blok zadat vlastní text[2], [3]. Na webu Printables lze po registraci a s licencí Prusa Education získat například modely ukázané na obrázku 2 a 3. Nevýhodou je, že k modelům není možné získat zdrojový kód a modifikovat parametry, kterými jsou text nebo velikost. Volně dostupná ke stažení je sada 4, která vizuálně vychází z jazyka Scratch. Nevýhodou tohoto řešení je velikost For bloků, ta je omezena stanoveným počtem příkazů, které lze do bloku vložit.

Obrázek 1: Hands On Coding
[4]



Obrázek 2: Bloky pro nácvik programování
[5]

Obrázek 3: Offline pomůcky pro výuku programování
[5]

Obrázek 4: Bloky inspirované vzhledem Scratch
[5]

2 3D tisk

Tato kapitola se věnuje technologii 3D tisku, fázím procesu 3D tisku a programům slicer, které připravují model pro 3D tisk. 3D tisk byl vyvinut v 80. letech 20. století. Jedná se o technologii aditivního výrobního procesu. Na rozdíl od subtraktivního procesu, kde finální výrobek vzniká odebíráním materiálu a je produkován odpad, jsou 3D objekty vytvářeny vrstvením nanášeného materiálu a odpad vzniká minimálně[6].

Proces 3D tisku lze rozdělit na fázi modelování, slicování, tisk a dodatečné úpravy. Ve fázi modelování vzniká 3D model. 3D model lze vytvářet graficky, parametricky nebo umělecky[7]. Graficky modelovat lze například v software AutoCAD, Inventor, SolidWorks, Fusion 360, TinkerCAD. Software AutoCAD, Inventor, SolidWorks a Fusion 360 se uplatňují v oblasti strojírenství, lze v nich vytvářet 2D i 3D modely a je možné vytvářet z 3D modelů technické výkresy. Software TinkerCAD je bezplatný a vhodný pro začátečníky[8]. Pro parametrické modelování pomocí instrukcí slouží například software OpenSCAD. Software OpenSCAD je dále popsán v kapitole 3. V CAD softwarech může být obtížné modelovat objekty, které nelze popsat geometrickými primitivy, například obličej nebo postavu. Tyto objekty je potřeba modelovat pomocí modelovacích software určených pro umělecké modelování, mezi které patří například Blender. Tento software obsahuje širokou škálu sochařských nástrojů pro vytváření 3D modelů, modely je možné vytvářet celé nebo upravovat již existující[9].

Vytvořený 3D model lze ukládat například do formátů STL, OBJ, 3MF, AMF. Nejstarším a nejčastěji používaným formátem je STL[10]. Soubor popisuje pouze geometrii povrchu trojrozměrného objektu pomocí trojúhelníkových ploch, která se v souhrnu nazývá mesh[9]. Každý z trojúhelníků je definovaný svými vrcholy v trojrozměrném prostoru. Nevýhodou formátu STL je, že neobsahuje informaci o barvě, textuře, materiálu ani další metadata[11].

V průběhu fáze slicování je 3D model připraven pro tisk na 3D tiskárně. Během této fáze jsou konfigurována nastavení tisku. V programu slicer je možné například nastavit tloušťku stěny výrobku nebo procentuální hustotu výplně a její vzor. Následně je generován G-kód. Problematice slicování je dále věnovaná podkapitola 2.1 Slicer.

Ve fázi tisku je využita tisková technologie daná typem 3D tiskárny, na které tisk probíhá. 3D tiskárny se liší způsobem tisku a použitým materiálem. Technologie FFF (Fused Filament Fabrication) je založena na použití termoplastu ve formě struny. Struna je působením vysoké teploty roztavena a pomocí extrudéru tryskou vytlačována na desku, kde jsou vytvářeny jednotlivé vrstvy. Nanesením jednotlivých vrstev a následným zchladnutím vzniká výsledný objekt. FDM (Fused deposition modeling) je patentové označení používané pro tuto technologii[12], [6]. Technologie SLA (Stereolitografie), MSLA (Masked Stereolithography) a DLP (Digital Light Processing) využívají jako materiál pryskyřici a způsob tisku je založen na jejím vytvrzování vrstvy po vrstvě za pomoci světla. Technologie SLA je považována za nejstarší technologii 3D tisku. Požadovaná

oblast tisku je na podložce ozařována laserem. Technologie MSLA využívá pro ozařování oblasti tisku UV světlo LCD displeje. Technologie DLP je založena na ozařování oblasti tisku digitálním projektorem. U technologií SLS (Selective Laser Sintering), DMLS (Direct Metal Laser Sintering) a SLM (Selective Laser Melting) je jako materiál využíván prášek, který se spéká laserem, který je výkonnější než v technologii SLA[12]. Technologii SLS je možné využít pro práci s materiály jako je polyamid nebo nylon. DMLS využívá různé kovy, kterými mohou být ocel, měď, hliník, titan, slitina kobaltu a chromu nebo zlato[9]. V následující části práce se budeme zabývat pouze technologií FDM.

Při dodatečných úpravách dochází k očištění vytištěného objektu. Podpůrné části jsou odstraněny odlomením nebo odřezáním, plochy jsou následně vyhlazeny smirkovým papírem. Také je možné povrch objektu opatřit lakem, nátěrem nebo textilními vlákny[9].

2.1 Slicer

Slicer je program, prostřednictvím kterého je 3D model připraven k 3D tisku. 3D model je rozdělen na jednotlivé vrstvy. Pro každou vrstvu jsou definovány instrukce pro 3D tiskárnu, jak vrstvu vytvořit[13].

Dále lze provádět nastavení výšky vrstvy. Tato konfigurace ovlivňuje kvalitu 3D tisku. Čím nižší je vrstva, tím detailnější výrobek může vzniknout. S nastavením nižší vrstvy se ale prodlužuje doba tisku a spotřebovaný materiál[13].

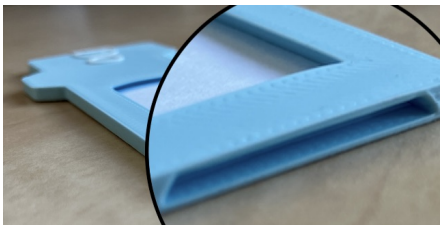
Program slicer obsahuje různé funkce. Mezi základní funkce patří například import 3D modelu, posouvání a otáčení modelu či jeho zvětšení a zmenšení nebo dělení na více částí. Je možné provádět duplikaci objektů, prostřednictvím které lze na tiskovou plochu připravit několik shodných modelů. Také lze nastavit kombinaci tisku více objektů současně[13].

Také lze konfigurovat výplň 3D modelu, označovanou jako infill, která ovlivňuje pevnost výrobku. Je možné vybrat z několika druhů výplní různého tvaru například plástev, mřížka, čára, hvězda a další[13]. Dále lze nastavit procentuální hodnotu hustoty výplně.

„Jelikož je objekt vytvářen vrstvu po vrstvě, je potřeba, aby každá vrstva byla podpírána vrstvou předchozí. Proto je u některých modelů nutné použít podpěry. Podpěry je potřeba generovat v místech, kde předchozí vrstva dostatečně nepodpírá následující vrstvu a to u převisů, mostů a u ostrovů.“[9] Grafické znázornění převisu, mostu a ostrova je na obrázku 5. Zobrazení vytištěného mostu je na obrázku 6. „Ne u všech převisů je potřeba podpěra. Existuje pravidlo 45°. Pro úhel 45° platí, že polovina nové vrstvy leží na předchozí vrstvě, tedy má dostatečnou oporu.“[9] Tiskem mezi dvěma opěrnými body bez použití podpěr vzniká most. Kvalita tisku mostu závisí na vzdálenosti mezi dvěma pevnými body, rychlosti posunu trysky, průtoku filamentu tryskou, teplotě a chlazení filamentu[14], [15], [16].

Existují různé druhy programu slicer, mezi které patří například UltiMaker Cura, Simplify3D nebo PrusaSlicer[17], které se liší počtem podporovaných tis-

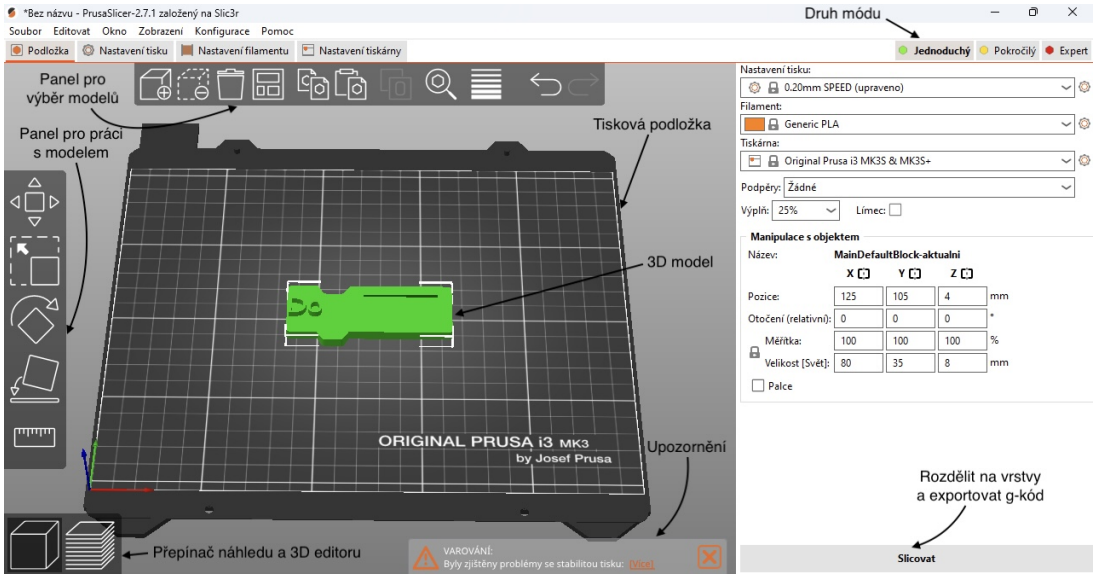
Obrázek 5: Znázornění převisu, mostu a ostrova
[9]



Obrázek 6: Vytištěný most bez podpěr

káren, rozšířením v uživatelské komunitě a v dostupnosti (zdarma nebo placená licence).

V této práci byl použit PrusaSlicer verze 2.7.1, který podporuje tiskárny Original Prusa a další typy, v uživatelské komunitě je rozšířen, má podporu a aktualizace od vydavatele a je dostupný zdarma[17]. PrusaSlicer vychází z Slic3r, což je volně licencovaný program z roku 2011. Slic3r Prusa Edition byla vydána v roce 2016[6]. Pro zajištění kompatibility je nástroj PrusaSlicer dlouhodobě vyvíjen firmou Prusa Research. Kompatibilitu s vlastními 3D tiskárnami firma deklaruje a na svých tiskárnách testuje[18]. Náhled PrusaSliceru verze 2.7.1 viz obrázek 7.

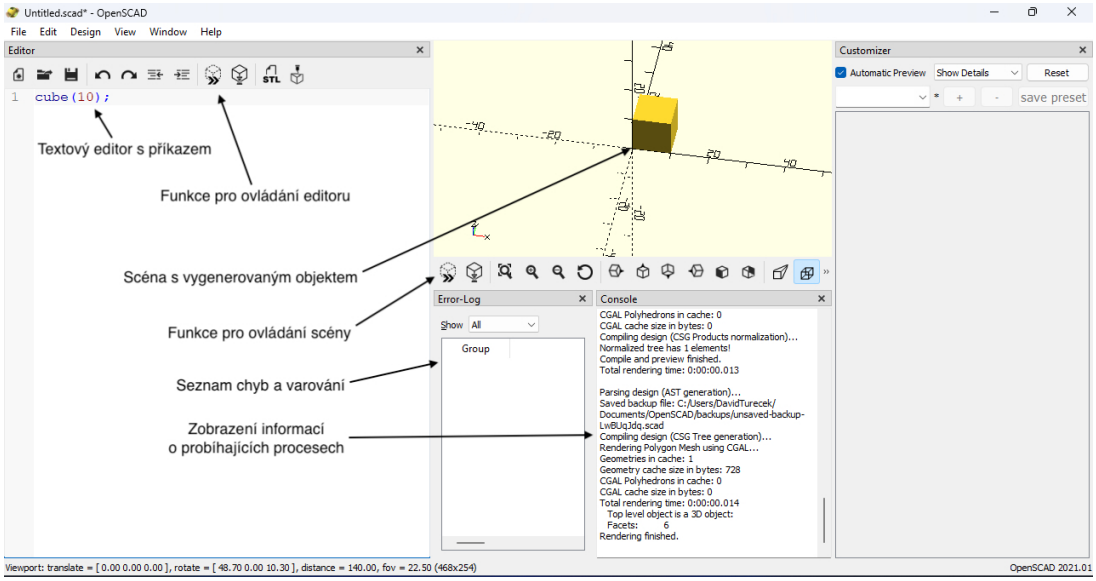


Obrázek 7: PrusaSlicer vzhled sliceru verze 2.7.1

3 OpenSCAD

Tato kapitola se věnuje programovacímu jazyku OpenSCAD a překladači, který je součástí vývojového prostředí se stejným názvem. Dále je v této kapitole vysvětlen pojem konstruktivní geometrie a je popsáno vytváření modelu pomocí geometrických primitiv, množinových operací a transformací objektů. V podkapitole 3.1 je popsáno vytváření objektů v OpenSCAD.

OpenSCAD je parametrický open-source software pro tvorbu 3D modelů skriptováním. Umožňuje parametrické modelování, které usnadňuje iterativní návrh úprav. Omezené grafické rozhraní neumožňuje uživatelům bez znalosti skriptování vytvářet objekty a použití může být obtížnější než práce s vizuálně orientovanými 3D modelovacími programy. OpenSCAD je dostupný pro operační systémy Windows, macOS a Linux[19]. Vzhled programu OpenSCAD viz obrázek 8.



Obrázek 8: OpenSCAD vzhled programu

Pro vytváření modelů je využíván princip konstruktivní geometrie. Konstruktivní geometrie jako součást matematiky vychází z deskriptivní geometrie. Jako první definoval pojem konstruktivní geometrie roku 1955 profesor Erwin Krupp. Základní myšlenkou konstruktivní geometrie je, že složité objekty lze vytvořit sjednocením, průnikem nebo rozdílem jednodušších geometrických tvarů. Konstruktivní geometrie jako matematický a počítačový modelovací přístup se využívá v 3D grafice a 3D modelování[20].

Modely jsou popisovány jazykem OpenSCAD. V něm je možné používat geometrická primitiva, moduly, podmínky a cykly. Lze také využívat zadání proměnných, vstupních parametrů, import knihoven nebo importování externích souborů.

3.1 Vytváření objektů v software OpenSCAD

Podkapitola se zaměřuje na tvorbu objektů v prostředí OpenSCAD. Popisuje základní principy vytváření objektů pomocí proměnných a příkazů. Dále představuje geometrická primitiva, jejich transformace a množinové operace a extruze, které umožňují vytvářet složitější geometrické tvary. Také se zabývá moduly, funkcemi a možností integrace externích souborů a knihoven. V závěru uvádí informace o řízení toku kódu pomocí cyklů a podmínek. Text této podkapitoly vychází z[21], [22] a [23]. K jazyku OpenSCAD je k dispozici dokumentace[22]. OpenSCAD pracuje s hodnotami, což mohou být čísla, řetězce, vektory a jiné. Hodnoty můžeme pojmenovat pomocí proměnných, stejně jako v jiných programovacích jazycích. Ty se definují tak, jak je ukázáno ve zdrojovém kódu 1,

```
1 a=10;
2 echo(a) //10
```

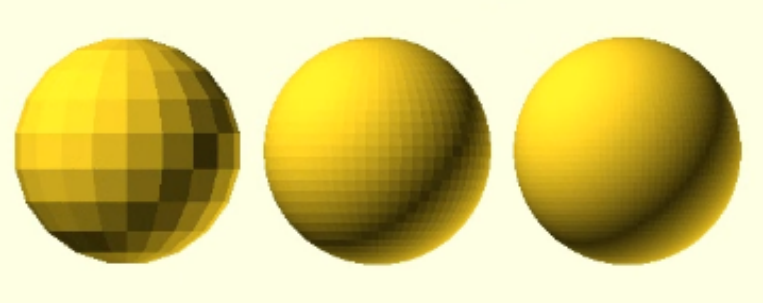
Zdrojový kód 1: OpenSCAD proměnné

kde `a` představuje název proměnné, které je operátorem = přiřazená hodnota 10. `echo` je funkce pro výpis hodnot v závorkách. Znaky `//` a `/**/` uvozují v OpenSCAD komentáře, prostřednictvím kterých je daná část kódu překladačem ignorována. OpenSCAD je deklarativní jazyk, takže proměnná má v celém programu stejnou hodnotu a to tu, která jí byla přiřazena jako poslední[24]. Pro práci v OpenSCAD se používají příkazy. Jednotlivé příkazy se oddělují středníkem (`;`). Příkazy se skládají ze jména, které většinou odpovídá názvu objektu. Za jménem následují parametry, které určují vlastnosti objektu a píší se do kulatých závorek přímo za název příkazu: `objekt (parametry);`. Parametry lze nastavovat a měnit tak například velikost objektu.

Geometrická primitiva jsou jednoduché geometrické tvary, ze kterých se pomocí množinových operací a prostorových transformací vytváří výsledný objekt. Mezi základní 3D geometrická primitiva patří například koule, krychle, válec a mnohostěn.

Pro vytvoření koule se používá příkaz `sphere()`. Její velikost lze specifikovat zadáním poloměru (parametr `r`) nebo průměru (parametr `d`). Koule se vytvoří vždy se středem v počátku souřadného systému. Přesnost vytvořené koule lze ovlivnit parametry `$fa`, `$fs` a `$fn`. Parametr `$fa` představuje minimální úhel každého fragmentu, `$fs` minimální délku oblouku a `$fn` počet fragmentů v 360°. Čím vyšší je hodnota parametru, tím je větší přesnost. Vždy se ale specifikuje jen jeden z těchto parametrů. Výsledek použití parametru `$fn` viz obrázek 9.

Krychle nebo kvádr se vytváří pomocí příkazu `cube()`, kde v kulatých závorkách je třeba specifikovat velikost pomocí parametru `size` a také pozici, kde se má těleso vytvořit pomocí parametru `center`. Parametr `size` může být zadán jedním číslem, pak se vytvoří krychle o zadané délce stran, nebo trojicí čísel v hranatých závorkách `[x, y, z]`, pak se vytvoří kvádr s délkami stran

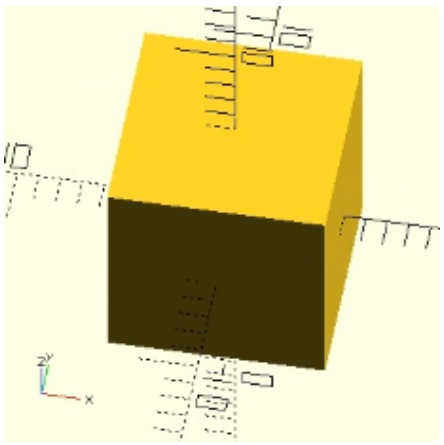


Obrázek 9: OpenSCAD různé nastavení parametru `$fn`

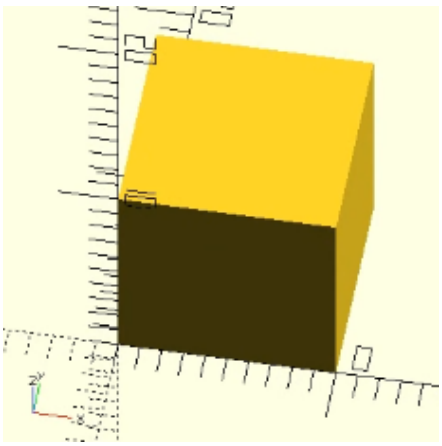
x, y, z. Pokud je parametr `center` nastaven na `true`, střed vytvořeného tělesa leží v počátku souřadného systému. V případě nastavení na `false` leží v počátku souřadného systému jeho vrchol. Například vytvoření krychle o velikosti 10 lze provést následujícím způsobem viz ukázka zdrojového kódu 2. Výsledek při nastavení parametru `centre=true` viz obrázek 10, při nastavení parametru `centre=false` výsledek viz obrázek 11.

```
1 cube(size=10, center=true);
```

Zdrojový kód 2: OpenSCAD `cube`



Obrázek 10: OpenSCAD krychle s parametrem `center=true`



Obrázek 11: OpenSCAD krychle s parametrem `center=false`

Válec se vytváří příkazem `cylinder()` a je nutné specifikovat jeho výšku pomocí parametru `h` a poloměr nebo průměr jeho podstavy pomocí parametru `r`, respektive `d`. Pokud chceme specifikovat poloměr každé podstavy zvlášť, použijeme místo parametru `r` parametry `r1` a `r2`. Pozici válce specifikuje parametr `center`. Přesnost kruhové podstavy válce specifikují parametry `$fa`, `$fs` a `$fn`, stejně jako u koule[24].

Mnohostěn se vytváří příkazem `polyhedron()` a jde o nejobecnější geometrické primitivum. Lze jej využít k vytváření pravidelných i nepravidelných tvarů. Mnohostěn je zadán parametrem `points`, což je seznam vrcholů a parametrem `faces`, což je seznam mnohoúhelníků, které tvoří jeho povrch. Množina vrcholů je výčet vrcholů v hranatých závorkách. Každý vrchol je zadán trojicí `[x, y, z]`, která představuje souřadnice v 3D prostoru. Vrcholy jsou v množině očíslovány od 0 do n-1, kde n je počet vrcholů ve výčtu. Toto pořadí se označuje jako index. Každý mnohoúhelník je zadán n-ticí vrcholů `[v1, v2, v3, ...]`. Indexy vrcholů `v1, v2, v3, ...`, odkazují na vrcholy v množině `points`. Při zadávání parametru `faces` je nutné zadávat vrcholy n-úhelníku při pohledu zvenku proti směru hodinových ručiček. Aby bylo možné objekty zkonstruovat, musí být uzavřené a nesmí obsahovat dotýkající se stěny, hrany a vrcholy.

Polohu a velikost vytvořených geometrických primitiv lze ovlivnit pomocí transformací. Mezi základní transformace patří změna velikosti, otočení, posunutí a zrcadlení. Velikost objektu se mění pomocí příkazu `scale()`, který mění velikost objektu nebo množiny objektů ve složených závorkách o zadanou hodnotu nebo vektor tří hodnot reprezentujících zvětšení nebo zmenšení v jednotlivých osách. Další možností je využít příkaz `resize()`, který změní velikost objektu na konkrétní hodnotu. Příkaz `rotate()` se užívá k otočení. Objekty je možné otáčet kolem všech os o stejný úhel nebo okolo každé osy různě. Úhel se zadává ve stupních. Otáčení probíhá podle pravidla pravé ruky. Pokud palec ukazuje v kladném směru osy otáčení, pokrčené prsty ukazují směr. Pro posun objektu se používá příkaz `translate()`, kdy se udává posun po ose `x`, `y` a `z`. Příkaz `mirror()` slouží k zrcadlení objektu. Zadáním bodu v prostoru se určí přímka, která vede k počátku souřadného systému. Objekt se pak zrcadlí podle roviny, která je kolmá k této přímce. Jednotlivé transformace lze skládat. V tomto případě se transformace oddělují mezerou. Operace se vykonávají v opačném pořadí, než jsou uvedeny (čili zprava doleva). Následující příklad ukazuje posunutí o 10 v ose `x`, rotaci o 45° v ose `z` a 2× zvětšení na celý objekt kvádru o rozměrech 10, 20, 30 viz ukázka zdrojového kódu 3, výsledek viz obrázek 12.

```
1 translate([10, 0, 0])
2 rotate([0, 0, 45])
3 scale([2, 2, 2])
4 cube([10, 20, 30]);
```

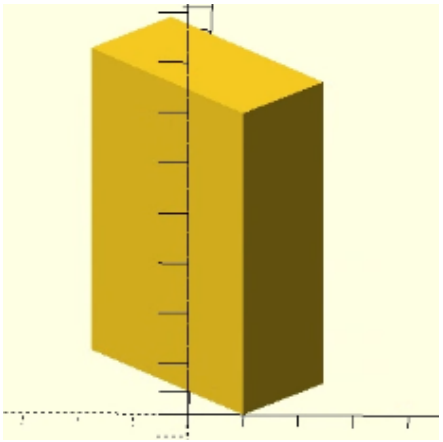
Zdrojový kód 3: OpenSCAD transformace

Množinové operace umožňují kombinovat objem více těles dohromady a vytvářet složitější tvary. Mezi tyto operace patří sjednocení, rozdíl a průnik. Sjednocením dvou množin získáme množinu patřící tělesu, která obsahuje všechny prvky z obou množin. Pro sjednocení se používá příkaz `union()`, který se aplikuje na objekty uvedené ve složených závorkách. Příklad použití množi-

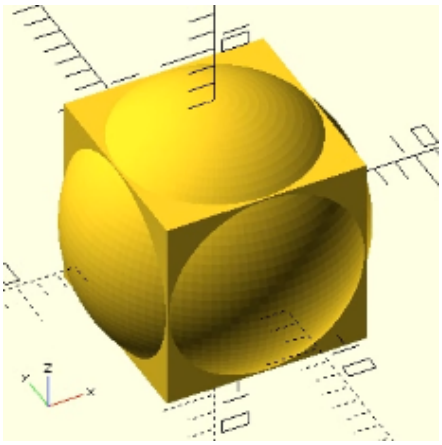
nové operace `union` viz ukázka zdrojového kódu 4 a výsledek viz obrázek 13. Rozdílem dvou množin rozumíme všechny prvky, které patří do první množiny, ale nepatří do druhé. Pro rozdíl se používá příkaz `difference()`. Tento příkaz se aplikuje tak, že od prvního objektu ve složených závorkách se odečítají všechny ostatní z výčtu. Průnik dvou množin je množina společných prvků, tedy prvků, které patří do obou množin. V OpenSCAD se pro průnik používá příkaz `intersection()`. Tímto způsobem lze efektivně využívat množinové operace k vytváření komplexních tvarů v OpenSCAD.

```
1 union() {
2     cube(10, true);
3     sphere(7);
4 }
```

Zdrojový kód 4: OpenSCAD `union`



Obrázek 12: OpenSCAD transformace



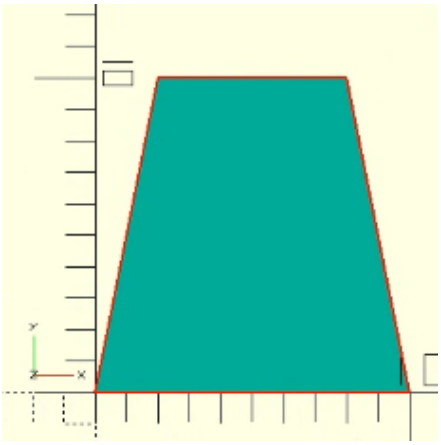
Obrázek 13: OpenSCAD množinová operace `union`

3D objekty je možné vytvářet také z 2D objektů pomocí extruze. Dvourozměrné objekty jsou nekonečně tenké a vytváří se v rovině os `x` a `y`. Nejobecnějším 2D objektem je mnohoúhelník, mezi další 2D objekty patří čtverec a kruh. Mnohoúhelník se vytváří příkazem `polygon()`. Jako parametry se mu předávají vrcholy (`points`) zadané dvojicí `[x, y]` a cesty (`paths`). Viz ukázka zdrojového kódu 5, výsledek viz obrázek 14. Cesty jsou sekvence indexů, které odkazují do množiny bodů představující hranice mnohoúhelníku. Čtverec se vytváří pomocí příkazu `square()`. Parametry jsou `size` a `center`, jako u krychle. Kruh se vytváří příkazem `circle()`. Parametry jsou poloměr (`r`) nebo průměr (`d`). Přesnost lze ovlivnit parametry `$fa`, `$fs` a `$fn`, podobně jako u koule. Mezi 2D objekty patří také objekt text, který se vytváří příkazem `text()`, kdy parametr `text` udává text, který má být zobrazen. Na 2D objekty lze aplikovat transformace a množinové operace podobně jako na 3D objekty.

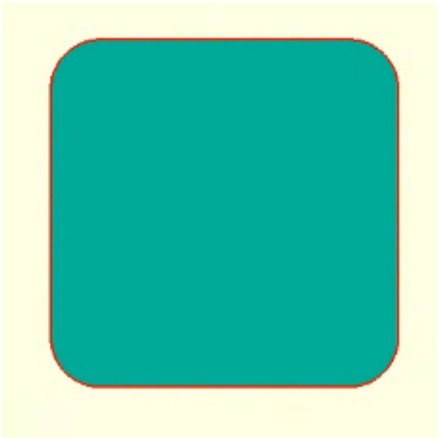
```
1 polygon([[0,0],[10,0],[8,10],[2,10]]);
```

Zdrojový kód 5: OpenSCAD `polygon`

Ke změně velikosti či zaoblení dvourozměrných tvarů lze použít operaci `offset()`. Příkaz offset vytváří nový 2D vnitřní nebo vnější obrys ze stávajícího obrysu a má dva parametry: `r` nebo `delta`. Oba parametry určují, o kolik se má objekt zvětšit (kladné číslo) nebo zmenšit (záporné číslo). Rozdíl mezi parametry je v tom, zda budou rohy výsledného objektu zakulacené (nastavíme parametr `r`) nebo ostré (nastavíme parametr `delta`). Výsledek použití `offsetu` viz obrázek 15.



Obrázek 14: OpenSCAD `polygon`



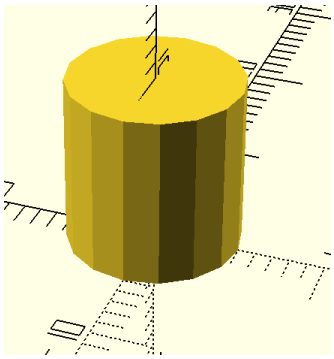
Obrázek 15: OpenSCAD zobrazení offsetu

Dvourozměrné objekty samy o sobě nelze použít při konstrukci 3D modelů. Nejdříve se z nich musí vytvořit 3D objekty pomocí extruze. V OpenSCAD existuje extruze lineární a rotační. Lineární extruze „vytáhne“ 2D objekt do prostoru a vytvoří tak 3D objekt, viz ukázka zdrojového kódu 6 a výsledek viz obrázek 16. Provádí se příkazem `linear_extrude()`. Tento příkaz má parametr `height`, který určuje výšku a jeho hodnota musí být kladná. Lze využít řadu dalších parametrů. Parametr `scale` změní měřítko horní podstavy v zadaném poměru. Extruze se sbíhá směrem k ose `z` (nebo se rozbíhá od osy `z`). Parametr `twist` určuje počet stupňů, o kolik se objekt během extruze otočí kolem osy `z` po směru hodinových ručiček. Pro opačný směr otáčení se zadává záporná hodnota. Dalším parametrem je `center`, který určuje, kde bude výsledné těleso mít střed. Parametr `slice` určuje počet vodorovných řezů. Lze využít také `$fn`, který určuje počet segmentů použitých v oblouku a kruhu ve 2D objektu[24].

Druhým typem extruze je extruze rotační, pro kterou se využívá příkaz `rotate_extrude()`. V tomto případě je 3D objekt vytvořen rotací 2D profilu kolem osy `z`. Původní 2D tvar se nachází v rovině XY. Rotace kolem osy `z`

```
linear_extrude(height=10) {
    circle(r=5);
}
```

Zdrojový kód 6: OpenSCAD `linear_extrude`



Obrázek 16: OpenSCAD `linear_extrude`

o specifikovaný úhel generuje výsledný 3D objekt. O kolik stupňů se má objekt otočit určuje parametr `angle`. Objekt se otáčí po směru hodinových ručiček. Pokud se parametr `angle` nezadá, otočí se o 360°. Přesnost vykreslení lze ovlivnit parametrem `$fn`[24].

Pro vytvoření 2D objektu z 3D objektu slouží modul `projection()`, který promítne 3D objekt na rovinu XY. Při nastavení parametru `cut=true` vrátí projekce řez rovinou XY, viz ukázka zdrojového kódu 7, výsledek viz obrázek 17.

```
projection()
    cube(10);
```

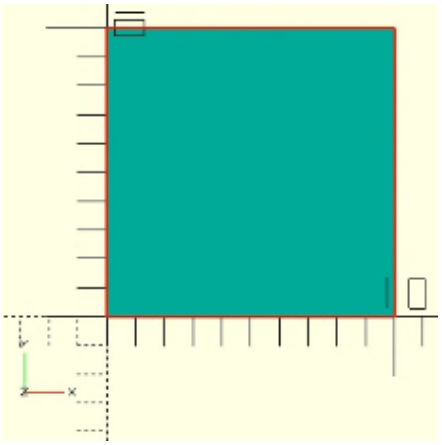
Zdrojový kód 7: OpenSCAD `projection`

Základními stavebními bloky pro tvorbu 3D modelů jsou v OpenSCAD moduly a funkce. Moduly umožňují organizovat kód do opakovaně použitelných bloků a zjednodušit tak tvorbu složitějších těles. Kromě již zmíněných předdefinovaných modulů například `cube()`, `cylindre()`, `polyhedron()`, `circle()` je možné vytvářet vlastní (viz ukázka zdrojového kódu 8). Modul se definuje pomocí klíčového slova `module`, za kterým následuje identifikátor (název modulu). V kulatých závorkách se následně uvedou parametry, které modul přijímá. Ve složených závorkách je definice modulu. Výsledek viz obrázek 18. Vlastní moduly mohou obsahovat podmíněné výrazy, cykly a proměnné. Tyto moduly mohou zahrnovat libovolné kombinace geometrických primitiv, booleovských operací a transformací objektů[24].

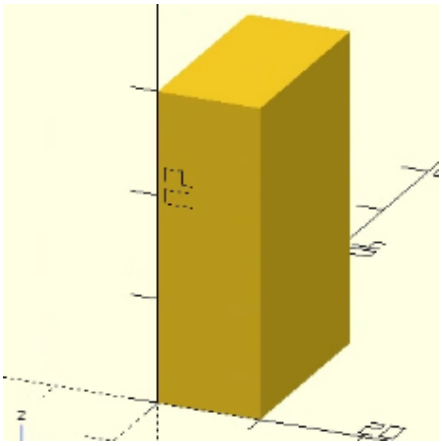
Funkce neslouží k tvorbě objektů, ale vypočítají hodnotu na základě para-

```
module VlastniKostka(velikost) {
    cube(velikost);
}
VlastniKostka([10,20,30]);
```

Zdrojový kód 8: OpenSCAD `module`



Obrázek 17: OpenSCAD `projection`



Obrázek 18: OpenSCAD `VlastniKostka([10,20,30])`

metrů. Funkce se vytváří pomocí klíčového slova `function`, za kterým opět následuje identifikátor s parametry dané funkce. Tělo funkce se zapisuje do složených závorek[24]. Viz ukázka zdrojového kódu 9.

```
function NazevFunkce(parametr1, parametr2, ...) =
    // Tělo funkce zakončené středníkem ;
```

Zdrojový kód 9: OpenSCAD `function`

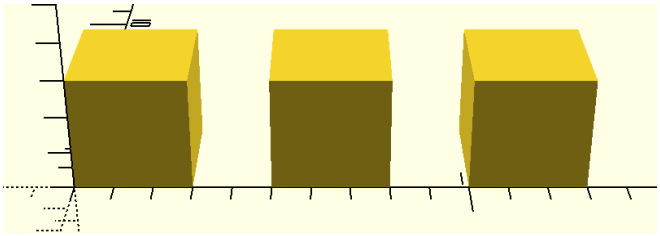
Pro integraci externích souborů a knihoven se v OpenSCAD užívají moduly `import()` a `use()`. Modul `import()` vkládá do CSG stromu STL a DXF soubory. Argumentem je relativní nebo absolutní cesta k souboru. Modul načte a vykoná kód obsažený ve specifikovaném souboru. Modul `use()` slouží k použití definovaného modulu nebo funkce z jiného souboru. Tento modul umožňuje přístup k funkcím a modulům definovaným v jiném souboru, aniž by bylo nutné importovat celý soubor[24].

Pro řízení toku kódu je možné v OpenSCAD využít cykly a podmínky. Cyklus `for` se zadává následovně viz ukázka části zdrojového kódu 10, výsledek viz obrázek 19. V inicializaci cyklu se do proměnné přiřadí vektor hodnot. V těle cyklu pak proměnná z inicializace cyklu nabývá hodnot z předaného vektoru. Tělo cyklu se zapisuje do složených závorek. For cyklus vytvoří v každé iteraci

objekt. Tyto objekty se následně sjednotí.

```
1     for (i = [0 : pocet_krychli - 1]) {
2        translate([i * posunuti_x, i * posunuti_y, i * posunuti_z]) {
3           cube();
4        }
5     }
```

Zdrojový kód 10: OpenSCAD `For cyklus`



Obrázek 19: For cyklus, krokové zvětšení

OpenSCAD obsahuje dva druhy podmínek: `if` a ternární operátor. Podmínka `if` umožňuje vykonávat různé části kódu podle platnosti podmínky. Používají se klíčová slova `if` viz ukázka zdrojového kódu 11, `else` a `if else`.

```
1  if (podmínka) {
2     //kód
3  }
```

Zdrojový kód 11: OpenSCAD If podmínka

Ternární operátor například nastaví výšku na 20, pokud je x větší než 5, jinak na 10, viz ukázka zdrojového kódu 12.

```
x = 10;
vyska = (x > 5) ? 20 : 10;
cube([10, 10, vyska]);
```

Zdrojový kód 12: OpenSCAD ternární operátor

4 Vizuální programování

Tato kapitola se zaměřuje na vizuální programování a představuje některé vybrané programovací jazyky. Vizuální programování je styl programování, který využívá grafické uživatelské rozhraní a vizuální prvky. Tyto formy programování, známé jako vizuální programovací jazyky, lze rozdělit do různých kategorií, jako je blokové kódování (např. Scratch, Kodu Game Lab), diagramové nebo data-flow jazyky a další[25].

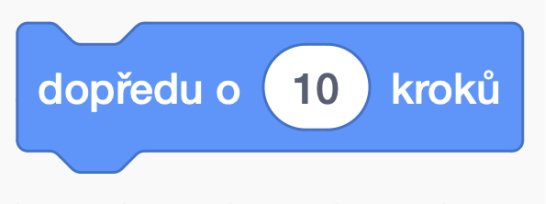
V blokových programovacích jazycích jsou základními prvky bloky, které symbolizují jednotlivé funkce, podmínky nebo operace. Uživatelé je umisťují do vývojového prostředí. Program vzniká jako stavebnice, kdy se jednotlivé bloky spojují dohromady[25]. Umisťování a manipulace s bloky je často prováděna pomocí drag-and-drop. Pokud grafiku bloku nahradíme textovým ekvivalentem, mohou blokové jazyky připomínat textový kód. Jejich výhodou oproti textovému kódu je přiblížení se fyzickému světu a prevence chyb v syntaxi, například při používání středníků, odsazení tabulátory nebo rozlišování velkých a malých písmen. Organizace do bloků a zajištění správné návaznosti prostřednictvím tvarů bloků zajišťuje syntaktickou správnost, avšak nezaručuje smysluplnost programu. Vizuální podoba kódu může být přístupnější i při výuce dospělých, kteří se s koncepty programování dosud nesetkali[25].

Další kategorií jsou diagramové, neboli data-flow programovací jazyky. Pro tyto jazyky je charakteristické propojování vizuálních prvků čárami nebo šipkami. Diagramové jazyky připomínají vývojové nebo jiné diagramy, které znázorňují závislosti mezi jednotlivými komponentami programu. Jednou ze silných stránek diagramových jazyků je jejich vizualizační schopnost, neboť je lze často využít jako náhled do aktuálně běžícího systému. Poskytují také možnost přeskočení fáze nákresů a diagramů při vývoji, jelikož místo nákresu je možné vytvořit základ programu, který již představuje začátek spustitelného programu[25].

Ačkoliv vizuální programovací jazyky nabízejí zřetelné výhody, mají také své omezení. Ve srovnání s tradičním textovým programováním je jejich využití omezené při tvorbě rozsáhlejších programů nebo řešení složitějších úkolů.

Mezi výukové vizuální programovací jazyky se řadí například Baltík, Scratch, Kodu Game Lab a další. Baltík je programovací jazyk vyvinutý v roce 1996 českou společností SGP Systems a stal se inspirací pro další vizuální programovací jazyky, například Scratch. Je určený pro výuku programování na základních a středních školách. Program využívá programování pomocí obrázkových ikon, které nahrazují kód. Jeho syntaxe je shodná jako v textovém programovacím jazyce Pascal a C[26], [27].

Scratch je programovací jazyk a webová i mobilní aplikace, která vznikla v roce 2003 v USA. Nabízí bohaté vzdělávací prostředí pro všechny věkové kategorie. Jako učební pomůcka poskytuje propracované instruktážní návody. Dále nabízí instrukce pro tvorbu vlastní postavy, kterou lze animovat, nebo pro změnu pozadí. Vzhled bloků ve Scratch sloužil jako inspirace pro tvorbu bloků v této bakalářské práci. Na obrázku 20 je ukázka základního bloku.



Obrázek 20: Scratch základní blok

Dalším vizuálním programovacím jazykem je Kodu Game Lab. Ten slouží pro vytváření her i bez znalosti designu a programování. Je tak velmi přínosný k rozvoji kreativity, řešení problémů a dovednosti vyprávění příběhů prostřednictvím tvorby herního děje[28].

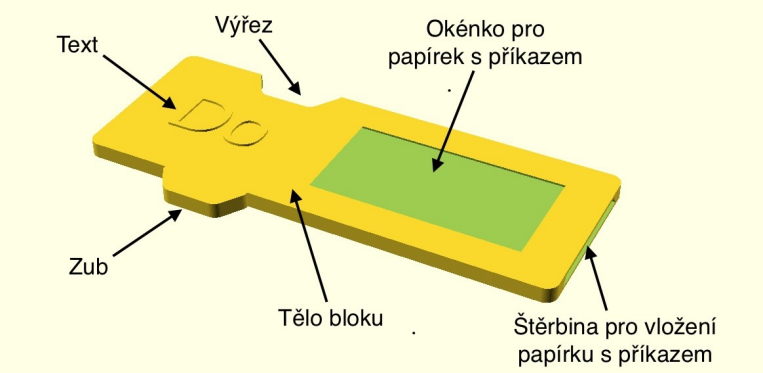
5 Programátorská dokumentace

V této kapitole je popsán proces tvorby bloků. Jsou zde popsány jednotlivé bloky a kroky jejich modelování. Dále je diskutováno, jakým způsobem byly bloky tištěny.

V rámci práce byly vytvořeny následující bloky:

- `DefaultBlock`,
- `StartBlock`,
- `ConnectionBlock`,
- `ForBlock`,
- `IfBlock`.

Bloky se skládají z několika modulů tak, aby vytvořily požadovaný celek. Například `DefaultBlock` je složen z šesti částí: tělo bloku, zub, výřez, okénko pro papírek s příkazem, štěrbina pro vložení papírku s příkazem a text viz obrázek 21.



Obrázek 21: Popis částí `DefaultBlock`

U každého bloku je možné nastavit jeho parametry. Například `DefaultBlock` má následující parametry:

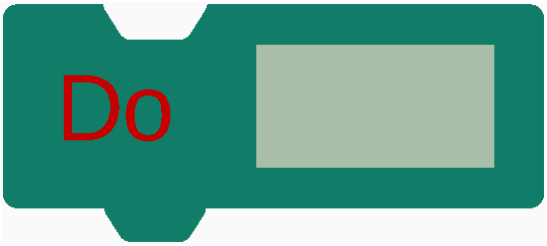
- `height_block` slouží k nastavení šířky bloku,
- `lenght_arry_box` nastavuje délku okénka pro papírek s příkazem,
- `main_text` umožňuje měnit text na bloku,
- `paper_hole` nastavuje, zda bude mít blok okénko pro papírek s příkazem.

Nastavení velikosti bloku má několik kritérií, která je vhodné zohlednit. Menší bloky mají nižší spotřebu materiálu na tisk a jsou vhodné pro výuku a bádání jednotlivce. Velké bloky jsou doporučeny pro skupinovou výuku nebo prezentování na tabuli. Velikost je vhodné konfigurovat i dle technických parametrů 3D

tiskárny. Velikost bloků programovaných v této práci je určena pro individuální použití tak, aby se s bloky příjemně pracovalo a daly se vytisknout na běžných 3D tiskárnách s optimální spotřebou materiálu. Výška bloku je nastavena tak, aby byla dělitelná 0.2, protože hodnota 0.2 je doporučena jako výchozí výška vrstvy při tisku. Defaultně je nastavena výška bloku na 2.6, což umožňuje vytisknout kvalitní bloky, které nevyžadují další zásadní úpravy. Výška textu je 0.4, což odpovídá dvěma vrstvám. Výška textu je zvolena s cílem zajistit jeho optimální čitelnost.

5.1 DefaultBlock

Modul `DefaultBlock` slouží k vytváření základního bloku se specifikovanými rozměry, který zahrnuje tělo bloku, zub, výřez, okénko pro papírek s příkazem, štěrbinu pro vložení papírku s příkazem a text. Parametry modulu umožňují nastavení délky, výšky a šířky bloku, stejně jako okénka pro papírek s příkazem. Ukázka bloku je uvedena na obrázku 22.



Obrázek 22: OpenSCAD `DefaultBlock`

Použité moduly

K vytvoření byly využity moduly `MainPart`, `MainTooth`, `PaperHole`, `TextUniversal`.

Modul `MainPart` je tělo bloku, který je vytvořený z polygonu, u kterého jsou použitím operace offset zaobleny rohy. Výsledkem je obdélník se zaoblenými rohy.

Modul `MainTooth` je zub vyčnívající z bloku nebo je využit pro vytvoření výřezu. Je vytvořen z polygonu a použitím operace offset jsou zaobleny rohy. Výsledkem je lichoběžník se zaoblenými rohy.

Modul `PaperHoleThrough` je štěrbina pro vložení papírku s příkazem na pravé straně bloku. Je vytvořena z modulu `square`, na který aplikujeme operaci `linear_extrude` a nastavíme parametr `scale`. Výsledkem je hranol s lichoběžníkovou podstavou.

Modul `PaperHole` je okénko pro papírek s příkazem. Je vytvořen z modulu `MainPart`, na který aplikujeme operaci `linear_extrude` a nastavíme parametr `scale`. Výsledkem je hranol s lichoběžníkovou podstavou, ke kterému je připojen modul `PaperHoleThrough`.

Modul `TextUniversal` vytvoří text. Pomocí modulu color byla pro názornost změněna barva.

Postup vytvoření

Moduly `MainPart` a `MainTooth` jsou nejprve sloučeny. Následně se provede odečet posunutého modulu `MainTooth`. Celý výsledek je poté extrudován pomocí funkce `linear_extrude`. Od takto vytvořeného objektu se odečte posunutý modul `PaperHole`. Nakonec je přidán posunutý modul `TextUniversal` s textem „Do“.

Syntaxe

Ukázka syntaxe viz zdrojový kód 13.

```
1 module DefaultBlock(lenght_block = lenght_default, height_block =
      default_height_block, width_block = width, count_of_block =
      default_count_of_block, paper_hole = true);
```

Zdrojový kód 13: OpenSCAD syntaxe `DefaultBlock`

Parametry

- `length_block`: Číslo definující délku bloku na ose `x`. Musí být číslo větší nebo rovno 80.
- `height_block`: Číslo definující výšku bloku na ose `y`. Musí být číslo větší nebo rovno 30.
- `width_block`: Číslo definující šířku bloku na ose `z`. Musí být číslo v rozmezí 2 až 4.
- `paper_hole`: Logická hodnota (`true` nebo `false`) určující, zda blok obsahuje okénko pro papírek s příkazem.
  - `true`: Blok bude obsahovat okénko pro papírek s příkazem.
  - `false`: Blok nebude obsahovat okénko pro papírek s příkazem.

Příklad použití

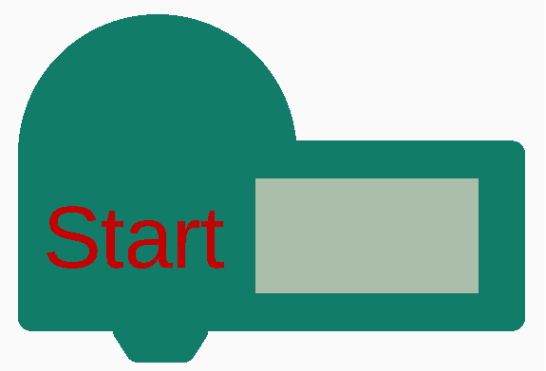
Ukázka použití viz zdrojový kód 14 nebo zdrojový kód 15.

```
1 DefaultBlock();
```

Zdrojový kód 14: OpenSCAD použití `DefaultBlock`

```
1 DefaultBlock(lenght_block = 10, height_block = 5, width_block = 3,
    count_of_block = 1, paper_hole = false);
```

Zdrojový kód 15: OpenSCAD použití `DefaultBlock` s vlastními parametry



Obrázek 23: OpenSCAD `StartBlock`

5.2 StartBlock

Modul `StartBlock` se používá k vytváření startovacího bloku se specifikovanou délkou, výškou a šířkou. Tento modul zahrnuje půlkruh a další tvary, které lze přizpůsobit pomocí parametrů. Ukázka bloku viz obrázek 23.

Použité moduly

K vytvoření byly využity moduly `MainPart`, `MainTooth`, `PaperHole`, `TextUniversal`. Tyto moduly jsou popsány v podkapitole 5.1.

Postup vytvoření

Nejprve jsou sloučeny moduly `MainPart` a `MainTooth`. Poté je vytvořen kruh, od něhož se odečte posunutý čtverec, čímž vznikne půlkruh. Tento půlkruh je následně posunut a na všechny objekty je aplikována operace `linear_extrude`. Od takto vytvořeného objektu se odečte posunutý modul `PaperHole`. Nakonec je přidán posunutý modul `TextUniversal` s textem „Start“.

Syntaxe

Ukázka syntaxe viz zdrojový kód 16.

```
1 StartBlock(lenght_block = lenght_default, height_block =
    default_height_block, width_block = width, paper_hole = true) {
```

Zdrojový kód 16: OpenSCAD syntaxe `StartBlock`

Parametry

- `length_block`: Číslo definující délku bloku na ose `x`. Musí být číslo větší nebo rovno 80.
- `height_block`: Číslo definující výšku bloku na ose `y`. Musí být číslo větší nebo rovno 30.
- `width_block`: Číslo definující šířku bloku na ose `z`. Musí být číslo v rozmezí 2 až 4.
- `paper_hole`: Logická hodnota (`true` nebo `false`) určující, zda blok obsahuje okénko pro papírek s příkazem.
  - `true`: Blok bude obsahovat okénko pro papírek s příkazem.
  - `false`: Blok nebude obsahovat okénko pro papírek s příkazem.

Příklad použití

Ukázka použití viz zdrojový kód 17 nebo zdrojový kód 18.

```
1    StartBlock()
```

Zdrojový kód 17: OpenSCAD použití `StartBlock`

```
1 StartBlock(lenght_block = 100, height_block = 40, width_block = 3,
    paper_hole = true);
```

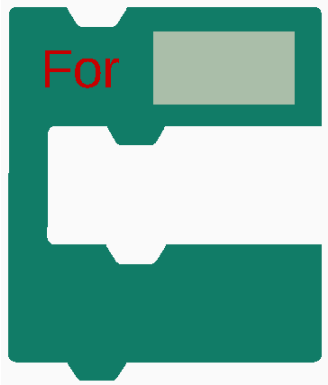
Zdrojový kód 18: OpenSCAD použití `StartBlock` s vlastními parametry

5.3 ForBlock

Modul `ForBlock` se skládá ze dvou částí: začátku a konce. Začátek obsahuje text „For“ a konec uzavírá celý blok. Mezi tyto části se vkládají příkazy. Parametry modulu umožňují nastavení počtu bloků mezi začátkem a koncem, délky, výšky, šířky bloku, aktivaci puzzle prvku, přítomnosti okénka pro papírek s příkazem a textu zobrazovaného na bloku. Ukázka bloku viz obrázek 24.

Použité moduly

K vytvoření byly využity moduly `MainPart`, `MainTooth`, `PaperHole`, `TextUniversal`. Tyto moduly jsou popsány v podkapitole 5.1. Dále byl využit `ConnectionBlock`, který je popsán v kapitole 5.5.



Obrázek 24: OpenSCAD `ForBlock`

Postup vytvoření

Nejprve použijeme modul `MainPart`, od kterého odečteme posunutý modul `MainPart`, `MainTooth` a další `MainTooth`. Poté je použit modul `MainTooth` a posunutý modul `MainTooth`. Na tyto objekty je aplikována operace `linear_extrude`. Od celého objektu je následně odečten posunutý modul `PaperHole` a `ConnectionBlock`. Nakonec je použit posunutý modul `TextUniversal` s textem „For“.

Syntaxe

Ukázka syntaxe viz zdrojový kód 19.

```
1 ForBlock(count_of_block_sum = 1, lenght_block = lenght_default,
    height_block = default_height_block, width_block = width,
    puzzle_active = true, paper_hole = true, text_block = "For");
```

Zdrojový kód 19: OpenSCAD syntaxe `ForBlock`

Parametry

- `count_of_block_sum`: Číslo definující počet bloků. Musí být kladné celé číslo.
- `length_block`: Číslo definující délku bloku na ose `x`. Musí být číslo větší nebo rovné 80.
- `height_block`: Číslo definující výšku bloku na ose `y`. Musí být číslo větší nebo rovno 30.
- `width_block`: Číslo definující šířku bloku na ose `z`. Musí být číslo v rozmezí 2 až 4.
- `puzzle_active`: Logická hodnota (`true` nebo `false`) určující, zda je puzzle prvek aktivní.
  - `true`: Puzzle prvek bude aktivní.
  - `false`: Puzzle prvek nebude aktivní.
- `paper_hole`: Logická hodnota (`true` nebo `false`) určující, zda blok obsahuje okénko pro papírek s příkazem.
  - `true`: Blok bude obsahovat okénko pro papírek s příkazem.
  - `false`: Blok nebude obsahovat okénko pro papírek s příkazem.
- `text_block`: Řetězec určující text, který bude zobrazen na bloku. Musí být řetězec s hodnotou „`For`“ nebo „`If`“.

Příklad použití

Ukázka použití viz zdrojový kód 20 nebo zdrojový kód 21.

```
1 ForBlock();
```

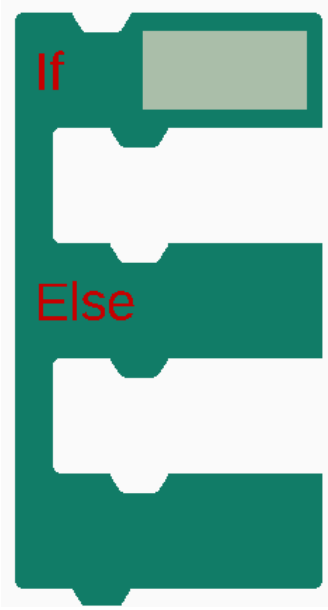
Zdrojový kód 20: OpenSCAD použití `ForBlock`

```
1 ForBlock(count_of_block_sum = 2, lenght_block = 100, height_block =
    40, width_block = 3, puzzle_active = false, paper_hole = true,
    text_block = "For");
```

Zdrojový kód 21: OpenSCAD použití `ForBlock` s vlastními parametry

5.4 IfBlock

Modul `IfBlock` vychází z bloku `ForBlock`. Parametry modulu umožňují nastavení počtu horních a dolních bloků, délky, výšky a šířky bloku, aktivaci puzzle prvku, přítomnosti okénka pro papírek s příkazem a určení, zda blok bude „If“ blok bez části „Else“. Ukázka bloku viz obrázek 25.



Obrázek 25: OpenSCAD `IfBlock`

Použité moduly

K vytvoření byly využity moduly `MainPart`, `MainTooth`, `PaperHole`, `TextUniversal`. Tyto moduly jsou popsány v kapitole 5.1. Dále byl využit `ConnectionBlock`, který je popsán v kapitole 5.5 a také `ForBlock`, který je popsán v kapitole 5.3.

Postup vytvoření

Nejprve použijeme modul `MainPart` a od něj odečteme dva posunuté moduly `MainPart` a tři posunuté moduly `MainTooth`. Na tyto objekty je aplikována operace `linear_extrude`. Od celého objektu poté odečteme dva moduly `ConnectionBlock`, které jsou různě posunuté. Následně odečteme modul `PaperHole`. K objektu jsou přidány tři posunuté moduly `MainTooth` a na tyto objekty je opět aplikována operace `linear_extrude`. Na závěr je přidán posunutý modul `TextUniversal` s hodnotami „If“ a „Else“.

Syntaxe

Ukázka syntaxe viz zdrojový kód 22.

```
1 IfBlock(upper = 1, lower = 1, lenght_block = lenght_default,
    height_block = default_height_block, width_block = width,
    puzzle_active = true, paper_hole = true, if_without_else = false
    );
```

Zdrojový kód 22: OpenSCAD syntaxe `IfBlock`

Parametry

- `upper`: Číslo definující počet horních bloků. Musí být kladné celé číslo.
- `lower`: Číslo definující počet dolních bloků. Musí být kladné celé číslo.
- `length_block`: Číslo definující délku bloku na ose `x`. Musí být číslo větší nebo rovné 80.
- `height_block`: Číslo definující výšku bloku na ose `y`. Musí být číslo větší nebo rovné 30.
- `width_block`: Číslo určující šířku bloku na ose `z`. Musí být číslo v rozmezí 2 až 4.
- `puzzle_active`: Logická hodnota (`true` nebo `false`) určující, zda je puzzle prvek aktivní.
  - `true`: Puzzle prvek bude aktivní.
  - `false`: Puzzle prvek nebude aktivní.
- `paper_hole`: Logická hodnota (`true` nebo `false`) určující, zda blok obsahuje okénko pro papírek s příkazem.
  - `true`: Blok bude obsahovat okénko pro papírek s příkazem.
  - `false`: Blok nebude obsahovat okénko pro papírek s příkazem.
- `if_without_else`: Logická hodnota (`true` nebo `false`) určující, zda blok bude „If“ blok bez části „Else“.
  - `true`: Blok bude pouze „If“ blok.
  - `false`: Blok bude obsahovat jak „If“ tak „Else“ část.

Příklad použití

Ukázka použití viz zdrojový kód 23 nebo zdrojový kód 24.

```
1 IfBlock();
```

Zdrojový kód 23: OpenSCAD použití `IfBlock`

```
1 IfBlock(upper = 2, lower = 1, lenght_block = 100, height_block = 40,
      width_block = 3, puzzle_active = true, paper_hole = true,
     if_without_else = false);
```

Zdrojový kód 24: OpenSCAD použití `IfBlock` s vlastními parametry

5.5 ConnectionBlock

Modul `Connectionblok` slouží ke spojení bloků v případě, že jsou bloky `IfBlock` a `ForBlock` rozděleny na části parametrem `puzzle_active=true`. Ukázka bloku viz obrázek 26.

Obrázek 26: OpenSCAD `ConnectionBlock`

Použité moduly

Byly využity moduly `add_connection` a `groove`. Modul `groove` je drážka vytvořena z polygonu, u kterého bylo pomocí operace `offset` provedeno zaoblení rohů.

Modul `add_connection` je tvořen z modulu `groove`, který je otočen přes operaci `mirror`. Dále si vytvoříme pomocí modulu `groove` druhý objekt a otočíme ho pomocí operace `mirror`. Výsledkem je model obrácené T.

Postup vytvoření

Pomocí modulu `square` je vytvořen obdélník, od něhož je odečten posunutý modul `add_connection`, a následně je přidán modul `add_connection`. Pro vysunutí všech objektů je poté aplikována operace `linear_extrude`, čímž je vytvořen objekt pro vložení. Alternativně lze použít stejný postup s jiným parametrem `remove_block`, což vede k vytvoření objektu pro odstranění bloku.

Syntaxe

Ukázka syntaxe viz zdrojový kód 25.

```
ConnectionBlock(count_of_block_sum = 1, remove_block = false,
    height_block = default_height_block, width_block = width);
```

Zdrojový kód 25: OpenSCAD syntaxe `ConnectionBlock`

Parametry

- `count_of_block_sum`: Číslo definující počet vložených bloků. Musí být číslo větší nebo rovno 0.5.
- `remove_block`: Logická hodnota (`true` nebo `false`) určující, zda bude blok odstraněn.
  - `true`: Blok bude odstraněn.
  - `false`: Blok bude vložen.
- `height_block`: Číslo určující výšku bloku na ose `y`. Musí být kladné číslo.
- `width_block`: Číslo určující šířku bloku na ose `z`. Musí být kladné číslo.

Příklad použití

Ukázka použití viz zdrojový kód 26 nebo zdrojový kód 27.

```
ConnectionBlock();
```

Zdrojový kód 26: OpenSCAD použití `ConnectionBlock`

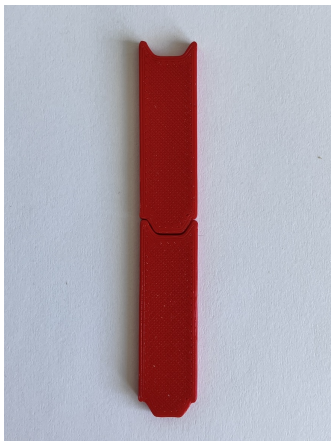
```
ConnectionBlock(count_of_block_sum = 1, remove_block = false,
    height_block = 50, width_block = 3);
```

Zdrojový kód 27: OpenSCAD použití `ConnectionBlock` s vlastními parametry

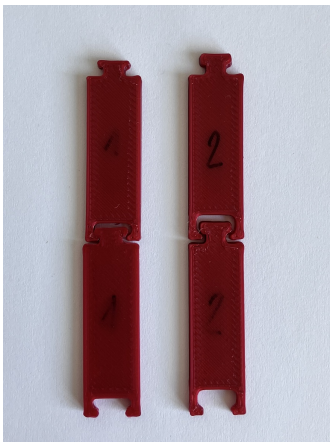
Optimalizace tvaru spojovacího modulu

V prvním návrhu tohoto bloku jsem plánoval využít modul zubu, jak je možné vidět na obrázku 27, a tím maximalizovat využití stejných modulů, které jsou použity v ostatních blocích. Po vytištění se však toto řešení ukázalo jako nevyhovující, protože při skládání bloků nedošlo k pevnému spojení. S bloky bylo nutné zacházet velmi jemně, aby nedocházelo k rozpojování, což celkově snižovalo komfort při manipulaci při skládání bloků.

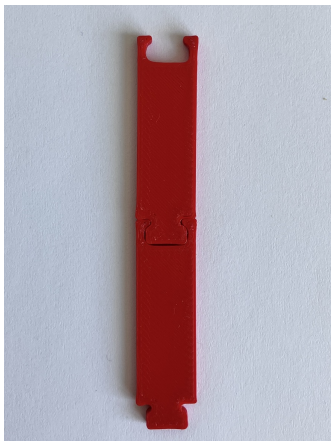
Pro odstranění tohoto problému byl změněn tvar spojovacího bloku a přidán modul, který měl původně tvar podobný obrácenému T se zkosenými hranami. Na obrázku 28 lze vidět, že po vytištění je zkosení velmi malé a zanedbatelné a tudíž nezajistí pevné spojení při skládání bloků. Proto byla následně použita varianta, ve která má blok tvar obráceného T bez zkosení, viz obrázek 29. Tím vznikl pevný spoj, který drží bloky spojené i v případě, že se s některou částí manipuluje.



Obrázek 27: Spojení module tooth



Obrázek 28: Spojení se zkosenými hranami



Obrázek 29: Spojení obrácené T

5.6 Ukázka použití knihovny

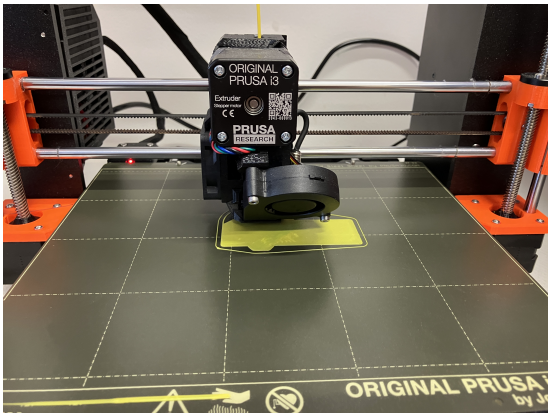
Postup vložení vlastní knihovny a její využití k vygenerování základního bloku je patrný ve zdrojovém kódu 28. Tímto se vygeneruje `DefaultBlock`, který lze následně uložit například jako STL soubor. Doporučuje se umístit knihovnu do stejného adresáře jako nový soubor, což umožní použití relativní cesty ke knihovně.

```
1 use <./library.scad>;
2 DefaultBlock();
```

Zdrojový kód 28: Volání `DefaultBlock`

5.7 3D tisk bloků

V této bakalářské práci byla využita 3D tiskárna Original Prusa i3 MK3S+. Viz obrázek 30. Ve sliceru byla nastavena změna barvy ve výšce, kde začíná text bloku. Když tiskárna dosáhne vrstvy, kde se začíná tisknout písmo, zastaví se a nabídne výměnu filamentu. Po vysunutí původního filamentu a zavedení nového bude tisk pokračovat a další vrstvy budou vytištěny odlišnou barvou.



Obrázek 30: 3D tisk `StartBlock`

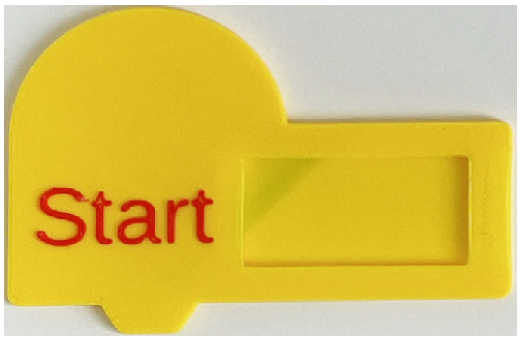
Některé bloky obsahují štěrbinu, nad níž během procesu 3D tisku vzniká most. Vzhledem k tomu, že se jedná o krátkou vzdálenost, lze předpokládat, že prověšení filamentu bude zanedbatelné. Pro tisk mostu není v takovém případě nutné používat podpěry, což významně zjednodušuje proces tisku a zároveň snižuje spotřebu materiálu. Na most nebude působit žádné mechanické namáhání, jde hlavně o jeho vzhled a funkčnost v rámci použití bloků. Po vytištění objektu je nezbytné během dodatečných úprav štěrbinu zkontrolovat, zda v ní nezůstal prověšený filament. Pokud ano, je vhodné jej odstranit například nožem.

Bloky byly navrženy tak, aby se použila výška vrstvy 0.2.

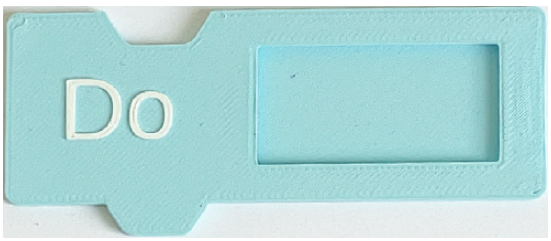
Jako vhodný vzor výplně se doporučuje použít mřížku s hustotou výplně nastavenou na 15 %. Alternativně lze využít vzory výplně trojúhelníky nebo plástev, přičemž v tomto případě lze použít shodnou hustotu výplně.

Přehled všech vytištěných bloků je prezentován na následujících obrázcích. Blok `DefaultBlock` je zobrazen na obrázku 32, `StartBlock` je zobrazen na obrázku 31, `IfBlock` je zobrazen na obrázku 33 a `ForBlock` je zobrazen na obrázku 34.

Barva bloků je volena tak, aby byla kontrastní vůči textu. `StartBlock` má žlutou barvu a červený text, `DefaultBlock` má barvu světle modrou s bílým textem. `IfBlock` a `ForBlock` mají červenou barvu a bílý text. Zvolení vlastní barvy umožňuje vytvářet různé kombinace, a nabízí možnost vytvořit inkluzivní didaktickou pomůcku i pro osoby s vadou zraku nebo barvoslepostí. Kontrast mezi barvou bloku a textem je klíčový pro zajištění čitelnosti a uživatelské přívětivosti.



Obrázek 31: Vytištěný `StartBlock`



Obrázek 32: Vytištěný `DefaultBlock`

Obrázek 33: Vytištěný `IfBlock`

Obrázek 34: Vytištěný `ForBlock`

6 Metodika využití vytištěných bloků

Tato kapitola ukazuje, jak s jednotlivými bloky pracovat ve výuce. Dále jsou představeny ukázky jednotlivých výukových lekcí. Tyto lekce je vhodné zařadit do kurzů pro začátečníky.

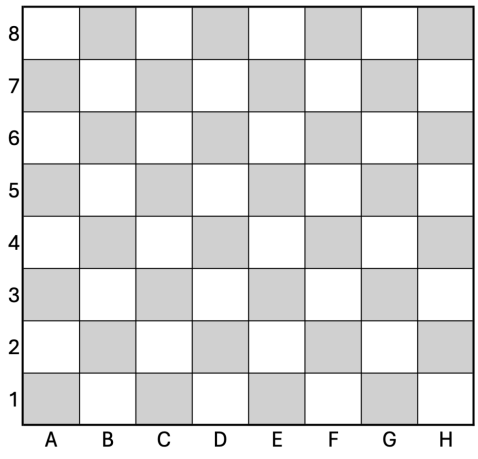
Žáci získají základy logického uvažování a zkušenosti s prací ve vizuálních programech. Během výuky si sami vyzkouší, jaké je to pracovat jako program a vykonávat zadané příkazy. Zapojení hmatu dopřeje žákům smyslovou zkušenost a využití motorických dovedností. Použití bloků je vhodné i pro žáky se zrakovým hendikepem, kteří tak mají možnost se s kódováním seznámit prostřednictvím hmatového vnímání.

Výuka kódování prostřednictvím bloků je vhodná pro mimoškolní aktivity. Například pro trávení volného času rodičů s dětmi. Rodič zde vystupuje jako průvodce, který umožňuje dítěti samostatně objevovat, chybovat a společně s ním řeší dané problémy. Je důležité, aby rodič měl k danému tématu alespoň základní znalosti nebo se s ním před začátkem aktivity seznámil.

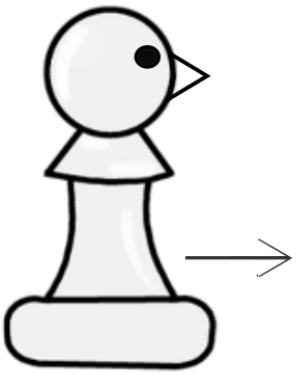
6.1 Ukázka výukových lekcí

Pro dále zmíněné lekce je potřeba připravit čtvercové hrací pole 8×8, příklad viz obrázek 35 a figurku s vyznačenou přední stranou, příklad viz obrázek 36. Záleží na uživateli, zda pole nakreslí na papír, použije šachovnici nebo využije venkovního prostranství, kde pole nakreslí na zem a děti budou představovat figurky.

Pro snadnější začátky je možné si vytisknout cheatlist A, kde jsou jednotlivé bloky graficky zpracovány a je možné na jednom místě vidět možnosti použití.



Obrázek 35: Hrací pole



Obrázek 36: Figurka s přední stranou a směrem pohybu doprava

6.1.1 První lekce

V rámci první lekce bude žákům představena vizuální programovací aplikace Scratch. Pro seznámení je možné využít zveřejněných tutoriálů a přehrát žákům část videa z části „Začínáme“[29]. V úvodní lekci se žáci seznámí s vizuálním programováním. Zjistí, jak je možné si sestavit první kód.

Motivace žáků: Žáci budou mít zájem dozvědět se, jak začít vytvářet vlastní program.

Cíle: Žáci si osvojí dovednost zadat příkaz pohybu vpřed a příkaz změny směru.

Pomůcky: Figurka, hrací pole, vytištěné bloky 1 ks `StartBlock`, 8 ks `DefaultBlock`

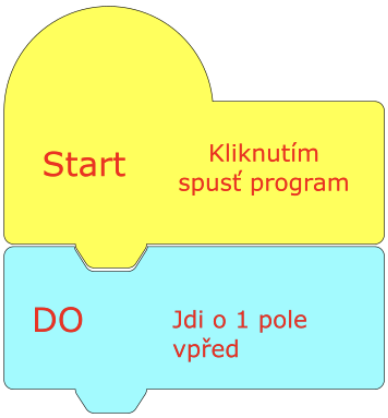
Úkol číslo 1

Úkolem je vytvořit kód, který bude řídit pohyb figurky na hracím poli. Figurka se musí pohybovat vpřed a měnit směr, aniž by opustila hrací pole. Jeden krok znamená posun o jedno pole na hracím poli. Výchozí pozice figurky je na poli A1 a přední strana figurky směřuje nahoru.

Příklad sestavení libovolného programu:

1. Program začíná blokem `StartBlock`.
2. Následuje první blok `DefaultBlock` s příkazem („Jdi o 1 pole vpřed“).
3. Po sestavení prvního programu je program vykonán podle zadaných příkazů.
4. Figurka je posunuta o krok vpřed.
5. Dále je přidán druhý `DefaultBlock` s příkazem („Otoč se o 90° vpravo“).

Řešení úkolu viz obrázek 37 a 38.



Obrázek 37: První lekce úkol číslo 1

Obrázek 38: První lekce úkol číslo 1 varianta 2

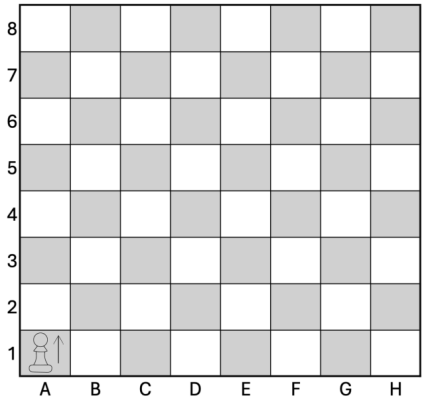
Úkol číslo 2

V tomto úkolu je cílem vytvořit jednoduchý program, který umožní figurce projít celým hracím polem po obvodu a vrátit se do výchozí pozice pomocí 8 bloků. Výchozí pozice figurky je na poli A1 a přední strana figurky směřuje nahoru viz obrázek 39.

Příklad sestavení programu:

1. `StartBlock`
2. `DefaultBlock` („Jdi o 7 polí vpřed“)
3. `DefaultBlock` („Otoč se o 90° vpravo“)
4. Krok 2 a 3 opakujeme 4×.

Tímto postupem vznikne velmi dlouhý program. Viz obrázek 40 Ten bude sloužit jako úvod pro následující lekci.



Obrázek 39: První lekce úkol číslo 2 výchozí pozice figurky

Obrázek 40: První lekce úkol číslo 2

Rozšíření:

Je vhodné nechat žáky, aby si navzájem zadávali příkazy a snažili se je plnit v prostoru učebny, přičemž budou sami vystupovat jako figurky. Další možností je zadat žákům vymyslet podobu kódu, pokud dojde ke změně velikosti hracího pole.

6.1.2 Druhá lekce

V této lekci se navazuje na předchozí lekci, konkrétně na příkazy pro procházení hracího pole. Je vhodné připomenout řešení úkolu. Po zopakování je představen `ForBlock`, který umožní provést úplně stejný program s využitím menšího množství bloků.

Cíle: Žáci si osvojí dovednost vytvořit cyklus opakování příkazu s využitím `ForBlock`.

Pomůcky: Figurka, hrací pole, vytištěné bloky 1 ks `StartBlock`, 4 ks `DefaultBlock`, 1 ks `ForBlock`, 4 ks `ConnectionBlock`

Úkol číslo 3
Úkolem je vytvořit program, který pomocí cyklu umožní figurce projít celým hracím polem po obvodu a vrátit se do výchozí pozice s co nejmenším počtem bloků. Výchozí pozice figurky je na poli A1 a přední strana figurky směřuje nahoru.

Řešení:

1. `StartBlock`
2. `ForBlock` („Opakuj 4×“)
3. `DefaultBlock` („Jdi o 7 polí vpřed“)
4. `DefaultBlock` („Otoč se o 90° vpravo“)

Úkol číslo 4
Úkolem je projít hrací pole úhlopříčně. Výchozí pozice figurky je na poli A1 a přední strana figurky směřuje nahoru.

Řešení:

1. `StartBlock`
2. `ForBlock` („Opakuj 7×“)
3. `DefaultBlock` („Jdi o 1 pole vpřed“)
4. `DefaultBlock` („Otoč se o 90° vpravo“)
5. `DefaultBlock` („Jdi o 1 pole vpřed“)
6. `DefaultBlock` („Otoč se o 90° vlevo“)

Rozšíření:
Rozmístěte na hracím poli různé předměty. Úkolem je vytvořit program, který umožní figurce tyto předměty posbírat. Místo předmětů můžete také například vložit „tajný“ vzkaz, který žáci po nasbírání všech položek můžou vyluštit. Pro zvýšení obtížnosti lze stanovit pořadí, ve kterém mají být předměty sbírány.

6.1.3 Třetí lekce

Tuto lekci je vhodné začít zopakováním předchozího učiva a následně představit využití větvení programu.

Cíle: Žáci získají zkušenost s větvením programu.

Pomůcky: Figurka, hrací pole, vytištěné bloky 1 ks `StartBlock`, 4 ks `DefaultBlock`, 2 ks `ConnectionBlock`, 1 ks `IfBlock`

Úkol číslo 5

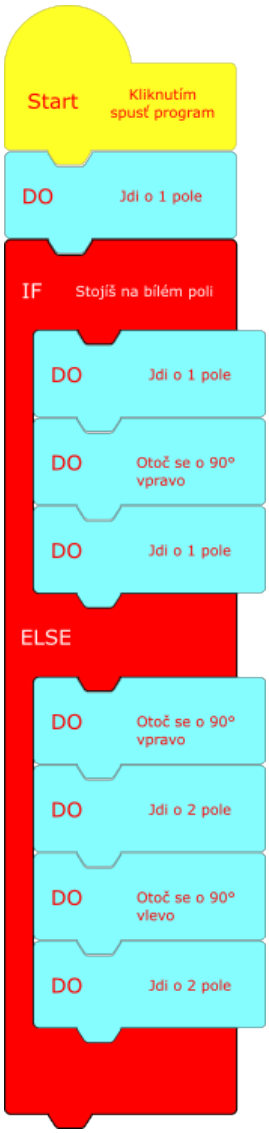
Úkolem je vytvořit program s následujícím pohybem figurky. Pokud se figurka nachází na bílém poli, otočí se třikrát o 90° doprava pomocí cyklu for a poté se posune o tři pole vpřed. Pokud se figurka nachází na černém poli, otočí se o 90° doleva a poté se posune o dvě pole vpřed.

Program, který bude pracovat následujícím způsobem:

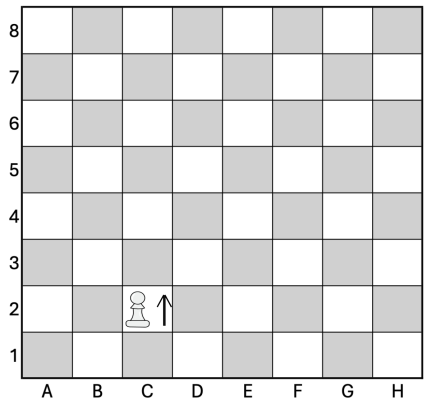
1. `StartBlock`
2. `IfBlock` („Pozice na bílém poli“)
3. `ForBlok` („3×“)
4. `DefaultBlock` („Otoč se o 90° vpravo“)
5. `DefaultBlock` („Jdi o 3 pole vpřed“)
6. `Else`
7. `DefaultBlock` („Otoč se o 90° vlevo“)
8. `DefaultBlock` („Jdi o 2 pole vpřed“)

Úkol číslo 6

Úkolem je určit, na kterém poli se bude nacházet figurka po vykonání programu uvedeného na obrázku 41 a kam bude směrovat její přední strana. Výchozí pozice figurky je na poli C2 viz obrázek 42 a přední strana figurky směřuje nahoru.



Obrázek 41: Třetí lekce úkol číslo 6



Obrázek 42: Třetí lekce úkol číslo 6 výchozí pozice figurky

Řešení:

Po provedení programu bude figurka stát na poli E5 a bude směřovat nahoru.

Rozšíření:

Na hracím poli určete jedno nebo více polí, na která nesmí figurka stoupnout.

Závěr

Tato bakalářská práce se zabývá možnostmi využití 3D tisku k výuce kódování. Byla vytvořena knihovna v jazyce OpenSCAD, která obsahuje programovací bloky založené na vzhledu bloků v Scratch. Tato knihovna umožňuje modifikovat velikost bloků dle parametrů uživatele a vytvářet tak didaktické pomůcky přizpůsobené konkrétnímu druhu použití pro skupiny či jednotlivce.

V práci je popsána problematika 3D tisku, dále je popsán software a programovací jazyk OpenSCAD, ve kterém byly modely vytvořeny. Text se také zabývá vizuálním programováním a popisuje vybrané programovací jazyky. Je představena programátorská dokumentace a proces tvorby bloků. Jsou popsány jednotlivé bloky i postupné kroky průběhu jejich vytvoření.

Pro usnadnění integrace do výuky byly vytvořeny ukázkové lekce. Tyto výukové materiály slouží jako úvod do vizuálního kódování a poskytují základ pro další rozvoj a přípravu na budoucí studium programování. Pro lektory jsou rovněž přiloženy SVG soubory, které umožňují tvorbu vlastních zadání, čímž se zvyšuje přizpůsobení výuky individuálním potřebám žáků.

Pro lepší orientaci v používání bloků mohou lektoři pro žáky vytisknout cheatsheet A, kde jsou jednotlivé bloky graficky zpracovány a přehledně uspořádány, což usnadňuje jejich praktické využití.

Kód je důkladně okomentován, což umožňuje jeho budoucí rozšíření nebo modifikaci. Jako návrh na další rozšíření je vhodné uvažovat o modulární úpravě bloků `IfBlock` a `ForBlock`. Tato úprava by zahrnovala vytvoření systému „šuplíků“, který by umožňoval snadnější sestavení celého bloku a eliminoval potřebu puzzle dílků.

Na základě navržených bloků je rovněž možné vyvinout aplikaci, která by pomocí fotoaparátu převáděla fyzické bloky do digitální podoby a vytvářela tak funkční programy. Scratch umožňuje nahrát soubor s příponou .sb3. V případě tohoto rozšíření by byly jednotlivé bloky identifikovány pomocí textu uvedeném na bloku.

Ukázky výukových lekcí byly testovány na letním táboře skautského oddílu Mustangové z Uherského Hradiště. Při testování se zapojilo celkem 15 dětí. Jednalo se o skupinu dětí ve věku 10 až 12 let. Některé z nich mají zkušenost s vizuálními programovacími jazyky (například Scratch) z výuky informatiky. Tato část skupiny pozitivně reagovala na možnost využít bloky v off-line prostředí. Testování poskytlo zpětnou vazbu a ukázalo, že navržená velikost bloků je vhodná pro starší děti s ohledem na dovednost psát a zvolit vhodnou velikost písma na papírek s příkazem. Pro mladší děti, které píšou velkými písmeny, je vhodné mít bloky ve větší velikosti. Úkoly v jednotlivých lekcích řešily děti různého věku odlišně. Úroveň obtížnosti v první lekci byla pro mladší děti vhodně zvolena, pro začátečníky šlo o přiměřeně složité úkoly. Pro starší děti byly tyto úkoly snadné a rychle plnily úkoly z druhé lekce.

Conclusions

This bachelor's thesis explores the potential of 3D printing as a pedagogical tool in the context of code education. A library in the OpenSCAD language was constructed, comprising a program based on the Scratch layout. The library permits the user to modify the size of the blocks according to their own parameters, thus creating didactic tools for a specific type of use for groups or individuals.

The thesis begins by delineating the challenges associated with 3D printing. It then proceeds to describe the software and program utilized in the creation of the models, namely the OpenSCAD programming language. Moreover, the text addresses the topic of visual programming and provides an overview of the Scratch application and programming language. Moreover, the methodology employed by the author in the creation of the blocks is elucidated. The individual blocks are described in detail, as are the subsequent steps in the process of their creation.

Sample lessons have been developed with the intention of facilitating integration into the classroom. The lessons serve as an introduction to visual coding and provide a foundation for further development and preparation for future programming studies. Instructors may also find the SVG files useful for creating custom assignments, thereby enhancing the flexibility and adaptability of teaching to meet individual student needs. To facilitate the use of the blocks, students may wish to print out the cheat sheet A, which presents the individual blocks in a graphical arrangement and clearly organised manner, thus facilitating practical use. The code is meticulously documented, thereby facilitating prospective extensions or potential differentiation by any interested party. It is recommended that a modular modification of the `IfBlock` and `ForBlock` blocks be considered as a potential avenue for further extension. Such a modification would entail the creation of a system whereby the entire block could be assembled with greater ease, obviating the necessity for puzzle pieces.

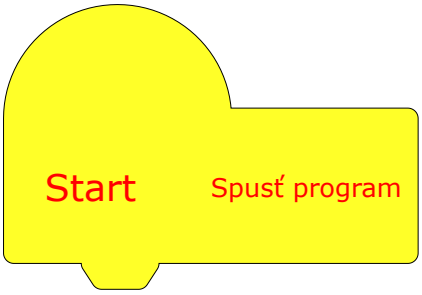
Furthermore, the aforementioned blocks could be employed in the development of an application that would utilise a camera to convert the physical blocks into digital form, thereby creating functional programs. Scratch permits the uploading of a file with a .sb3 extension. In the case of this proposed extension, the individual blocks would be identified by the text displayed on the block itself.

The efficacy of the sample lessons was evaluated at a Scout summer camp of troop named Mustangs from Uherské Hradiště. A total of 15 children participated in the testing phase. The participants were a group of children aged between 10 and 12 years old. Some of the participants had prior experience with visual programming languages, such as Scratch, which they had encountered in computer science lessons. The majority of the group expressed enthusiasm for the opportunity to utilise the blocks in an offline setting. The trial yielded valuable insights, indicating that the proposed block size was well-suited for older children with regard to writing skills and selecting an appropriate font size for command paper. For younger children who write in capital letters, it is appropriate to

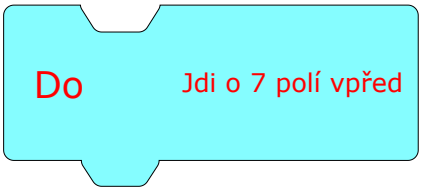
have blocks in a larger size. The tasks in each lesson were solved by children of different ages in a variety of ways. The level of difficulty in the first lesson was appropriately chosen for the younger children, and the tasks were reasonably challenging. The tasks set for the older children were relatively straightforward and were completed with minimal effort in the second lesson.

A Cheatsheet

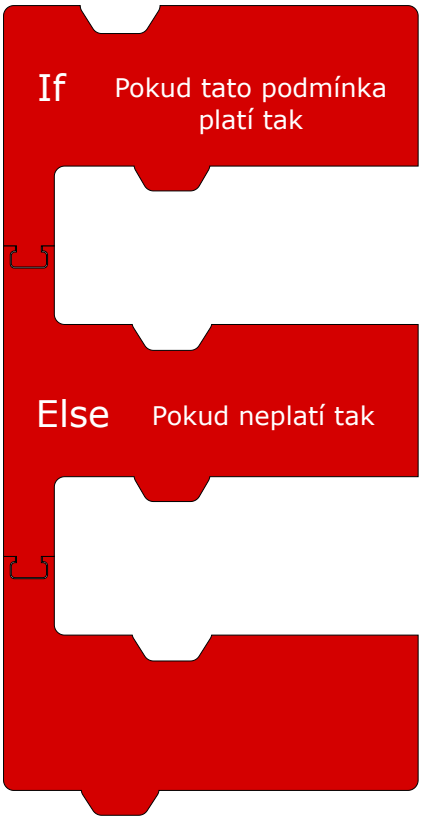
StartBlock



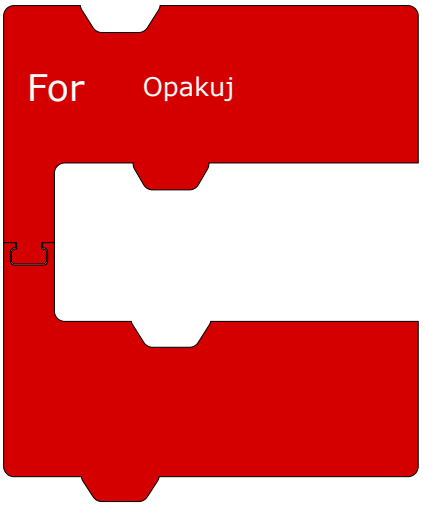
DefaultBlock



IfBlock



ForBlock



B Obsah přiloženého datového média

`library.scad`
: Knihovna vytvořená v rámci bakalářské práce.

`test_case.scad`
: Ukázka použití knihovny.

`svg_bloky/`
: Složka se všemi vygenerovanými svg bloky.

`text/`
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, apod.

`readme.txt`
: Instrukce pro spuštění knihovny `library.scad`, včetně všech požadavků pro její bezproblémové spuštění a vygenerování požadovaných souborů.

Literatura

[1] Prusa Research a.s. *Průša pro školy* [online]. [cit. 2023-11-2]. Dostupný z: ⟨https://proskoly.prusa3d.cz⟩.

[2] Handsoncoding. *Handsoncoding* [online]. [cit. 2024-1-8]. Dostupný z: ⟨https://www.handsoncoding.org/⟩.

[3] Csforall. *Hands on Coding* [online]. [cit. 2024-1-8]. Dostupný z: ⟨https://www.csforall.org/members/hands_on_coding/⟩.

[4] SSWW. *Hands-On Coding 12-Piece Coding Block Set* [online]. [cit. 2024-1-9]. Dostupný z: ⟨https://www.ssww.com/item/hands-on-coding-12-piece-coding-block-set-LR4136/⟩.

[5] Printables. *Handsoncoding* [online]. [cit. 2024-1-8]. Dostupný z: ⟨https://www.handsoncoding.org/⟩.

[6] Vochozka, Vladimír. *3D tisk ve výuce fyziky.* Třetí vyd. Praha: MIT Press, 2022. 332 s. ISBN 978-80-7235-665-2.

[7] materialpro3d. *Programy pro 3D modelování* [online]. [cit. 2024-3-3]. Dostupný z: ⟨https://www.materialpro3d.cz/blog/programy-pro-3d-modelovani/⟩.

[8] ZDnet. *Everything you need to know about 3D printing and its impact on your business* [online]. [cit. 2024-1-9]. Dostupný z: ⟨https://www.zdnet.com/article/everything-you-need-to-know-about-3d-printing-and-its-impact-on-your-business/⟩.

[9] Trnečková, Markéta. *Základy 3D tisku* [online]. [cit. 2023-10-10]. Dostupný z: ⟨https://www.dropbox.com/scl/fi/rhmhudj9d6drj7us7iihk/skripta_3dt.pdf?rlkey=qg6xhndw4g4atbxjvui6u6wg0&e=1&dl=0⟩.

[10] Adobe. *Soubory STL* [online]. [cit. 2023-11-6]. Dostupný z: ⟨https://www.adobe.com/cz/creativecloud/file-types/image/vector/stl-file.html⟩.

[11] Tech CAD solution s.r.o. *5 nejčastějších formátů 3D souborů* [online]. [cit. 2024-8-1]. Dostupný z: ⟨https://www.techcad.cz/l/5-nejcastejsich-formatu-3d-souboru/⟩.

[12] Stříteský, Ondřej. *Základy 3D tisku s Josefem Průšou: Technologie 3D tisku* [online]. Verze 1.0. Praha: Prusa research, 2020, 2008 [cit. 2023-10-10]. 62 s. Dostupný z: ⟨https://www.prusa3d.com/cs/stranka/zaklady-3d-tisku-s-josefem-prusou_490/?gad=1&gclid=Cj0KCQjw7JOpBhCfARIsAL3bobcj0SmRb7jVM1DImQ8y98pe-UfaeKGgzgql6tq56_ul7FhbPwltkzwaAuPXEALw_wcB⟩.

[13] Císař, Dominik. *Slicuj jako bůh! Průvodce začátečníka po Slic3r Prusa Edition* [online]. [cit. 2024-4-10]. Dostupný z: ⟨https://josefprusa.cz/slicuj-jako-buh-pruvodce-zacatecnika-slic3r-prusa-edition/⟩.

[14] 3D Stisk s.r.o. *Špatné přemostění (bridging) v 3D tisku: Příčiny, problémy a řešení* [online]. [cit. 2024-6-1]. Dostupný z: ⟨https://3dstisk.cz/spatne-premosteni-bridging-v-3d-tisku-priciny-problemy-a-reseni/⟩.

[15] Horončok, Miro. *Tisk: Mosty* [online]. [cit. 2024-3-9]. Dostupný z: ⟨https://courses.fit.cvut.cz/BI-3DT/tutorials/bridges.html⟩.

[16] Prusa Research a.s. *Špatné přemostění (bridging)* [online]. [cit. 2024-6-1]. Dostupný z: ⟨https://help.prusa3d.com/cs/article/spatne-premosteni-bridging_1802⟩.

[17] Bricknell, James. *Best 3D Printing Slicer: PrusaSlicer, Cura and More* [online]. [cit. 2024-1-9]. Dostupný z: ⟨https://www.cnet.com/tech/computing/the-best-3d-printing-slicer/⟩.

[18] Prusa Research a.s. *PrusaSlicer 2.7.1* [online]. [cit. 2024-1-9]. Dostupný z: ⟨https://www.prusa3d.com/cs/stranka/prusaslicer_424/⟩.

[19] OpenSCAD. *About OpenSCAD* [online]. [cit. 2024-4-10]. Dostupný z: ⟨https://openscad.org/about.html⟩.

[20] Kočandrlová, Milada; Černý, Jaroslav. *Konstruktivní geometrie*. Třetí vyd. Praha: České vysoké učení technické v Praze, 2016. 262 s. ISBN 978-80-01-06049-0.

[21] Trnečková, Markéta. *3D modelování v OpenSCAD* [online]. [cit. 2024-4-25]. Dostupný z: ⟨https://mfi.upol.cz/files/32/3204/mfi_3204_293_312.pdf⟩.

[22] OpenSCAD. *OpenSCAD User Manual* [online]. [cit. 2024-4-12]. Dostupný z: ⟨https://en.wikibooks.org/wiki/OpenSCAD_User_Manual/The_OpenSCAD_Language⟩.

[23] Uithoven, Peter; Amersfoort, Fablab. *OpenSCAD* [online]. [cit. 2024-4-12]. Dostupný z: ⟨https://openscad.org/cheatsheet/index.html⟩.

[24] Horončok, Miro. *OpenSCAD* [online]. [cit. 2024-1-9]. Dostupný z: ⟨https://courses.fit.cvut.cz/BI-3DT/tutorials/openscad.html⟩.

[25] *Visual programming language*. [online]. [cit. 2024-5-2]. Dostupný z: ⟨https://en.wikipedia.org/wiki/Visual_programming_language?oldid=893636949⟩.

[26] SGP Systems. *Originální výuka programování* [online]. [cit. 2024-4-14]. Dostupný z: ⟨https://sgpsys.com/cz/⟩.

[27] *Baltík*. [online]. [cit. 2024-4-14]. Dostupný z: ⟨https://cs.wikipedia.org/wiki/Balt%C3%ADk⟩.

[28] Kodu Game Lab. *What is Kodu?* [online]. [cit. 2024-4-14]. Dostupný z: ⟨https://www.kodugamelab.com/about/⟩.

[29] Marji, Majed. *Learn to program with Scratch: A visual introduction to programming with games, art, science, and math*. Druhé vyd. Praha: William Pollock, 2014. 261 s. Pokusná edice. ISBN 1-59327-543-9.

[30] Wallach Kloski, Liza; Kloski, Nick. *Začínáme s 3D tiskem.* První vyd. Brno: Computer Press, 2017. 211 s. ISBN 978-80-251-4876-1.