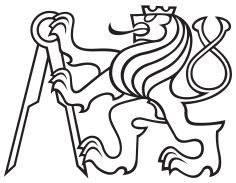
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická

Krabičky v nástroji I3T

Sofie Šašorina

Vedoucí práce: Ing. Petr Felkel, Ph.D.
Obor: Softwarové inženýrství a technologie
Květen 2021

Poděkování

Ráda bych poděkovala panu Ing. Petru Felkelovi, Ph.D. za vedení této práce. Velmi oceňuji jeho nasazení při řízení celého projektu a cenné rady. Taktéž děkuji kolegům Martinu Herichovi, Miroslavu Müllerovi, Jaroslavu Holečkovi a Danielu Grunclovi, se kterými mi bylo potěšením spolupracovat na tomto projektu. Nakonec bych chtěla poděkovat svému drahému příteli Martinu Turkovi za psychickou podporu při tvorbě práce.

Prohlášení

Prohlašuji, že jsem tuto práci napsala samostatně a výhradně s použitím citovaných pramenů.

Abstrakt

Tato práce řeší modernizaci nástroje I3T, který nedosahuje požadovaných standardů, kladených na moderní výukové nástroje.

Celá modernizace nástroje I3T je řešena v několika závěrečných pracích. Tato konkretní práce se zabývá jednou z částí přípravy náhrady stávajícího nástroje I3T a přechodem do nového uživatelského rozhraní implementováním pomocí knihovny ImGui.

To obnáší především návrh nové architektury a zpřehlednění starého kódu.

Výstupem práce je dokončená část nástroje I3T odpovidajicí za zobrazení okna Workspace.

Klíčová slova: I3T, GUI, ImGui, GLFW, 3D transformace, počítačová grafika, OpenGL

Vedoucí práce: Ing. Petr Felkel, Ph.D. Praha 2, Karlovo náměstí 13, E-413.

Abstract

This thesis resolve modernization of interactive teaching tool of transformations I3T.

Unfortunately, the application does not satisfy requirments for modern teaching tool. It is not attractive and intuitive for using.

Full modernization of the tool is described in various thesises. The main goal of this work is to prepare transition to new user interface implemented using Dear ImGui library. It involves to design new architecture and make less complex and more readable code.

The result of the work is functional Workspace modul.

Keywords: I3T, GUI, ImGui, GLFW, 3D transformations, computer graphics, OpenGL

Title translation: Node editor in I3T tool

Obsah

Obrázky

Tabulky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Šašorina** Jméno: **Sofie** Osobní číslo: **469907**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Převedení krabiček v interaktivním nástroji na výuku transformací do knihovny ImGui

Název bakalářské práce anglicky:

Conversion of Blocks in the I3T tool into the ImGui library

Pokyny pro vypracování:

V interaktivním nástroji na výuku transformací (I3T) dokončete pracovní okno, ve kterém se rozmísťují jednotlivé moduly (krabičky).
To znamená:
– definujte postup, jak převézt stávající implementaci modulů, do implementace pomocí knihovny ImGui, funkce vhodně rozvrhněte, abyste minimalizovala podobný kód
– patřičně upravte zpracování událostí (vstupu uživatele)
– implementujte návrh GUI dle studie Víta Zadiny a grafického návrhu Lukáše Pilky
– definujte způsob propojení výkonné funkce modulu se vstupy a výstupy
– převeďte všechny stávající moduly do navržené nové verze
Popište postup, jak v nové implementaci přidat nový modul a definovat jeho rozhraní.
Implementaci ověřte na jedenácti existujících výukových scénách.

Seznam doporučené literatury:

1. Michal Folta. Teaching of Transformations. Diplomová práce, FEL ČVUT, 2016.
http://dcgi.fel.cvut.cz/theses/2016/foltamic
2. Petr Felkel, Alejandra Magana, Michal Folta, Alexa Gabrielle Sears, Bedrich Benes
I3T: Using Interactive Computer Graphics to Teach Geometric Transformations.
Eurographics Education Papers 2018, http://www.i3t-tool.org/
3. Lukáš Pilka: Grafické návrhy rozhraní pro nástroj I3T, interní komunikace, 2018
4. Vít Zadina. Testování užitečnosti nástroje pro výuku transformací
Fakulta informačních technologií, ČVUT, 2019
5. Ocornut. “Ocornut/Imgui.” GitHub, January 17, 2020.
https://github.com/ocornut/imgui.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Petr Felkel, Ph.D., Katedra počítačové grafiky a interakce

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **15.02.2021** Termín odevzdání bakalářské práce: **21.05.2021**

Platnost zadání bakalářské práce: **19.02.2023**

Ing. Petr Felkel, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

Kapitola 1

Úvod

Tato práce se zabývá modernizaci nástroje I3T, což je nástroj pro interaktivní výuku transformací. Současně používaný nástroj I3T nesplňuje požadavky týkající se především přehlednosti a intuitivního ovládání.

Celá modernizace nástroje I3T je řešena v několika závěrečných pracích jejichž výstupy můžeme shrnout do několika globálních cílů.

Globální cíle projektu jsou:

1. Úprava kódu, aby bylo možné aplikaci snadno rozšířit či modifikovat.
2. Tvorba více atraktivního a intuitivního uživatelského rozhraní.
3. Přizpůsobit program pro různorodé platformy.

Výstupem této práce by měla být v rámci projektu úprava okna Workspace a node editoru, aby odpovídali výše uvedeným cílům.

Práci lze rozdělit do několika kroků:

- Seznámení s knihovnami ImGui a Imgui Node Editor
- Navržení nové struktury kódu - Hierarchie tříd.
- Převední všech "krabiček"z původní verze programu I3T do nové podoby.
- Připravit návod pro možné modifikace programu.
- Vytvořit vizuální podobu "krabiček"na základě studie Víta Zadiny a grafického návrhu Lukáše Pilky.
- Ověření implementaci na jedenácti existujících výukových scénách.

Vytvoření této práce by mělo obnášet navržení nové struktury kódu s použitím knihoven ImGui a Imgui Node Editor.

Výstupem práce by měla být dokončená část nástroje I3T odpovídající za zobrazení okna Workspace.

Kapitola 2
Současná podoba I3T

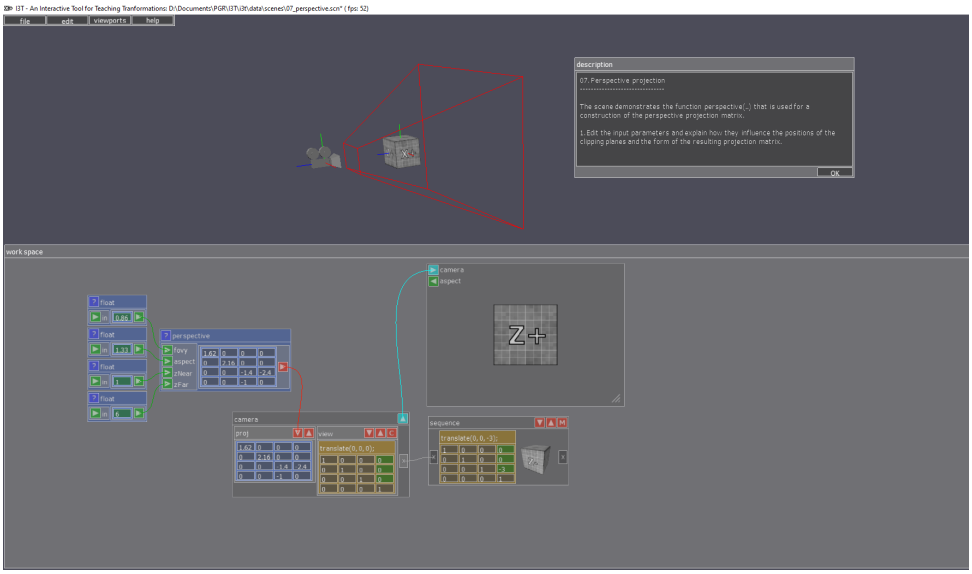
2.1 Historie I3T

I3T je interaktivní nástroj pro výuku transformací, který je využíván na Katedře počítačové grafiky a interakce v předmětu Programování grafiky. Tento program vznikl v roce 2016, jako výstup diplomové práce Michala Folty na Katedře počítačové grafiky a interakce při Fakultě Elektrotechnické ČVUT v Praze. [1] V dalších letech byl program dalšími závěrečnými pracemi několikrát modifikován. [2][4]

V roce 2020 vznikla práce, jejímž autorem byl pan Vít Zadina, která se zabývala užitečností tohoto programu. [3] Výstupem této práce byla analýza užívání tohoto programu studenty předmětu Programování grafiky a následné stanovení jeho hlavních nedostatků a důvodů pro minimální využívání studenty.

2.2 Pracovní rozhraní

Současný program I3T je napsán v jazyce C++ s využitím knihovny OpenGL. Uživatelské rozhraní programu je rozděleno na dvě části. Pohled na 3D scénu a pracovní plochu. Vlevo nahoře se nachází ovládací lišta, kde lze například načíst soubor s již rozpracovanou scénou, či vytvořit scénu novou. Konkrétní podoba je zřejmá z obrázku 2.1.



Obrázek 2.1: Původní podoba nástroje

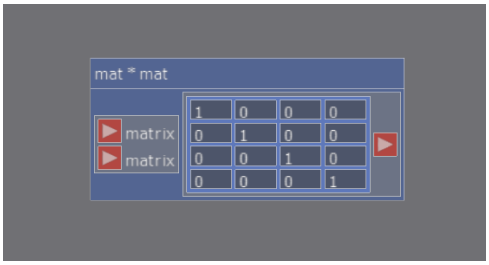
Ve 3D scéně může uživatel myší měnit pozici a směr pohledu kamery na transformovaný objekt. Uprostřed scény uživatel vidí barevně znázorněny osy x,y,z v prostoru.

Na pracovní ploše může uživatel modifikovat a ovládat vytvořené objekty, která jsou přítomné na scéně. S objektem je možné pracovat pomocí tzv. "krabiček". Jedná se o jednotlivé tabulky, ve kterých je možné upravovat požadované hodnoty. Každá "krabička"znázorňuje konkretní operaci. Jedná se o:

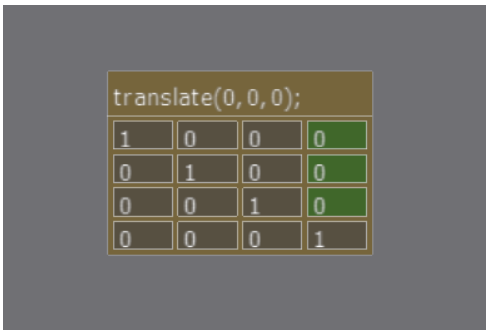
- Translaci
- Eulerovskou rotací podle určité osy
- Rotaci s pomoci kvaternionu
- Scale
- Look At
- Ortogonalní projekce
- Perspektivá
- Frustum
- Inverzní matice
- TrackBall
- Determinant
- Transpozice

- Násobení Matic
- Operace s Matici a jinými datovými typy
- Rozklad matice na translační a rotační složku
- Operace s Vektory
- Jednoduchý matematický operace s čísly
- Operace s kvaterniony
- Cyklus

"Krabičky"je možné rozdělit na dvě velké kategorie: operátory (modré, viz obrázek 2.2)) a transformace (žluté, viz obrázek 2.3)). Dále existuje několik speciálních "krabiček", které nelze zařadit do zmíněných kategorií. Jedná se o sekvenci, kameru a screen.



Obrázek 2.2: příklad "krabičky"operátoru

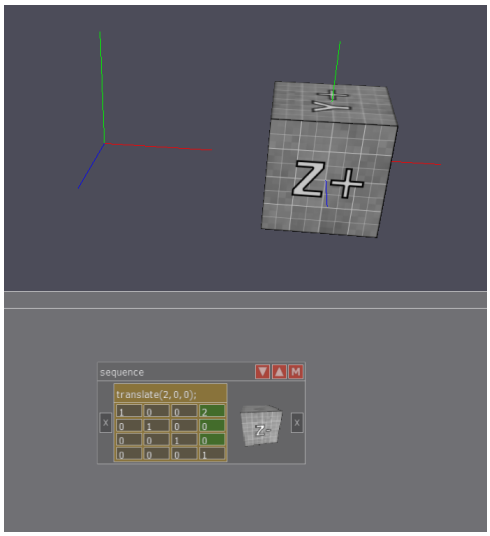


Obrázek 2.3: příklad "krabičky"transformace

Operátory zpravidla (až na pár výjimek) nelze nastavovat na konkretní hodnoty a měli by sloužit především, jako "kalkulačka"pro lepší pochopení výpočtu transformace. Taktéž nemají vliv na objekty na scéně, pokud je nenapojíme do matice v sekvenci. "Krabička"sekvence slouží k vynásobení "krabiček", které obsahuje, mezi sebou. Sekvenci lze navázat na konkretní objekt ve 3D scéně. Výsledek násobení matic je pak přímo aplikován na daný objekt. Do sekvence je možné vkládat pouze "krabičky"transformace. "Krabičky"transformace se od operátoru odlišují především tím, že neobsahují

vstupy, ani výstupy a je možné v nich editovat hodnoty. Editace je možná podle toho, jakou funkci daná "krabička"nese. Např. v "krabičce"translace lze editovat pouze první, druhou a třetí hodnotu v posledním sloupci matice, které znázorňují pohyb tělesa po x, y a z souřadnicích v prostoru (viz obrázek 2.4).

Je také možné "krabičky"mezi sebou spojovat. V řadě případů je to dokonce nutné pro dosažení požadovaných výsledků. Spojením "krabiček"dojde k přenesení dat (pozn. typ dat určuje typ výstupu) od výstupu první "krabičky"do vstupu druhé "krabičky", která tento vstup zpracuje dle typu operace, kterou daná "krabička"znázorňuje.



Obrázek 2.4: příklad aplikované translace

2.3 Nedostatky nástroje I3T

Program poskytuje uživateli široké možnosti a názorným způsobem dokáže předvést aplikaci transformace, ovšem není studenty využíván. Mezi hlavní důvody, proč program není využíván, řadí práce pana Víta Zadiny především strohý vzhled a neintuitivní ovládaní. Dalším důvodem je malá povědomost mezi studenty o existenci programu. Nástroj taky obsahoval sérií výukových scénářů, ovšem povědomost o existenci těchto scénářů byla mezi studenty taktéž velmi nízká.[3]

Jednou částí řešení problému neintuitivnosti ovládání se v modernizované verzi zabývá ve své závěrečné práci pan Miroslav Müller. Výstup jeho závěrečné práce by měl obsahovat sérii několika tutoriálů, které by měli uživatele seznámit s ovládáním programu.

Taktéž by měla být předělána grafická podoba "krabiček". Na to se zaměřuje táto závěrečná práce.

Nedostatky současné podoby programu jsou taktéž v backendu. Jednotlivé moduly programu jsou velice silně mezi sebou provázané a chybějící dokumentace nebo komentáře v kódu programu přispívali k jeho nečitelnosti.

Tento problém byl částečně vyřešen přidáním logovacího systému, který usnadnil proces dalšího vývoje[2], ale zcela problém nevyřešil.

Dalším nedostatkem současného programu je také skutečnost, že lze spustit pouze na operačním systému Windows.

Kapitola 3

Návrh řešení

Při navrhovaní nové architektury na základě provedené analýzy předchozího kódu bylo rozhodnuto o přenesení propojení výkonné funkce modulu se vstupy a výstupy z "krabičky"do zvláštního modulu "Jádro". Implementaci "Jádra"se ve své závěrečné práci zabývá kolega Martin Herich.

Navržení struktury "krabiček"vycházelo z předpokladu, že na většinu "krabiček"budou kladeny podobné uživatelské požadavky a budou mít obdobné vykreslovací funkce. Byla tedy vytvořena základní třída "krabičky", která v sobě obsahovala základní parametry. Následně vytvořené "krabičky"od této základní třídy dědili parametry a funkce a v případe potřeby přetížili potřebné vykreslovací funkce. Podrobnější struktura kódu bude rozepsána v kapitole 4. Implementace.

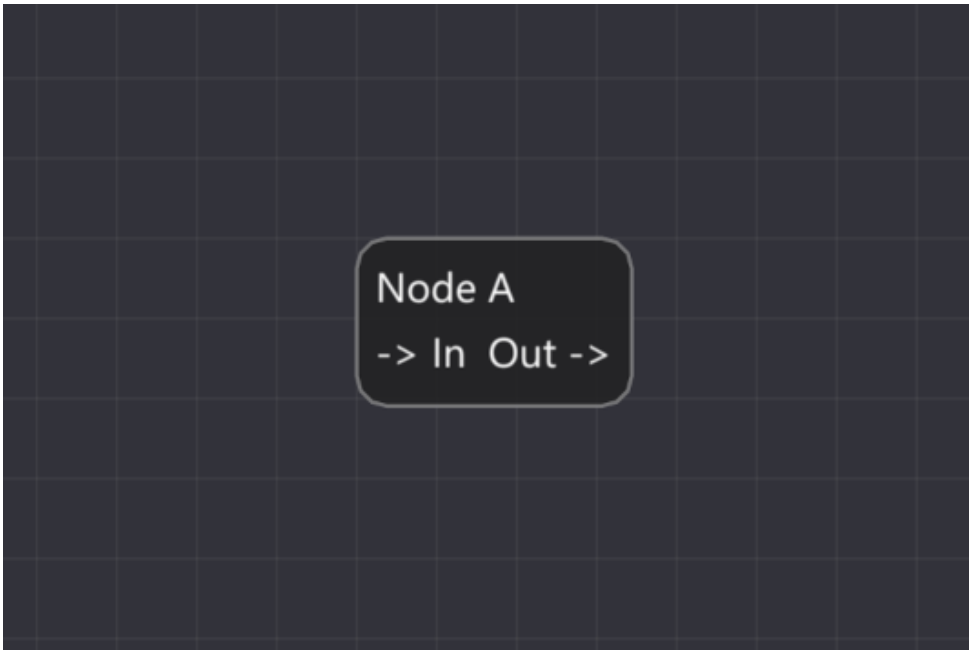
K vytvoření rozhraní v celém programu byla zvolena knihovna Dear ImGui. Pro vytvoření rozhraní Workspace byla zvolena její nadstavba a to konkretně ImGui Node Editor [5]. Tato knihovna byla inspirována node editorem, který je využíván v Unreal enginu. Mezi výhody této knihovny patří její snadné integrovaní do projektu. Mezi její nevýhody patří nulová dokumentace. Jediné možnosti, jak pochopit danou knihovnu používat, bylo možné pouze z příkladů, které tvůrce přidal do repositáře, příp. z diskuzí na Git Hubu. (V současné době autor na otázky nereaguje).

Tvorba "krabiček"byla prováděna ve spolupráci s kolegou Bc. Jaroslavem Holečkem, konkrétně se jednalo o rozvrhovaní funkcí do knihovny ImGui.

Ukázky kódu 3.1: Ukázka kódu ImGui Node Editor

```
1  # include <application.h>
2  # include <imgui_node_editor.h>
3
4  namespace ed = ax::NodeEditor;
5
6  static ed::EditorContext* g_Context = nullptr;
7
8  void Application_Initialize()
9  {
10     g_Context = ed::CreateEditor();
11 }
12
13 void Application_Finalize()
```

```
14 {
15     ed::DestroyEditor(g_Context);
16 }
17
18 void Application_Frame()
19 {
20     ed::SetCurrentEditor(g_Context);
21
22     ed::Begin("My Editor");
23
24     int uniqueId = 1;
25
26     // Start drawing nodes.
27     ed::BeginNode(uniqueId++);
28         ImGui::Text("Node A");
29         ed::BeginPin(uniqueId++, ed::PinKind::Input);
30             ImGui::Text("-> In");
31         ed::EndPin();
32         ImGui::SameLine();
33         ed::BeginPin(uniqueId++, ed::PinKind::Output);
34             ImGui::Text("Out ->");
35         ed::EndPin();
36     ed::EndNode();
37
38     ed::End();
39 }
```



Obrázek 3.1: příklad "krabičky"v knihovně ImGui Node Editor

3.1 Spolupráce s dalšími závěrečnými pracemi

Celý projekt modernizace nástroje I3T byl z důvodu obsáhlosti tématu rozdělen do několika závěrečných prací. Rozdělení projektu proběhlo následovně:

- Back-end aplikace (autor: Martin Herich)
- 3D Scéna (autor: Daniel Gruncl)
- Tutoriály (autor: Miroslav Müller)
- Workspace (autor: Sofie Šašorina a Jaroslav Holeček)

V práci již byla například zmíněna spolupráce s některými z těchto kolegů, především s kolegy Martinem Herichem a Jaroslavem Holečkem. Celý projekt taktéž navazoval na závěrečné práce z minulých let a to konkretně na práce:

- Aplikace I3T (autor: Michal Folta)
- Logovací system (autor: Uhlík F.)
- Testovaní užitečností nástroje (autor: Vit Zadina)

Kapitola 4

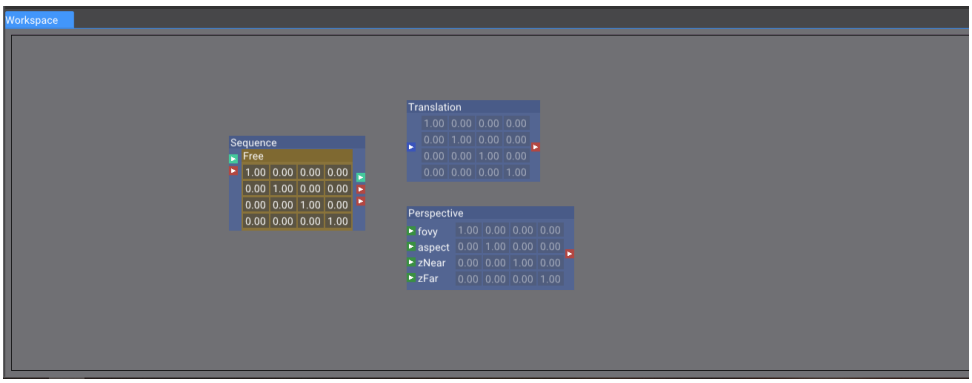
Implementace

Prvotní implementace vycházela především z příkladu knihovny Imgui Node Editor, konkrétně že souboru blueprints-example.cpp, který byl součástí knihovny. Tento příklad byl pak postupně značně modifikován. Jednalo se především o logické dělení kódu na jednotlivé funkce, protože původní kód byl z větší části napsaný v podobě jedné jediné funkce. Po rozčlenění kódu na logické prvky a seznámením s fungováním knihovny a způsobu, jak její jednotlivé části používat se přešlo k dalšímu kroku. Jednalo se o implementaci okna Workspace, ve kterém se nachází prostředí node editoru.

4.1 Workspace Window

Jedná se o základní okno node editoru. Obsahuje v sobě základní smyčku render a kontroluje frontu interakcí uživatele s node editorem.

Nejdůležitější částí této komponenty je funkce render, která je zavolána při každém snímku. Tato funkce nejprve zavolá vykreslení všech "krabiček"a poté všech spojů, kterými si "krabičky"předávají mezi sebou data. Následně zpracuje, zda uživatel nechce vytvořit nebo smazat objekt a nebo jestli nevyvolal jedno z možných vyskakovacích menu.



Obrázek 4.1: nová pracovní plocha

Zde se také nachází implementace podoby kontextového menu, které uživatel vyvolá v případe, když klikne pravým tlačítkem myši na prázdnou plochu.

V tomto menu se nachází výběr všech druhů "krabiček", které program obsahuje a také pomocné funkce. Pomocné funkce obsahují příkazy vybrat vše, invertovat výběr, přesunout se k výběru a přesunout se ke všem "krabičkám". Po vytvoření prostředí bylo možné začít do vyvolávacího menu postupně přidávat "krabičky", které byly implementovány v dalších částech práce.

4.2 Hierarchie tříd

Po prozkoumaní původního programu byly "krabičky"rozděleny do několika skupin na základě jejich datových typů, které jsou vykreslovaný na jejich výstupech. Tyto skupiny jsou:

- "krabičky"s matici,
- "krabičky"s čtyř složkovým vektorem,
- "krabičky"se tři složkovým vektorem,
- "krabičky"s kvaternionem,
- jednoduché číslo.

Nicméně všechny "krabičky"opět spojovala podobná grafická struktura. To znamenalo, že u transformace byla vykreslena hlavička a pod tím matice. U operátorů byla vykreslena hlavička, vstupy, výstupní hodnota a výstupy. Nicméně existovaly i "krabičky", které byly velice specifické, které musely být řešeny individuálně (více v kapitole 7. Problémy vzniklé během implementace, podkapitola 7.2. Nestandardní "krabičky"). Všechny nové implementované "krabičky"musí obsahovat referenci na "Jádro", aby vstupní hodnoty byly patřičně spočítány.

Hierarchie tříd je následující: Z WorkspaceNode dědí WorkspaceNodeWithCoreData. Z této třídy dědí třídy základních datových typů zmíněných vyše, jako například WorkspaceMatrix4x4 nebo WorkspaceFloat. Z těchto tříd dědí třídy pro jednotlivé "krabičky".

Mezistupeň v podobě WorkspaceNodeWithCoreData byl vytvořen pro případné modifikace programu, kdyby bylo nutné vytvořit "krabičku"bez napojení na "Jádro". Základ hierarchie vytvořil kolega Jaroslav Holeček, ve kterém byly následně upraveny vykreslovací funkce.

4.2.1 WorkspaceNode

Jedná se o základní třídu "krabičky". Obsahuje v sobě základní údaje, jako je velikost a barva "krabičky". Zadává také základní virtuální funkce. Každá "krabička"obsahuje tyto parametry a vlastní implementaci níže uvedených funkci.

Ukázky kódu 4.1: Kód třídy WorkspaceNode.h

```
40 class WorkspaceNode
41 {
```

```
42 protected:
43
44 const ne::NodeId m_id;
45 std::string m_label;
46
47 ImColor m_color; /*! \brief Color of Node */
48 ImVec2 m_size;   /*! \brief Size of box */
49 float m_touchTime;
50
51 std::string m_headerLabel;
52 ImTextureID m_headerBackground;
53
54 public:
55
56 WorkspaceNode(ne::NodeId const id, ImTextureID
      headerBackground, WorkspaceNodeArgs const& args);
57
58 WorkspaceNode(
59     ne::NodeId const id,
60     ImTextureID headerBackground,
61     std::string headerLabel = "Node",
62     std::string nodeLabel = "Node");
63
64 ne::NodeId const getId() const;
65
66 virtual std::string getHeaderLabel();
67 virtual ImTextureID getHeaderBackground();
68
69 virtual void drawNode(
70     util::NodeBuilder& builder,
71     Core::Pin* newLinkPin=nullptr,
72     bool withPins=true);
73
74 virtual void drawHeader(util::NodeBuilder& builder);
75
76 virtual void drawInputs(
77     util::NodeBuilder& builder,
78     Core::Pin* newLinkPin)=0;
79
80 virtual void drawData(
81     util::NodeBuilder& builder,
82     int index)=0;
83
84 virtual void drawOutputs(
85     util::NodeBuilder& builder,
86     Core::Pin* newLinkPin)=0;
87
88 virtual void drawMiddle(util::NodeBuilder& builder)=0;
89
90 void TouchNode(const float constTouchTime);
91
92 void UpdateTouch(const float constDeltaTime);
```

```
93
94  float GetTouchProgress(const float constTouchTime);
95
96  virtual bool dataAreValid();
97
98  };
```

4.2.2 WorkspaceNodeWithCoreData

V této třídě je už zahrnuto napojení na "Jádro". Taktéž zahrnuje naimplementované výchozí funkce na vykreslovaní vstupu a výstupu. Níže je uvedena ukázka kódu vykreslovaní výstupu (ukázka kódu 4.2). Data o množství a podobě výstupů "krabičky"taktéž se získávají z "jádra". Z tohoto důvodu nenastává problém, pokud funkci zavolá "krabička"transformace, protože přímo od "Jádra"dostane informaci, že se zde žádné výstupy nenachází, tudíž nic se nevykresli. Pokud se ovšem jedná o operátor, tak se nejprve vykreslí data dle implementace funkce konkrétní třídy. Následně se vypíše název výstupu, pokud je odlišný od názvu základních datových typů. Následně se vykreslí samotná ikonka.

Ukázky kódu 4.2: Vykreslovaní výstupního pinu

```
99  void WorkspaceNodeWithCoreData::drawOutputPin(
100     util::NodeBuilder& builder,
101     Ptr<WorkspaceCorePinProperties> const &pinProp,
102     Core::Pin* newLinkPin,
103     int outputIndex)
104 {
105
106 float alpha = ImGui::GetStyle().Alpha;
107
108 builder.Output(
109     pinProp->getId(),
110     WorkspacePinColor[pinProp->getType()]);
111
112 if(!isTransformation() &&
113     !isCamera() &&
114     !isSequence())
115     { //is Operator
116         ImGui::BeginVertical(pinProp->getNode().getId().
              AsPointer());
117         drawData(builder, outputIndex);
118         ImGui::EndVertical();
119         }
120
121 ImGui::Spring(1);
122 ImGui::PushStyleVar(ImGuiStyleVar_Alpha, alpha);
123
124
125 //label
126 if (pinProp->getShowLabel() &&
```

```
127     !isSequence() &&
128     !isCamera()){
129             if(pinProp->getLabel().empty()){
130 
131             auto label = pinProp->getCorePin().getLabel()
                    ;
132             if(label == "float" ||
133             label == "vec3" ||
134             label == "vec4" ||
135             label == "matrix" ||
136             label == "quat" ){
137                 ImGui::TextUnformatted("");
138             }else{
139                 ImGui::Spring(1,I3T::getSize(ESize::
                        Nodes_LabelIndent));
140                 ImGui::TextUnformatted(label);
141                 ImGui::Spring(1,I3T::getSize(ESize::
                        Nodes_LabelIndent));
142                       }
143 
144       }else{
145 
146         auto label = pinProp->getLabel();
147         if(label == "float" ||
148         label == "vec3" ||
149         label == "vec4" ||
150         label == "matrix" ||
151         label == "quat" ){
152             ImGui::TextUnformatted("");
153         }else{
154           ImGui::Spring(1,I3T::getSize(ESize::
                Nodes_LabelIndent));
155           ImGui::TextUnformatted(label.c_str());
156           ImGui::Spring(1,I3T::getSize(ESize::
                Nodes_LabelIndent));
157         }
158     }
159 }
160 
161 ImGui::Spring(1);
162 
163 //Icon
164 EColor type = WorkspacePinColor[pinProp->getType()];
165 EColor innerType = WorkspaceInnerPinColor[pinProp->
       getType()];
166 
167 ax::Widgets::Icon(
168     I3T::getSize(ESizeVec2::Nodes_IconSize),
169     WorkspacePinShape[pinProp->getType()],
170     false,
171         I3T::getColor(type),
172         I3T::getColor(innerType));
```

```
173
174 ImGui::PopStyleVar();
175
176 builder.EndOutput();
177
178 if(isTrackball()){
179         ImGui::EndVertical();
180 }
181
182 }
```

Tuto třídu dědí třídy pro jednotlivé datové typy: WorkspaceMatrix4x4, WorkspaceFloat, WorkspaceVec3, WorkspaceVec4 a WorkspaceQuat. Pak atypické "krabičky": WorkspaceTrackball, WorkspaceCycle, WorkspaceCamera, WorkspaceScreen a WorkspaceSequence. Příkladem implementace "krabičky", může být implementace "krabičky"s maticí, který je podrobněji rozepsán v podkapitole 4.5 Příklad implementace krabičky s maticí.

4.2.3 Příklad implementace krabičky s matici

Potřebný předpoklad pro implementaci "krabičky"s maticí byla implementace potřebných výkonných funkcí v "Jádru". Pokud byl tento předpoklad splněn bylo možné vytvořit novou třídu pro novou "krabičku". V ukázce kódu 4.5 je uvedena implementace "krabičky"operátoru Matice.

Ukázky kódu 4.3: příklad hlavičkového souboru

```
184 #include "WorkspaceMatrix4x4.h"
185
186 struct WorkspaceMatrixFreeArgs
187 {
188     WorkspaceLevelOfDetail levelOfDetail =
189         WorkspaceLevelOfDetail::Full;
190
191     std::string headerLabel = "default Matrix header";
192     std::string nodeLabel = "Matrix";
193
194     //Connect to Core
195     Ptr<Core::NodeBase> nodebase =
196         Core::Builder::createNode<ENodeType::Matrix>();
197 };
198
199 class WorkspaceMatrixFree : public WorkspaceMatrix4x4
200 {
201 public:
202     WorkspaceMatrixFree(
203         ImTextureID headerBackground,
204         WorkspaceMatrixFreeArgs const& args);
205
206     WorkspaceMatrixFree(
207         ImTextureID headerBackground,
208         std::string headerLabel ="Free",
209         std::string nodeLabel = "Free");
```

```
210
211     void drawDataSetValues(util::NodeBuilder& builder);
212
213 };
```

Pokud se jedná o operátor, který nevyžaduje žádné speciální prvky, tak do cpp souboru je nutné implementovat pouze konstruktory.

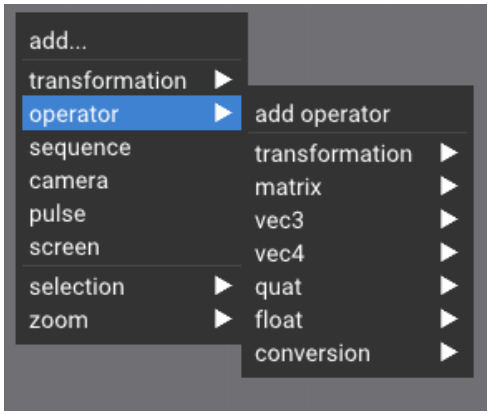
Ukázky kódu 4.4: příklad cpp souboru

```
214 #include "WorkspaceMatrixFree.h"
215
216 WorkspaceMatrixFree::WorkspaceMatrixFree(
217     ImTextureID headerBackground,
218     WorkspaceMatrixFreeArgs const& args):
219     WorkspaceMatrix4x4(headerBackground, {
220     .levelOfDetail=args.levelOfDetail,
221     .headerLabel=args.headerLabel,
222     .nodeLabel=args.nodeLabel,
223     .nodebase=args.nodebase})
224 {}
225
226 WorkspaceMatrixFree::WorkspaceMatrixFree(
227     ImTextureID headerBackground,
228     std::string headerLabel,
229     std::string nodeLabel):
230     WorkspaceMatrix4x4(
231     headerBackground,
232     Core::Builder::createNode<ENodeType::Matrix>(),
233     headerLabel,
234     nodeLabel)
235 {}
236
237 void WorkspaceMatrixFree::drawDataSetValues(util::
        NodeBuilder& builder)
238 {}
```

Nakonec je nutné přidat novou položku do vyskakovacího menu.

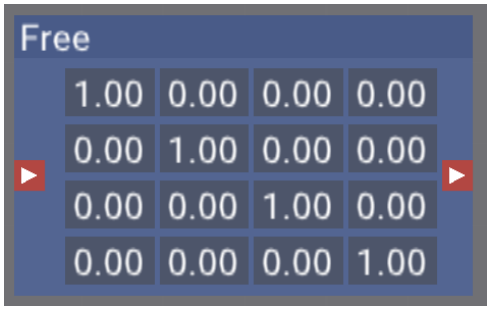
Ukázky kódu 4.5: položka krabičky v vyskakovacím menu

```
239 //....
240 if (ImGui::MenuItem("Node Label")){
241     m_workspaceCoreNodes.push_back(
242         std::make_unique<**Name of new class**>(
                m_headerBackgroundTexture));
243         ne::SetNodePosition(
244             m_workspaceCoreNodes.back()->getId(),
245             m_newNodePostion);
246 }
247 //....
```



Obrázek 4.2: vyskakovací menu

Výsledkem příkladu je takto vypadající "krabička"(viz obrázek 4.3)



Obrázek 4.3: krabička matice (Matrix)

4.3 Zpracovaní události

Potom co už některý byli "krabičky"byli implementovaný mohl se řešit uživatelský vstup. Většinu obsluhy událostí interakce s uživatelem mají v sobě implementovány knihovny ImGui a Node Editor, takže po zavolání příslušné funkce stačilo adekvátně reagovat. Ovšem nebylo možné takto postupovat vždy (viz podkapitola 4.3.1. Volání kontextových menu a kapitola 7. Problémy vzniklé během implementace).

4.3.1 Volaní kontextových menu

Bylo nutné implementovat následující kontextové menu:

- oobecné menu pro "krabičku", kde uživatel může zvolit přesnost s jakou chce vidět hodnoty, zdá chce vidět celou matici nebo jenom některé důležité hodnoty pro danou operaci, zdá chce zablokovat či povolit editaci, a nebo "krabičku"smazat.
- menu které se vyvolá při kliknutí pravým tlačítkem myši na číselnou hodnotu, a ve kterém uživatel najde užitečné a často používané hodnoty.

Knihovna imgui node editoru má v sobě implementovány volání kontextových menu pro jednotlivé objekty. A to konkretně kontrolu, zda-li uživatel klikl na jednu z níže uvedených věcí pravým tlačítkem myši:

1. Link
2. Pin
3. "Krabičku"mimo pin
4. Pozadí

Využívání této knihovny přineslo celou řadu výhod, týkajíce především urychlení práce, ale přineslo také řadu nevýhod především v podobě omezení, týkající se například tvorby vlastních objektů, které by vyvolali odlišnou reakci od výše zmíněných, bez přímého zásahu do knihovny.

Z tohoto důvodu bylo nutné pro vytvoření menu, které se objeví po kliknutí na políčko s číselnou hodnotou, přidat do "krabiček"proměnou navíc, která sleduje vstup uživatele a při volání kontextového menu, přihlíží k dané proměnné, zda se má otevřít klasické menu pro "krabičku"či ne.

Ukázky kódu 4.6: Nastavení proměnný v krabičce

```
248 //...
249 if (ImGui::IsMouseReleased(1) && ImGui::IsItemHovered(
        ImGuiHoveredFlags_AllowWhenBlockedByPopup))
250 {
251     fw.showMyPopup = true;
252     fw.id = fmt::format("##{}:r{}c{}", idOfNode, rows,
            columns);
253     fw.value = localData;
254     fw.columns = columns;
255     fw.rows = rows;
256 }
257 //...
```

Ukázky kódu 4.7: Rozhodovaní ktery menu vykreslit

```
258 //...
259 if (ne::ShowBackgroundContextMenu())
260 {
261         ImGui::OpenPopup("Create New Node");
262 }
263 else if (ne::ShowNodeContextMenu(&m_contextNodeId))
264 {
265         Ptr<WorkspaceNodeWithCoreData> node =
266             getWorkspaceCoreNodeByID(m_contextNodeId);
267         if (node->fw.showMyPopup){
268
269         "float_context_menu");
270                 node->fw.showMyPopup = false;
271
272         }else{
```

```
273                     ImGui::OpenPopup("Node Context Menu");
274             }
275     }
276     //...
```

4.4 Změny v kódu v knihovnách a utilitách

Některé featury, které bylo nutné vytvořit, ať už z pohledu funkčnosti, nebo grafického vzhledu, nebylo možné udělat bez zásahu do interních souborů knihoven. Z důvodu toho, že k potřebnimu procesu nešlo přistoupit zvenčí a nebo výsledek funkcí byl neuspokojivý. Problém je rozebrán podrobněji v následujících podkapitolách.

4.4.1 Změny v knihovně node editor

Každý datový typ obsahuje vlastní barvu ikonky. Díky tomuto formátování je možné jednoduše kontrolovat, který datový typ obsahují dané spojení mezi "krabičkami". Spojení má pak stejnou barvu, jako barva ikonky odkud spojnice vychází.

Při formátování statického vykreslovaní nebylo nalezeno mnoho problémů, především z důvodu toho, že volaní o vykreslovaní jednotlivých linek probíhá na straně I3T. Teoreticky je tedy možné některá spojení schovat a je možné si určit argument barvy spojnice.

Více problémů nastalo při formátování vykreslovaní spojení, které nastává, když uživatel klikne na vstup nebo výstup a táhne myší do prostoru. O toto vykreslování se starala samotná knihovna, a to vždy na konci snímku při volaní NodeEditor::End(). Proto bylo nutné do knihovny zařadit argument barvy ikonky. Byla proto přidána do třídy Pin uvnitř node editoru další proměnná inner_color, ze které třída Link přečte příslušnou barvu.

4.4.2 Změny v utilitě builder

Utilita builder, který využívá node editor, umožňuje zarovnaní a odsazení jednotlivých částí uvnitř "krabičky". Původní hodnoty byly moc velké, kdy jedna "krabička"zabírala velký prostor a jediná možná řešení byla zvětšit okno workspace na úkor ostatních oken v I3T a nebo oddálit pracovní plochu. Po tomto úkonu byl, ale text uvnitř "krabiček"nečitelný. Autor utility nepřipravil možnost tyto hodnoty změnit. Nepoužíval žádné proměnné a konstanty byly vepsané přímo v kódu. Jelikož byla snaha o co největší minimalistický styl, bylo potřeba tyto hodnoty změnit. Místo pevných konstant byly použity proměnné, které jsou nastavitelné ve stylu editoru.

4.4.3 Změny v utilitě Drawing

Utilita Drawing má v sobě implementováno vykreslovaní jednotlivých typů ikonek. (Enum IconType v Drawing.h) Před připravené ikonky nesplňovaly

naše požadavky, proto bylo nutné, vytvořit vlastní typ ikonky Arrow a implementovat její vykreslovaní. Po uživatelském testovaní je možná změna designu ikonky, či vytvoření dalších typů.

Ukázky kódu 4.8: Implementace vykreslovaní ikonky typu Arrow

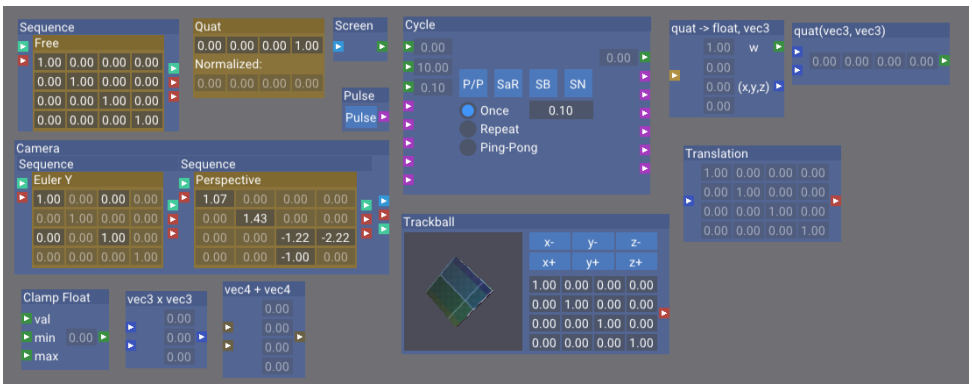
```
277 //...
278 if (filled){
279     const auto r = rect_w / 2.0f;
280 
281     const auto p0 = rect_center - ImVec2(r, r);
282     const auto p1 = rect_center + ImVec2(r, r);
283 
284         drawList->AddRectFilled(p0, p1, color, 0, 15 +
               extra_segments);
285 
286 }else{
287 
288         const auto r = rect_w / 2.0f;
289 
290         const auto p0 = rect_center - ImVec2(r, r);
291         const auto p1 = rect_center + ImVec2(r, r);
292 
293         if (innerColor & 0xFF000000)
294                 drawList->AddRectFilled(p0, p1, color, 0,
                       15 + extra_segments);
295 
296         drawList->AddRect(p0, p1, color, 0, 15 +
               extra_segments, 2.0f * outline_scale);
297 
298         const auto t0 = p0 + ImVec2(rect_w / 4.0f, rect_w
               / 4.0f);
299         const auto t1 = p0 + ImVec2(rect_w / 4.0f, (
               rect_w / 4.0f)*3);
300         const auto tTip = rect_center + ImVec2(rect_w /
               4.0f, 0.0f);
301 
302         drawList->AddTriangleFilled(tTip, t0, t1,
               innerColor);
303 }
304 //...
```

Kapitola 5

Design

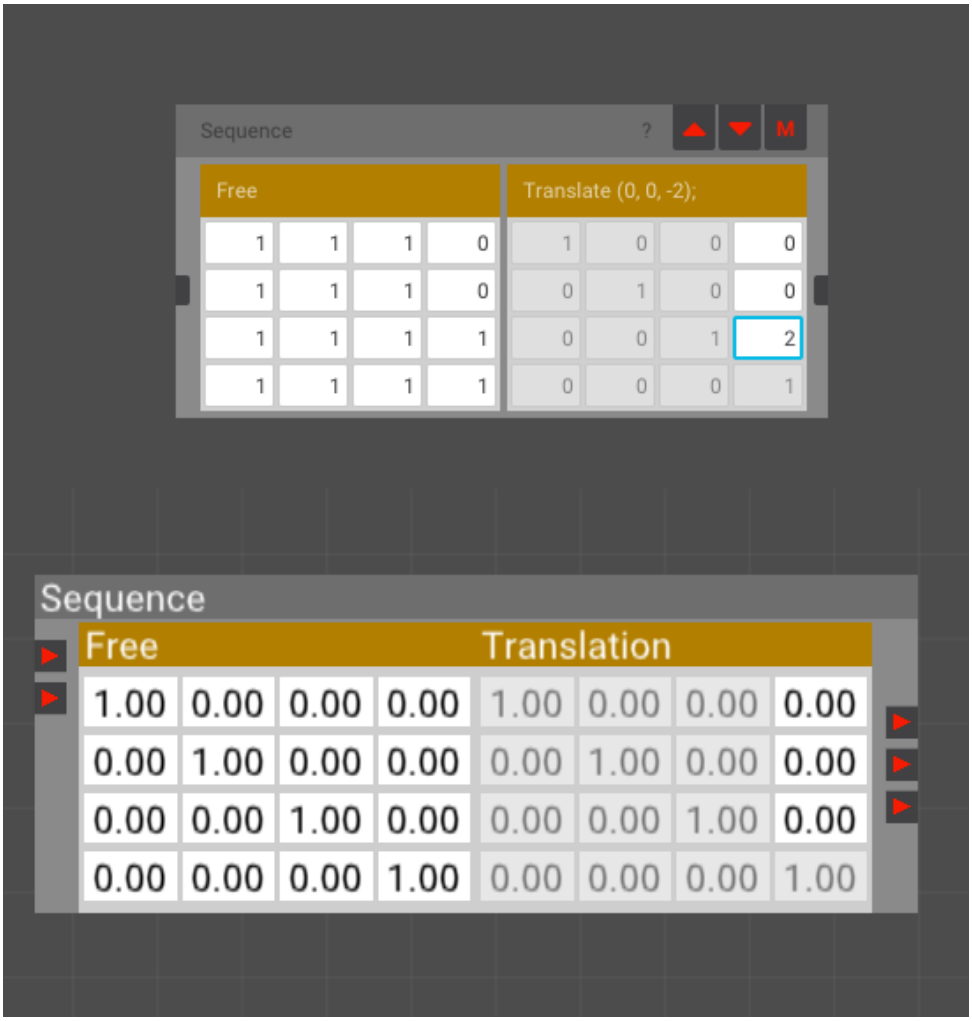
Při návrhu designu byla snaha se řídit poznámkami z práce pana Zadiny.[3] V průběhu práce byly vytvořeny dva styly, mezi kterými může uživatel přepínat.

První styl je inspirovaný původním stylem I3T. Styl je ovšem více minimalistický a barevné schéma obsahuje jemnější barvy.



Obrázek 5.1: Zástupci všech typu krabiček

Druhý styl je implementace návrhu pana Pilky, který vytvořil grafický námět pro novou podobu programu na žádost vedoucího práce pana Ing. Petra Felkela Ph.D. Konkrétní grafický námět samotných "krabiček"nebyl příliš odlišný od prvního stylu a liší se tedy především pouze barevným schématem.



Obrázek 5.2: Nahoře: návrh pana Pilky, dole: implementace stylu v I3T

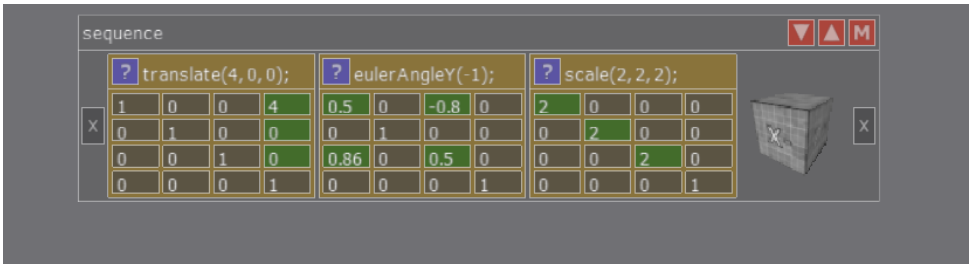
Kapitola 6

Ověření implementace na výukových scénářích

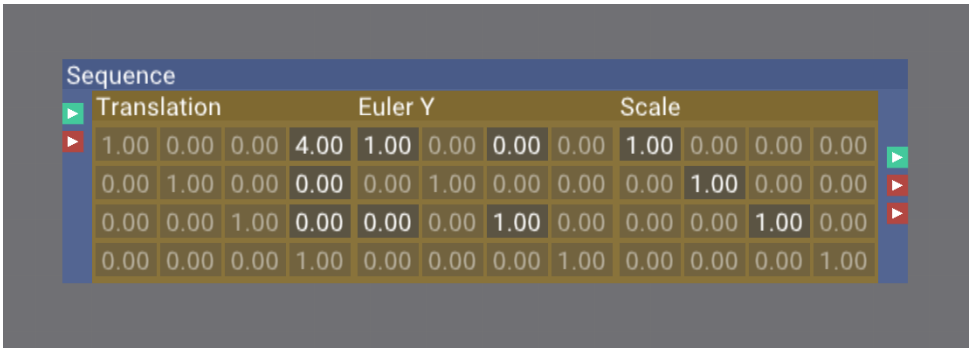
Ověření implementace na výukových scénářích mělo proběhnout v návaznosti na "Jádro"vytvořené panem Martinem Herichem. Nicméně jeho závěrečná práce byla odložena a tudíž ověření muselo probíhat za stavu, kdy "Jádro"nebylo zcela hotové. Z tohoto důvodu nebylo možné otestovat funkčnost nových "krabiček"v připravených scénářích. Některá spojení nešlo provést, neboť způsobili zavření programu. Na následujících obrázcích nalezneme především vizuální porovnaní. Samotné ověření v plné míře bude moci proběhnout, až po dokončení výše zmíněného komponentu.

6.1 Scénář 1

Výukový scénář 1 se zabývá základní aplikací translace, Eulerovy rotace podle osy Y a scale.



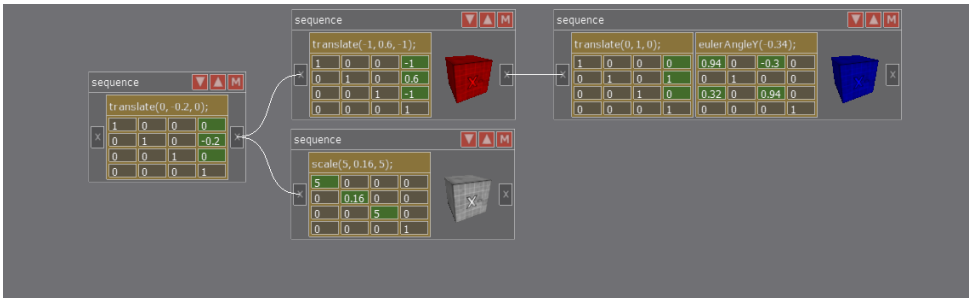
Obrázek 6.1: scenař 1. ve starém I3T



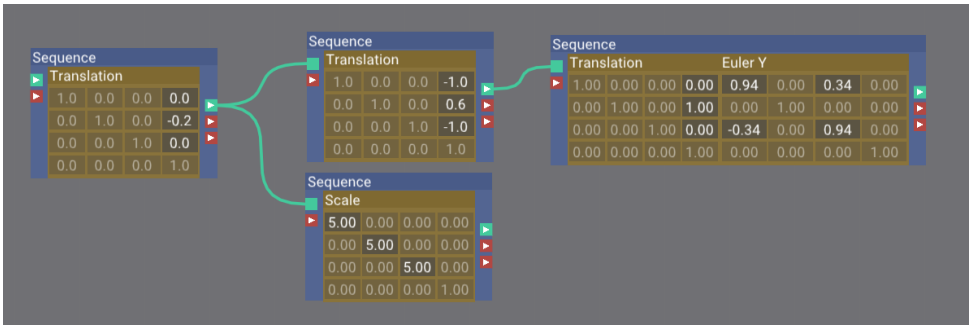
Obrázek 6.2: scenař 1. v novém I3T

6.2 Scénář 2

Výukový scénář 2 se zabývá vytvořením jednoduché scény s pomocí hierarchie transformací.



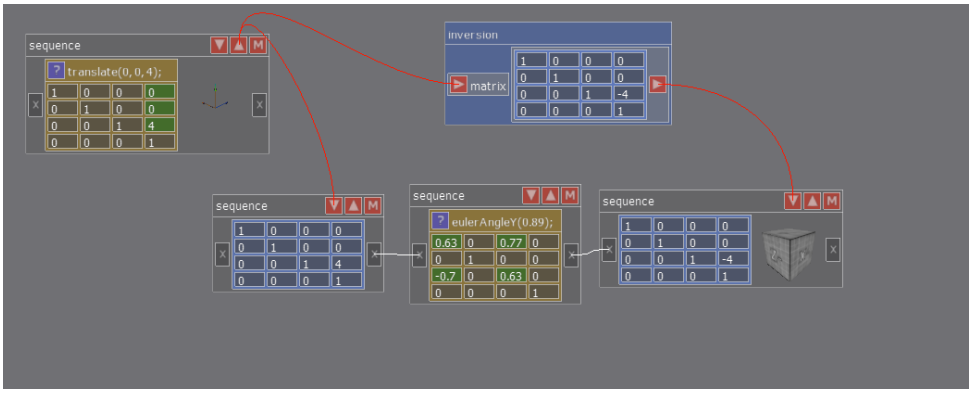
Obrázek 6.3: scenař 2. ve starém I3T



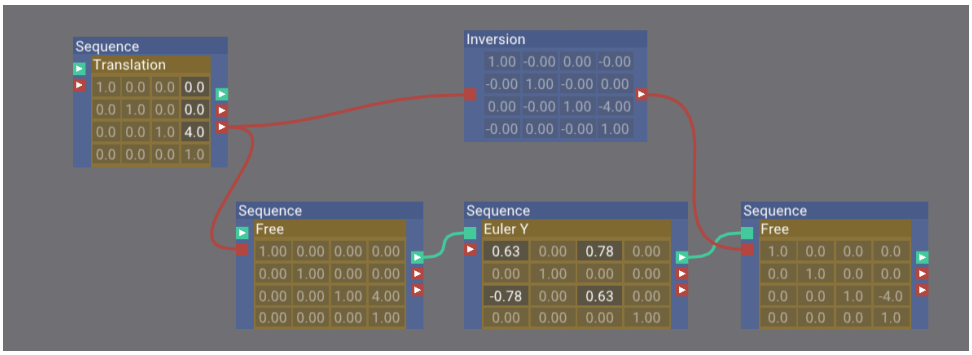
Obrázek 6.4: scenař 2. v novém I3T

6.3 Scénář 3

Výukový scénář 3 se zabývá rotací objektu kolem jednoho bodu.



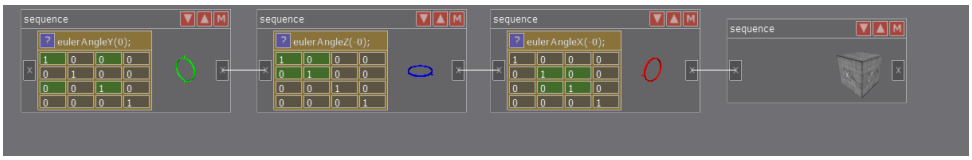
Obrázek 6.5: scenař 3. ve starém I3T



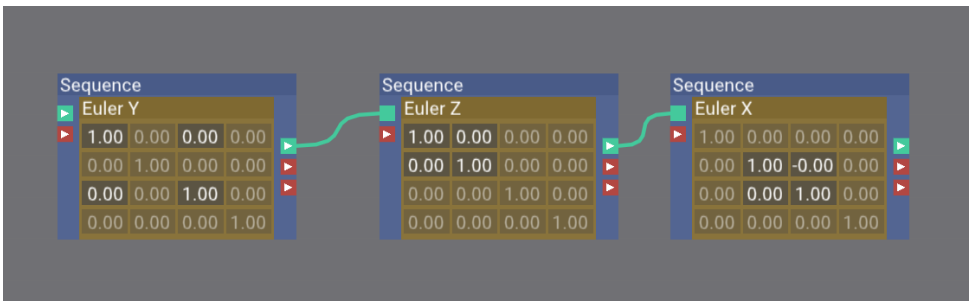
Obrázek 6.6: scenař 3. v novém I3T

6.4 Scénář 4

Výukový scénář 4 se zabývá ukázkou omezení Eulerových rotací a vytvoření gimbal-locku



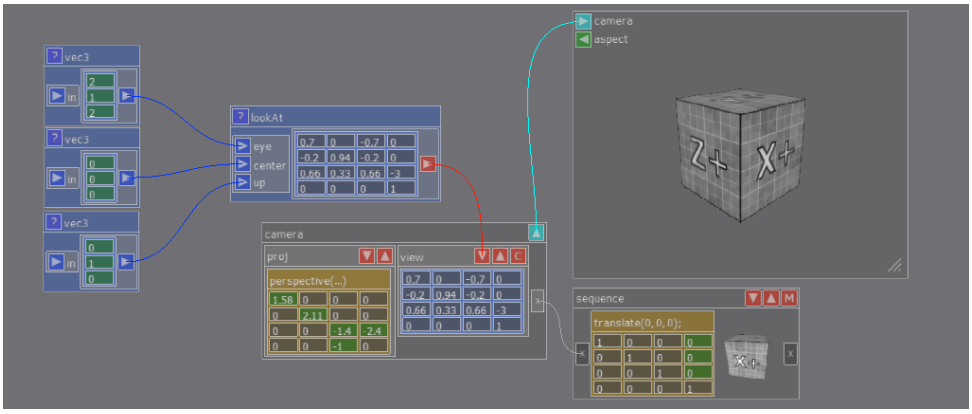
Obrázek 6.7: scenař 4. ve starém I3T



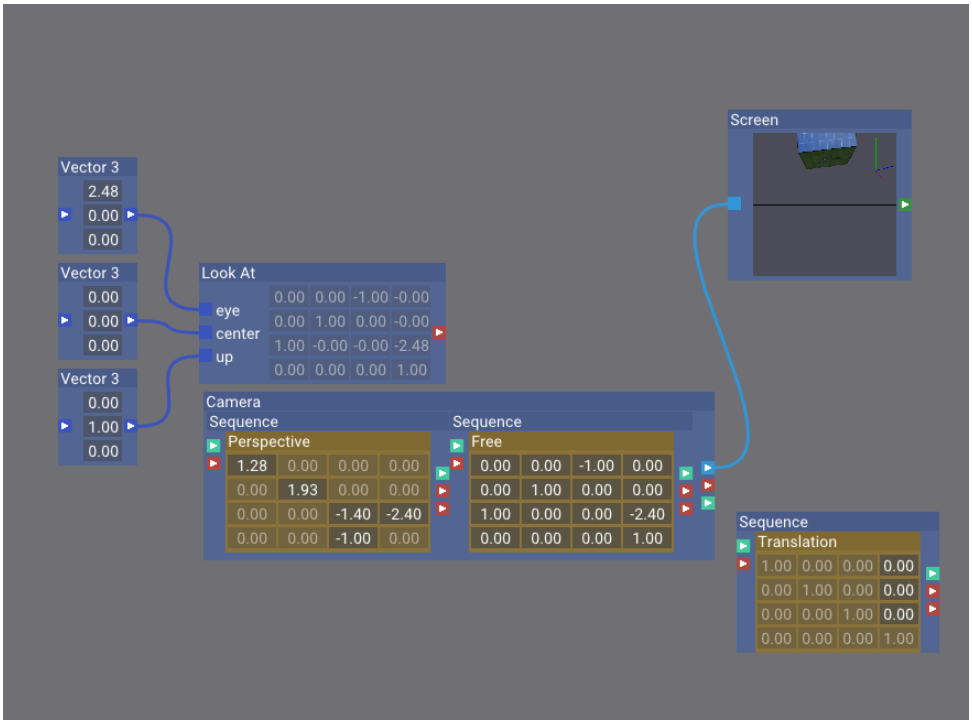
Obrázek 6.8: scenař 4. v novém I3T

6.5 Scénáře 5 až 9

Výukové scénáře 5 až 9 se zabývají procvičením různých matic projekce kamery. (Look At, Ortho, Perspective, Frustum) Scénáře 5 až 9 jsou obdobné a jejich rozdíl je pouze v jediné "krabičce", ve které se prostřídají všechny transformace projekce. Z tohoto důvodu je uveden pouze příklad scénáře 5, jako zástupce všech těchto scénářů.



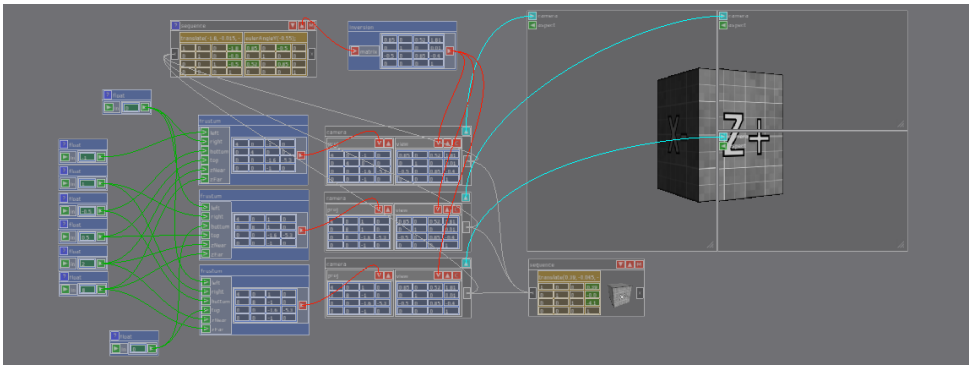
Obrázek 6.9: scenař 5. ve starém I3T



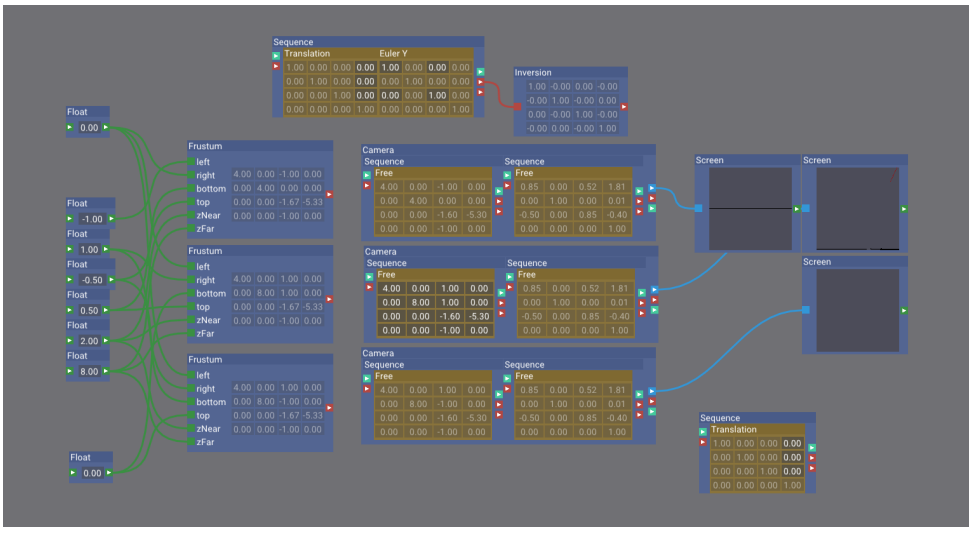
Obrázek 6.10: scenař 5. v novém I3T

6.6 Scénář 10

Výukový scénář 10 se zabývá komplexnější aplikací projekce Frustum, jehož výsledkem by měl být rozklad pohledu do scény na několik obrazovek.



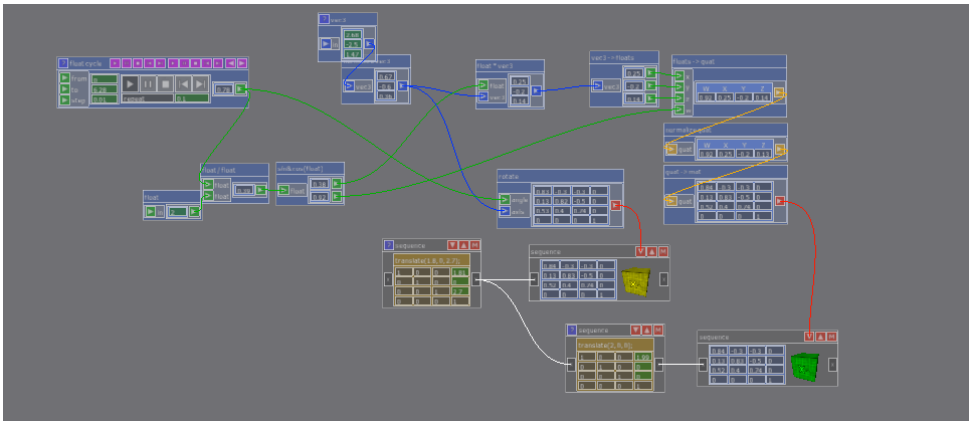
Obrázek 6.11: scenař 10. ve starém I3T



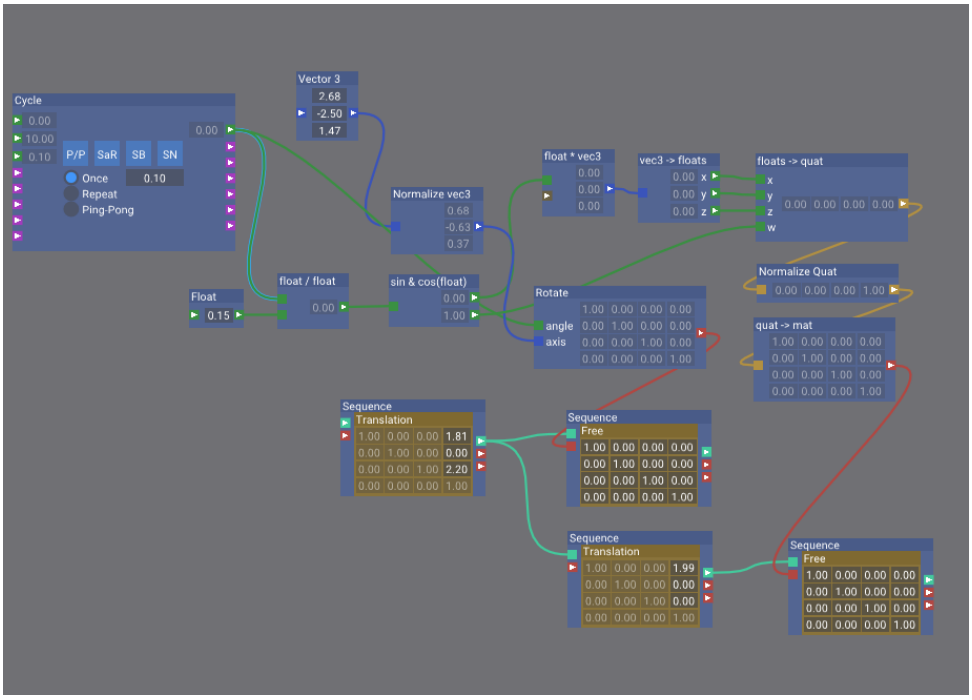
Obrázek 6.12: scenař 10. v novém I3T

6.7 Scénář 11

Výukový scénář 11 se zabývá vytvořením animace rotace objektu s pomoci kvaternionu.



Obrázek 6.13: scenař 11. ve starém I3T



Obrázek 6.14: scenař 11. v novém I3T

Kapitola 7

Problémy vzniklé během implementace

Během tvorby práce se vyskylo několik závažných problémů, které značně zpomalili vývoj programu. Jednalo se především o:

7.1 Knihovna Imgui Node Editor

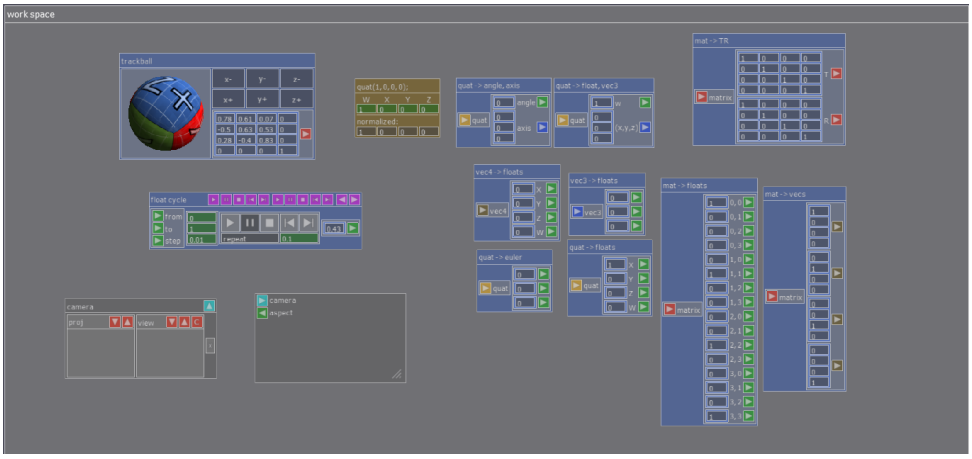
Velké problémy způsobila samotná knihovna imgui node editor. Výhoda jejího snadného nasazení do systému, byla vynegována její velkou mírou nepřizpůsobivosti, pokud uživatel požadoval větší míru stylizaci než je uvedeno v příkladech knihovny.

Ačkoliv je tato knihovna implementována s pomocí knihovny Dear ImGui, tak velká řada funkcí, především, které se týkají odsazování a zarovnávání, nebylo možné použít. Dodatečné stylizační prvky následně způsobovaly úplné rozsypání struktury "krabičky", či v horších případech i vyhození kritické chyby, což vyústilo v následné zavření programu.

Jelikož tato knihovna byla dlouho neaktualizována, bylo rozhodnuto knihovnu stáhnout, v potřebných místech upravit a odpojit od jejího repositáře na Git Hubu.

7.2 Nestandardní krabičky

Ačkoliv většina "krabiček"vypadá obdobně, nástroj I3T obsahuje i několik nestandardních "krabiček". (viz obrázek 7.1)



Obrázek 7.1: nestandartní krabičky

Mezi méně problematické patří "krabičky", které v sobě obsahují více než jeden výstup. Pro jejich fungování bylo nutné změnit způsob vykreslovaní "krabiček", protože implementace, která se starala o vykreslovaní běžných "krabiček", je nedokázala vykreslit (např: matice se rozpadla na jednotlivé vektory pod sebou), a nebo program vyhodil error a přestal fungovat.

"Krabička"cyklus, je specifická tím, že má v sobě ovládací tlačítka. Vstupy a výstupy, které se dříve nacházeli v hlavičce "krabičky"byly příliš zavádějící, proto bylo rozhodnuto v nové verzi přemístit tlačítka na levou a pravou stranu "krabičky". Problém nastal u ovladačích tlačítek a rozbalovacího menu. První řešením u tlačítek bylo využití ImGui::ImageButton, ale řešení se ukázalo nevhodné, protože node editor blokoval zachycení kliknutí na tlačítko uživatelem, ačkoliv obyčejná textová tlačítka zmačknutí zaznamenávaly. Vznikl nápad příslušné ikonky vykreslit s pomocí speciálních textových řetězců z kódovaní UTF-8. Jelikož import speciálních znaku (konkretně české diakritiky) řeši kolega Müller, bylo domluveno že to zprovozni potřebné znaky až se ktomu dostane. Na moment napsaní této práce jsou tam zatim pismenné náhrady. Klasické rozbalovací menu od ImGui bylo vykreslováno nevhodně. Proto vyběr modu bylo upraveno na radio buttony.

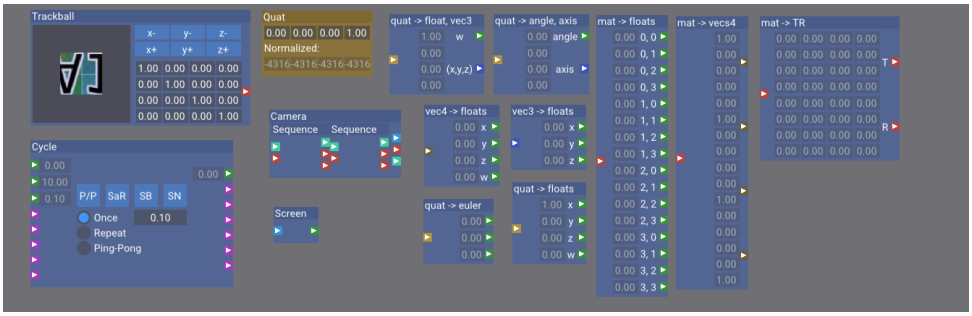
TrackBall - jako jediná "krabička"má zvláštní ovládací prvek, a tj. 3D koule, kterou je možné ovládat výstupní maticí. Zde bylo nutné implementovat zcela jiný prvek, který se potom vykresloval jako textura v "krabičce".

Screen - "krabička"atypická tím, že neobsahuje žádné ovládací prvky. Jejím účelem je vykreslit texturu na základě dat, které předá scéna jádru.

Sekvence - "krabička"specifická svojí přizpůsobivostí. Svoji velikost upravuje na základě vložených transformací. Při formátování této "krabičky"se projevily velké nedostatky knihovny Node Editor.

Kamera - "krabička"s podobným problémem, jako "krabička"sekvence. Velký problém této "krabičky"byl dán tím, že "krabička"Kamera obsahuje dvě "krabičky"Sekvence, které mají znázorňovat matici view a matici projection. Nicméně knihovnu Imgui Node Editor nebylo možné v tomto případě použít, jelikož nedokázala správně zpracovávat vstup uživatele. "Krabička"kamera,

jako jediná, není zcela funkční z hlediska grafického rozhraní, jelikož její úplná funkčnost by vyžadovala značnou úpravu knihovny Imgui Node Editor.



Obrázek 7.2: nestandartní krabičky v novém I3T

Mezi další příklady problémů je možné uvést například implementaci změnu kroku při zmačknutí různých funkčních kláves. Knihovna ImGui obsahuje zabudované chovaní u políčka float při zmačknutí příslušných kláves a žádným příkazem se nedalo toto chovaní zrušit. Proto byla provedena změna u inicializace programu. Do mapy kláves ImGui nejsou přidány klávesy SHIFT, CTRL a ALT. Pro kontrolu zmáčknutí daných kláves byl používán modul InputManager, který připravil kolega Martin Herich. To bohužel mělo za následek deaktivaci funkcí, jako je například CTRL+C. V rámci dalších optimalizací programu by bylo vhodné tyto funkce zpátky implementovat.

Kapitola 8

Závěr

Cílem závěrečné práce bylo převést všechny "krabičky"z původního I3T do nového formátu se zaměřením především na podobu a funkčnost. Implementovat návrh GUI dle studie Víta Zadiny a grafického návrhu Lukáše Pilky, a ověřit je v existujících výukových scénářích.

Výsledkem práce je nové GUI, které bere v potaz výsledky studie Víta Zadiny. Všechny "krabičky", s výjimkou "krabičky"kamera, byly úspěšné přeneseny do nového GUI. Kvůli nedostatkům knihovny Imgui Node Editor, které se zjistili až v průběhu práce, je "Krabička"kamera pouze částečné funkční. Ověření ve scénářích proběhlo pouze z vizuálního hlediska, z důvodu nehotové komponenty programu vytvářené, jako součást jiné závěrečné práce.

V průběhu práce bylo naraženo na několik specifických problémů, které jsou zmíněny ve zvláštní kapitole, jako byla například nepružnost knihovny Imgui Node Editor.

8.1 Jak by se dalo pokračovat

Problémy vzniklé při implementaci "krabičky"Kamera (kapitola 7. Problémy Vzniklé během implementace) vytváří nutnost úpravy knihovny Imgui Node Editor, aby splňovala potřeby aplikace.

Na závěr by bylo vhodné nové I3T podrobit uživatelským testům, ze kterých by vznikly nové nápady a připomínky týkající se stylizace nástroje a jeho funkčnosti.

Literatura

[1] Folta M. *Systém pro výuku transformací.* FEL ČVUT, 2016.

[2] Uhlík F. *Logovací systém pro nástroj na výuku transformací I3T.* FEL ČVUT, 2020.

[3] Zadina V. *Testování užitečnosti nástroje pro výuku transformací.* FIT ČVUT, 2020.

[4] Nechanský M. *Automatické rozmisťování propojených modulů v nástroji I3T.* FJFI ČVUT, 2019.

[5] *Node Editor using ImGui*, `https://github.com/thedmd/imgui-node-editor`, Naposledy navštíveno: 21. 5. 2021.