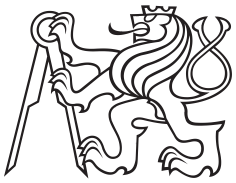
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Dokovatelná okna, historie, logování, jazykové mutace a optimalizace nástroje I3T

Matvei Korikov

Školitel: Ing. Petr Felkel, Ph.D.
Květen 2025

Poděkování

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 23. května 2025

Abstrakt

Cílem této bakalářské práce bylo rozšíření funkcionality interaktivního nástroje I3T (An Interactive Tool for Teaching Transformations), který je vyvíjen studenty Katedry počítačové grafiky a interakce FEL ČVUT. Práce se zaměřuje na implementaci ukládání a načítání pozic dokovatelných oken, optimalizaci výkonu aplikace, rozšíření systému historie akcí (podpora funkcí undo a redo), doplnění a zdokonalení logování interakcí uživatele a implementaci jazykových mutací aplikace.

Pro efektivní ukládání a načítání pozic dokovatelných oken byla využita knihovna Dear ImGui, která umožnila snadnou manipulaci s rozložením uživatelského rozhraní.

Profilování výkonu bylo provedeno jak na úrovni CPU, tak GPU, což umožnilo identifikovat slabá místa aplikace. Na základě výsledků profilování byly aplikovány optimalizační techniky, vedoucí k měřitelnému zrychlení nástroje.

Systém historie byl revidován a doplněn o chybějící funkcionality, což umožňuje plnohodnotné využití akcí kroků zpět a vpřed (undo a redo).

Nakonec byla realizována lokalizace aplikace, která nyní umožňuje přepínat mezi minimálně třemi jazykovými verzemi.

Výsledné rozšíření funkcionality přispělo ke zlepšení uživatelské přívětivosti, efektivity práce a dostupnosti nástroje I3T pro širší skupinu uživatelů.

Klíčová slova: Počítačová Grafika, Logování, Profilování, Optimalizace, Uživatelské Rozhraní

Školitel: Ing. Petr Felkel, Ph.D.
E-413,
Karlovo náměstí 13,
12000 Praha 2

Abstract

The aim of this bachelor's thesis was to extend the functionality of the interactive tool I3T (An Interactive Tool for Teaching Transformations), which is developed by students at the Department of Computer Graphics and Interaction, FEL CTU. The thesis focuses on implementing the saving and loading of positions for dockable windows, optimizing the application's performance, enhancing the action history system (supporting undo and redo functions), improving user interaction logging, and implementing language localization within the application.

The Dear ImGui library was used to effectively save and load positions of dockable windows, enabling easy manipulation of the user interface layout.

Performance profiling was conducted at both CPU and GPU levels, allowing the identification of application bottlenecks. Based on the profiling results, optimization techniques were applied, resulting in measurable performance improvements.

The history system was reviewed and supplemented with missing functionalities, enabling full support for undo and redo actions.

Finally, application localization was implemented, allowing users to switch between at least three language versions.

These enhancements significantly improved the user-friendliness, efficiency, and accessibility of the I3T tool for a broader range of users.

Keywords: Computer Graphics, Logging, Profiling, Optimization, User Interface

Title translation: Dockable windows, history, logging, language mutations, and I3T optimization

Obsah

Obrázky

Tabulky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Korikov** Jméno: **Matvei** Osobní číslo: **516195**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**

Studijní program: **Otevřená informatika**

Specializace: **Počítačové hry a grafika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Dokovatelná okna, historie, logování, jazykové mutace a optimalizace nástroje I3T

Název bakalářské práce anglicky:

Dockable windows, history, logging, language mutations, and I3T optimization

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: **12.02.2025** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **20.09.2026**

_____
podpis vedoucí(ho) ústavu/katedry

_____
prof. Mgr. Petr Páta, Ph.D.
podpis proděkana(ky) z pověření děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

_____ _____
Datum převzetí zadání Podpis studenta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Korikov** Jméno: **Matvei** Osobní číslo: **516195**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**

Studijní program: **Otevřená informatika**

Specializace: **Počítačové hry a grafika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Dokovatelná okna, historie, logování, jazykové mutace a optimalizace nástroje I3T

Název bakalářské práce anglicky:

Dockable windows, history, logging, language mutations, and I3T optimization

Pokyny pro vypracování:

I3T je interaktivní nástroj na výuku transformací vyvíjený studenty Katedry počítačové grafiky a interakce FEL ČVUT.
Nastudujte implementaci dokovatelných oken v knihovně Dear ImGui. V nástroji I3T implementujte ukládání aktuální pozice oken a nahrání zvoleného rozložení. Připravte standardní rozložení pro tutoriály, demonstrační číslované úlohy a běžný stav.
Seznamte se se způsoby profilování na CPU i GPU. Proveďte profilování aplikace I3T , detekujte slabá místa a kód optimalizujte. Změřte výsledné zrychlení.
Proveďte inventuru implementace historie (Ctrl-Z/Ctrl-Y) a doplňte chybějící části.
Dále se seznamte se stavem implementace logování [2] a doplňte chybějící interakce (např. logování změny hodnoty v políčku float). Zvažte možnost logovat pomocí skriptování.
Navrhněte a implementujte přepínání jazykové verze aplikace alespoň pro tři jazyky.

Seznam doporučené literatury:

[1] Aymar Cornut. Dear ImGui Bloat free Graphical User interface for C++. https://github.com/ocornut/imgui (Navštíveno: 6. 1. 2021). 2020.
[2] First look at profiling tools (C#, Visual Basic, C++, F#), https://learn.microsoft.com/en-us/visualstudio/profiling/profiling-feature-tour (Navštíveno: 8. 2. 2025). 2025.
[3] NVIDIA Nsight Graphics, https://developer.nvidia.com/nsight-graphics (Navštíveno: 8. 2. 2025)
[4] Martin Herich. „Restrukturalizace interaktivního nástroje na výuku transformací I3T a reimplementace grafického rozhraní pomocí knihovny Dear ImGui“. https://hdl.handle.net/10467/96746. Bakalářská práce. FEL ČVUT, 2021.

FAKULTA ELEKTROTECHNICKÁ
FACULTY OF ELECTRICAL ENGINEERING
Technická 2
166 27 Praha 6

PROHLÁŠENÍ

Já, níže podepsaný

Příjmení, jméno studenta:	Korikov Matvei
Osobní číslo:	516195
Název programu:	Otevřená informatika

prohlašuji, že jsem bakalářskou práci s názvem

Dokovatelná okna, historie, logování, jazykové mutace a optimalizace nástroje I3T

vypracoval samostatně a uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací a Rámcovými pravidly používání umělé inteligence na ČVUT pro studijní a pedagogické účely v Bc a NM studiu.

Prohlašuji, že jsem v průběhu příprav a psaní závěrečné práce použil nástroje umělé inteligence. Vygenerovaný obsah jsem ověřil. Stvrzuji, že jsem si vědom, že za obsah závěrečné práce plně zodpovídám.

V Praze dne 17.05.2025

Matvei Korikov
.....
podpis studenta

Kapitola 1

Introduction

Vývoj moderních počítačových technologií výrazně rozšířil možnosti vytváření interaktivních 3D aplikací používaných v různých oblastech - od počítačových her a systémů virtuální reality až po vědecký výzkum a vzdělávací programy. Jednou z takových aplikací je I3T (An Interactive Tool for Teaching Transformations), která je vyvíjena na katedře počítačové grafiky a interakce s cílem demonstrovat a zkoumat metody 3D transformací.

Účel a cíle práce.

Cílem práce je komplexně rozšířit funkčnost interaktivního nástroje I3T tak, aby aplikace byla rychlejší, stabilnější a uživatelsky přívětivější. K dosažení tohoto cíle je třeba splnit následující úkoly:

- **Prostudovat současnou architekturu I3T** a určit optimální místa pro zavedení nových funkcí.
- **Implementovat ukládání a načítání rozložení dokovacích oken.**

  Včetně přidaní možnosti ukládat poslední rozložení oken do souboru scény a definovaní rozložení v tutoriálu.
- **Vytvořit okno správce rozložení**, které dovolí prohlížet, přidávat a mazat uživatelská schémata rozmístění oken.
- **Provést profilování kódu** jak pomocí nástrojů třetích stran, tak i ručně psaných nástrojů.
- **Provést optimalizaci aplikace** na základě výsledků profilování
- **Obnovit a rozšířit mechanismus historie akcí**, zajistit spolehlivou funkci operací Undo a Redo ve všech scénářích.
- **Přidat systém logování uživatelské aktivity** s možností následné analýzy získaných dat.
- **Přidat plnou podporu vícejazyčnosti aplikace.**

  Lokalizovat uživatelské rozhraní do tří jazyků a vytvořit okno umožňující přepínat jazyk.

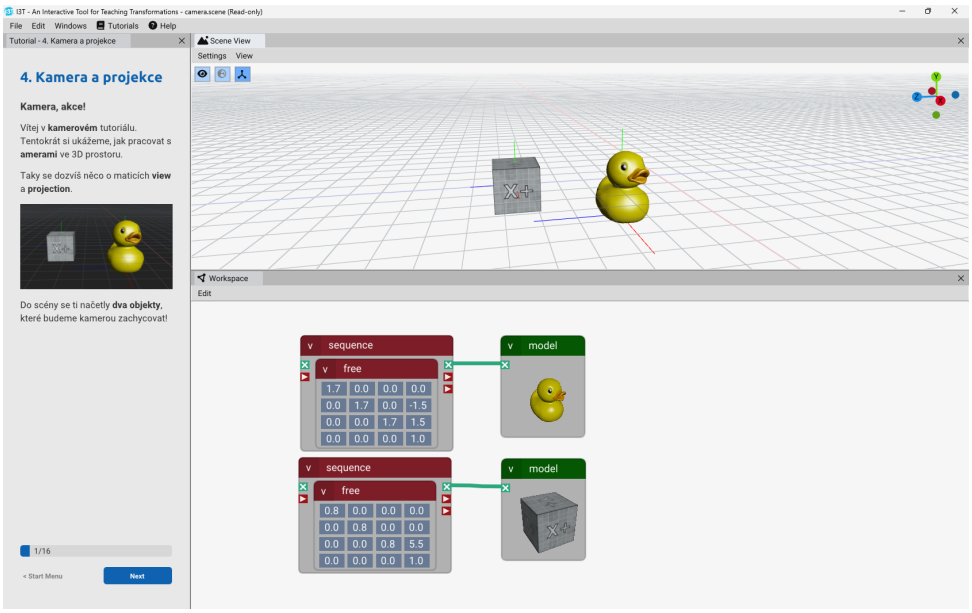
- **Realizovat uživatelské testování** nových prvků rozhraní a odstranit zjištěné nedostatky.

Kapitola 2

An Interactive Tool for Teaching Transformations, I3T

I3T [1] je program sloužící k výuce transformací v 3D prostoru a slouží jako doplněk výuky předmětu PGR (Programování grafiky), který je součástí zaměření „Počítačová grafika“ programu Otevřená informatika na ČVUT v Praze. Program byl vytvořen pro vizualizaci transformací a umožňuje interaktivní práci s transformacemi a modely.

Modely a transformace přidává uživatel ve formě propojených bloků do pracovního prostoru - interaktivního rozhraní, ve kterém uživatel vytváří trojrozměrné (3D) scény propojováním bloků v grafu scény. Modely vložené do pracovního prostoru se zobrazují v zobrazení scény, které zachycuje stav objektů po aplikaci transformace.



Obrázek 2.1: Prostředí I3T.

Klíčovou výhodou I3T je jeho interaktivita - uživatel může sám měnit hodnoty matic reprezentujících transformace a sledovat výsledky změn v reálném čase. Interakce probíhá především prostřednictvím výše zmíněných

bloků, které lze přidávat, odstraňovat, přesouvat a upravovat pomocí jejich kontextové nabídky.

2.1 Architektura aplikace I3T.

Architektura systému I3T je modulární. To znamená, že se aplikace skládá ze samostatných modulů, přičemž každý modul je zodpovědný za jinou část aplikace.

Konkrétně to jsou moduly:

- UIModule - odpovídá za sestavení a vykreslení všech grafických prvků aplikace.
- ViewportModule - odpovídá za sestavení a vykreslení okna Viewport, ve kterém se zobrazují 3D modely.
- ResourceManager - odpovídá za načítání a správu modelů, které mohou být importovány do aplikace.
- TestModule - odpovídá za testování aplikace.
- WorkspaceModule - odpovídá za okno Workspace, které obsahuje transformační uzly.
- StateManager - odpovídá za ukládání do souborů a čtení informací o scéně ze souborů.
- ScriptingModule - odpovídá za podporu skriptování v aplikaci, pomocí které je možné sestavovat uzly samostatně nebo je používat v tutoriálech.

V hlavní smyčce aplikace se tyto moduly v případě potřeby aktualizují a vykreslují. Protože většina mé práce bude sestávat z práce s grafickými prvky aplikace a z ukládání dat aplikace, jsou hlavními moduly UIModule a StateManager.

2.2 Grafická knihovna Dear ImGui.

I3T používá jako grafickou knihovnu knihovnu Dear ImGui [2]. Dear ImGui (Immediate Mode GUI) je multiplatformní knihovna s otevřeným zdrojovým kódem pro vytváření grafických uživatelských rozhraní v režimu „okamžitého“ vykreslování. Na rozdíl od „retenčních“ knihoven, kde je stav prvků uživatelského rozhraní uložen v interních datových strukturách a spravován frameworkem, Dear ImGui počítá a vykresluje každý prvek uživatelského rozhraní v každém snímku.

Kapitola 3

Ukládání pozic oken.

V rámci vývoje aplikace I3T (An Interactive Tool for Teaching Transformations) je jednou z přidaných funkcí ukládání pozic oken v souboru scény. Tato funkce dává uživateli možnost ukládat a obnovovat stav, polohu (souřadnice) a velikost oken rozhraní mezi jednotlivými spuštěními aplikace, což zlepšuje použitelnost při častém použití aplikace.

3.1 Rešerše.

V procesu implementace bylo jedním z klíčových úkolů vyřešit otázku, jak přesně by se měly ukládat pozice oken. Tento problém vyžadoval rešerši existujících technik a principů ukládání nastavení uživatelského rozhraní v různých aplikacích.

3.1.1 Analýza běžných přístupů k ukládání pozic oken.

Na začátku analýzy bylo zvažováno několik základních strategií ukládání polohy oken:

- Globální uživatelské nastavení (konfigurace aplikace).
  - Ukládání stavu všech oken do samostatného souboru, který není vázán na konkrétní scénu.
  - Výhody: jednoduchý mechanismus čtení a zápisu, nezávislost na struktuře konkrétního projektu.
  - Nevýhody: Pokud uživatel pracuje s více projekty, je nucen používat stejné umístění oken pro všechny projekty.
- Vestavěné ukládání v souboru projektu nebo scény.
  - Zápis pozic a velikostí oken uvnitř téhož souboru, kde jsou uloženy základní informace o scéně nebo projektu (jako součást jedné datové struktury).
  - Výhody: každé otevření určité scény automaticky obnoví rozhraní, pohodlí práce s příkazy, přenos souborů s uloženým nastavením.

- Nevýhody: komplikace samotné struktury souborů projektu, možné konflikty verzí.

Každý přístup má své oblasti použití a úroveň složitosti. V rámci aplikace I3T, která je vyvíjena s ohledem na práci na konkrétních 3D scénách, bylo nejlogičtější a pro uživatele nejpohodlnější vložení informací o poloze oken přímo do souboru projektu (scény).

3.1.2 Analýza přístupu použitého v programu Blender.

Jako jeden z nejlepších příkladů implementace mechanismů pro ukládání rozhraní byla zkoumána aplikace Blender [12]. V aplikaci Blender při každém uložení souboru .blend uloží nastavení uživatelského rozhraní (umístění karet, panelů, oken) jako součást projektu. Toho je dosaženo prostřednictvím následujících vlastností:

- Jednotný formát ukládání.

  Všechny informace o scéně a nastavení uživatelského rozhraní jsou zaznamenány v jediném souboru. Při otevření souboru Blender obnoví vše od geometrie objektu až po přesné umístění oken.

- Snadná spolupráce.

  Při přenosu souboru .blend jinému uživateli je vše potřebné uloženo na jednom místě. To usnadňuje reprodukci pracovního prostředí při sdílení projektů.

Studie tohoto přístupu ukázaly, že uložení stavů oken v samotném souboru scény je pohodlný způsob, jak umožnit uživatelům okamžitě se „ponořit“ do konkrétního projektu se známým rozložením oken před sebou.

3.1.3 Volba řešení: ukládání stavu oken do souboru scény.

Na základě výsledků analýzy a praktického vyhodnocení možností bylo rozhodnuto implementovat ukládání stavu oken do samotného souboru scény I3T. To znamená, že

- Při každém uložení scény se do datové struktury přidá blok obsahující informace o poloze, velikosti a dalších potřebných parametrech oken.

- Při otevření scény aplikace tato data přečte a obnoví rozhraní tak, jak je uživatel zanechal.

- Když uživatel otevře projekt na jiném počítači (pokud je k dispozici I3T a příslušná verze aplikace), získá známé rozložení oken a může okamžitě začít pracovat.

3.2 Realizace

Během vývoje systému pro ukládání pozic a stavů oken bylo zjištěno, že knihovna Dear ImGui ukládá souřadnice a velikosti oken do speciálního souboru .ini. Při každém spuštění aplikace ImGui načte údaje z tohoto souboru a v případě potřeby obnoví polohu oken. Při integraci této funkce do aplikace I3T se však vyskytlo několik problémů, které bylo třeba vyřešit.

- Žádná informace o stavu okna (otevřené/zavřené) v souboru .ini

  ImGui ze své podstaty ukládá pouze souřadnice a velikosti oken, ale nebere v úvahu logiku implementovanou nad nimi, jako je stav „otevřeno/zavřeno“. Je to proto, že ImGui pracuje v „přímém“ (okamžitém) režimu a předpokládá, že všechna okna, která chceme zobrazit v aktuálním rámci, budou deklarována přímo v kódu. Logika „skrývání“ okna, pokud není požadováno, je obvykle ponechána na samotném programu a ImGui si přítomnosti takových zaškrtávacích políček není vědomo.

- Mechanismus ukládání ImGui a použití obslužných funkcí (handlers).

  Standardní implementace ukládání v Dear ImGui zahrnuje práci s konfiguračním souborem prostřednictvím speciálních obslužných funkcí (handlers), které zpracovávají události uložení/nahrání do souboru .ini. Vývojář může přidat vlastní obslužné funkce, pokud mu standardní funkce nebudou mu vyhovovat.

  1. Při inicializaci ImGui automaticky načte údaje o poloze a velikosti oken ze souboru .ini.
  2. Při vypnutí nebo při změně rozhraní ImGui zapíše aktuální údaje do stejného souboru.
  3. Každý blok nastavení v souboru .ini je přiřazen ke konkrétnímu oknu (podle ID nebo názvu).

  Proto byla pro rozšíření informací uložených v ImGui využita možnost vytvoření vlastní obslužné funkce (handler). To umožnilo přidat do souboru .ini nová pole, která jsou zodpovědná za stav okna (isOpen=true/false), a také zpracovat tyto informace při načítání.

- Vytvoření vlastní obslužné funkce pro záznam dalších informací.

  S využitím možnosti ImGui přidávat funkce pro ukládání a načítání byla implementována vlastní funkce, která:

  1. Přistupuje k vnitřním strukturám aplikace, aby zjistil aktuální stav každého okna (otevřené/zavřené).
  2. Při zápisu dat do souboru .ini přidá příslušná metadata.
  3. Při čtení souboru načte uložené informace a předá je logice I3T, aby obnovila aktuální stav oken.

- Chybějící vazba souboru .ini na konkrétní scénu.

  Dalším problémem byla potřeba vázat uloženou konfiguraci oken na konkrétní projekt nebo scénu, nikoli na globální nastavení aplikace. Ve výchozím nastavení pracuje ImGui s jedním souborem .ini pro celý program, což neumožňuje ukládat jednotlivá nastavení rozhraní pro každou scénu zvlášť.

  V I3T je datová struktura každé scény uložena v souboru .scene, což je dokument JSON, který obsahuje informace od všech klíčových modulech aplikace. Aby bylo zajištěno, že rozhraní zůstane v kontextu konkrétní scény, bylo rozhodnuto „zapouzdřit“ výstup ImGui do samostatného bloku JSON:

  1. Mechanismus v modulu UIModule: při ukládání scény se zavolá funkce, která pomocí metody integrované v ImGui vygeneruje řetězec s údaji (obdoba údajů obvykle zapisovaných v .ini).
  2. Tento řetězec je pak umístěn do struktury JSON souboru scény, do pole „layout“.
  3. Při načítání souboru .scene se ze stejného pole „layout“ extrahuje řetězec s konfigurací oken a předá se zpět do ImGui prostřednictvím vlastní obslužné funkce (handler), která obnoví nejen polohu a rozměry, ale také stav „otevřeno/zavřeno“ pro každé okno.

  Tím bylo dosaženo cíle zajistit, aby všechny potřebné parametry oken (včetně jejich stavu) byly uloženy jako součást každého souboru .scene. Při otevření souboru scény se ImGui na základě těchto informací automaticky inicializuje a uživatel vidí rozhraní přesně tak, jak bylo zanecháno při předchozím uložení.

Kapitola 4

Okno pro manipulaci s rozložení

V rámci dalšího vývoje aplikace I3T (An Interactive Tool for Teaching Transformations) je jednou z přidaných funkcí vytvoření speciálního okna pro manipulaci s rozložením. Tato funkce má uživatelům zjednodušit práci s různými konfiguracemi rozhraní a zajistit pohodlné přepínání mezi nimi.

4.1 Účel a motivace.

Hlavním účelem zavedení nového okna je poskytnout uživateli jednoduchý a přehledný způsob, jak:

- Vytvářet vlastní rozložení uložením aktuálních umístění oken do samostatných konfiguračních souborů.
- Odstranit rozložení, které již nejsou potřeba.
- Vybrat existující rozložení, pokud chce uživatel rychle přejít na jinou konfiguraci.

Okno pro správu rozložení by se tak mělo stát intuitivním nástrojem pro rychlou změnu vzhledu aplikace, který ušetří čas a námahu při opětovném otevírání požadovaných oken na správných pozicích.

4.2 Celkový pohled na okno a jeho funkcionalita.

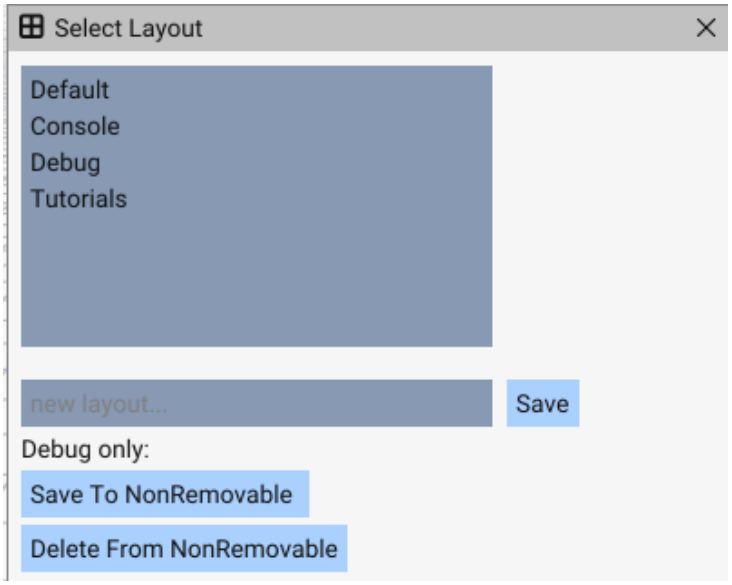
Vyvinuté okno vizuálně představuje seznam dostupných rozložení doplněný tlačítky pro mazání a vytváření nových konfigurací. Níže je uveden seznam hlavních prvků rozhraní:

1. Seznam (listbox) dostupných rozložení.
    a. Každé rozložení je reprezentováno názvem (odpovídajícím názvu souboru, v němž je konfigurace uložena).
    b. Vedle něj se může zobrazit tlačítko „Delete“, které umožňuje vybrané rozložení rychle odstranit.
2. Textové pole pro přidání nového rozložení.

a. Uživatel zadá název budoucího rozložení (např. MyLayout1).
b. Uživatel klikne na tlačítko „Save“ a program vytvoří nový soubor rozložení a přidá jej do seznamu.

3. Systém neodstranitelných rozložení.

a. Některá základní rozložení (např. výchozí) jsou chráněna před odstraněním, protože jsou součástí základní konfigurace aplikace.
b. Pokud je však I3T zkompilován v režimu ladění, uživatel má k dispozici další možnosti experimentování: mazání nebo vytváření nových „chráněných“ rozložení.



Obrázek 4.1: Select Layout okno.

4.3 Ukládání a načítání rozložení.

4.3.1 Mechanismus ukládání rozložení.

Samotná rozložení se v I3T ukládají pomocí funkcí, které poskytuje ImGui:

- Při ukládání ImGui vygeneruje soubor .ini se seznamem všech oken, jejich souřadnicemi a velikostmi.

Pro zjednodušení logiky jsou rozložení uložena v samostatných souborech .ini, jejichž názvy odpovídají rozložením zadaným uživatelem do textového pole.

4.3.2 Čtení seznamu souborů s rozložením.

Po otevření okna pro správu rozložení modul UIModule prohledá adresář /Data/Layouts, kde jsou uloženy soubory .ini rozložení, a vytvoří seznam dostupných konfigurací:

- Do seznamu jsou přidány soubory *.ini.
- Rozložení, která nelze odstranit, jsou ve vlastním adresáři /NonRemovable a jsou přidána do samostatného seznamu, odkud je uživatel nemůže v aplikaci odstranit.

4.3.3 Vytvoření nového rozložení.

Po kliknutí na tlačítko „Save“ program:

- Převezme název zadaný uživatelem (např. MyLayout1).
- Vygeneruje odpovídající soubor .ini s konfigurací okna pomocí ukládacího mechanismu ImGui.
- Přidá jméno souboru do seznamu rozložení.

4.3.4 Odstranění rozložení.

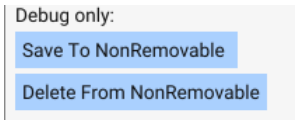
Tlačítko „Delete“ vedle každého rozložení, s výjimkou chráněných rozložení, umožňuje odstranit soubor s rozložením ze systému.

- Program kontroluje, zda se nejedná o chráněné rozložení (např. default).
- Pokud nebylo rozložení, které má být odstraněno, chráněným rozložením, rozhraní se přepne na jiné dostupné rozložení.
- Samotný soubor rozložení .ini je z disku odstraněn a položka je odstraněna z pole seznamu.

4.3.5 Funkce v režimu ladění (*Debug Mode*).

V režimu ladění se objevují další tlačítka a oprávnění (viz Obrázek 4.2):

- Odstranit a vytvořit neodstranitelné rozložení.

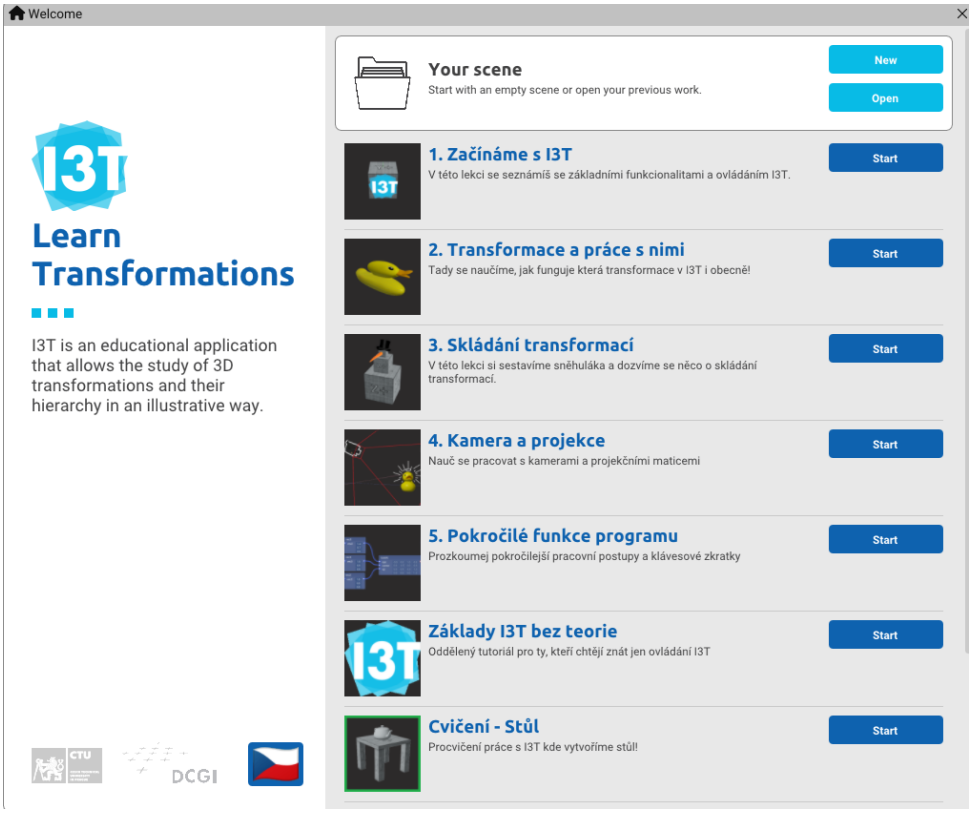


Obrázek 4.2: Tlačítka, která jsou dostupná pouze v Debug Mode.

Kapitola 5

Uložení tutoriálu do souboru scény.

Jedním z funkčních prvků aplikace I3T (An Interactive Tool for Teaching Transformations) jsou vestavěné tutoriály, které mají uživatelům pomoci naučit se rozhraní a předvést klíčové funkce aplikace a různé typy 3D transformací (viz Obrázek 5.1).



Obrázek 5.1: Welcome okno s tutoriály.

5.1 Účel a cíle.

V rámci rozšiřování funkcí systému I3T vznikla potřeba ukládat průběh tutoriálu zejména v konkrétní scéně:

1. Jazyk tutoriálu.
2. Cesta k výukovému souboru, který byl použit v aktuální scéně.
3. Číslo stránky, na které se uživatel při ukládání projektu zastavil.

Implementace takového mechanismu poskytuje velkou flexibilitu a pohodlí při pokračování výukových programů: uživatel může kdykoli projekt uložit a vrátit se k obsahu výukového programu přesně tam, kde skončil.

5.2 Formát ukládání dat.

I3T používá pro serializaci scén formát JSON, proto bylo rozhodnuto rozšířit již existující strukturu souboru scény o další pole popisující aktuální výukový program.

```
1    "tutorialData": {
2        "LanguageId": 0,
3        "CurrentTutorial": "Data/Tutorials/4TUT/4TUT.tut",
4        "CurrentStep": 4
5    }
```

5.3 Realizace mechanismu ukládání.

Funkce ukládání tutoriálu byla přidána do komponenty UIModule, která je zodpovědná za práci s uživatelským rozhraním a za serializaci řady UI parametrů ve scéně. Tento přístup je výhodný, protože modul již má informace o stavu oken, uživatelských nastaveních a může je tedy snadno doplnit o informace o aktuálním obsahu výukového programu.

5.3.1 Logika ukládání.

1. Získání údajů o aktuálním tutoriálu.

   Před uložením scény se UIModul zeptá příslušné komponenty na informace o:

   - Aktuálním jazyce tutoriálu (LanguageId).
   - Aktivní soubor tutoriálu (CurrentTutorial).
   - Aktuální stránce (CurrentStep).

2. Vytvoření struktury JSON.

   Po načtení údajů se v reprezentaci JSON scény vytvoří nebo aktualizuje blok tutoriálu.

3. Zápis do souboru .scene.

   Hotová struktura JSON včetně všech parametrů scény se uloží na disk. Při opětovném otevření souboru budou tutoriálová data k dispozici.

5.3.2 Logika načítání.

Při načítání scény ze souboru .scene:

1. Čtení bloku „tutorialData“. Krok deserializace v modulu UIModule kontroluje, zda je ve struktuře JSON přítomen objekt tutoriálu.

2. Načtení průběhu. Pokud je blok nalezen, načtou se z něj hodnoty LanguageId, CurrentTutorial a CurrentStep. Aplikace pak otevře příslušný tutoriál a nastaví aktivní stránku tutoriálu na dříve uložené číslo. Pokud tutoriál obsahoval scénu (je vždy ve formátu pouze pro čtení (Read Only)), pak bude tato scéna při načítání ignorována a uživateli se zobrazí scéna, ve které pracoval.

Kapitola 6

Možnost zadat rozložení v souboru tutoriálu.

Jedním z úkolů pro další vývoj aplikace I3T (An Interactive Tool for Teaching Transformations) bylo přidat možnost přiřadit každému tutoriálu konkrétní rozložení. To má zvýšit přehlednost a pohodlí výuky tím, že uživatel na první pohled uvidí sadu oken potřebných pro práci s konkrétním tutoriálem.

6.1 Motivace k úloze.

V předchozích verzích I3T bylo rozložení nastaveno buď ručně uživatelem, nebo pomocí výchozího nastavení. Některé výukové lekce však mohou vyžadovat specifická okna, která jsou důležitá pro úspěšné a pohodlné učení. Proto vyvstala otázka dynamického načítání vhodného rozložení při spuštění příslušného tutoriálu.

6.2 Struktura souborů s tutoriály.

Výukové programy v I3T jsou soubory YAML s příponou .tut. Každý výukový soubor obsahuje část s hlavičkou:

- Název tutoriálu.

  Zobrazuje se uživateli v seznamu dostupných tutoriálů a při spuštění tutoriálu.

- Jazyk.

  Informace o jazyce, ve kterém je tutoriál vytvořen. To umožňuje implementovat vícejazyčnost bez změny struktury aplikace.

- Scéna.

  Pokud je tento tutoriál spojen s konkrétní 3D scénou, lze přímo v tutoriálu zadat odpovídající cestu k souboru .scene.

- Popis.

  Stručný popis tutoriálu, který se zobrazí v seznamu dostupných tutoriálů.

- Obrázek.

Příklad staré struktury hlavičky YAML v souboru .tutorial (viz Kód 6.1):

```
---
title: 1. Začínáme s I3T
description: V této lekci se seznámíš se základními
    funkcionalitami a ovládáním I3T.
thumbnail: thumbnail.png
language: cs
---

...
```

Kód 6.1: Struktura hlavičky YAML v souboru .tutorial

6.3 Přidání parametru „layout“.

Aby bylo možné realizovat úlohu automatického výběru rozvržení při spuštění tutoriálu, bylo rozhodnuto přidat do souboru .tut další parametr (viz Kód 6.2):

```
---
title: 1. Začínáme s I3T
description: V této lekci se seznámíš se základními
    funkcionalitami a ovládáním I3T.
thumbnail: thumbnail.png
language: cs
layout: Data/Tutorials/1TUT/Tutorial.ini
---
...
```

Kód 6.2: Nová struktura hlavičky YAML v souboru .tutorial

V tomto příkladu layout: Data/Tutorials/1TUT/Tutorial.ini odkazuje na soubor, ve kterém je uložena konfigurace okna pro tento tutoriál. Tento přístup poskytuje:

- Flexibilitu.

  V případě potřeby lze snadno změnit rozložení na jiné rozložení pouhým zadáním jiné cesty v souboru YAML, aniž by bylo nutné měnit zdrojový kód I3T.

- Jednoduchost.

  Není třeba překompilovávat program ani provádět změny globálních nastavení - úpravy se provádějí v jediném souboru .tut.

6.4 Mechanismus načítání rozložení při spuštění tutoriálu.

6.4.1 Načtení a parsování souboru .tut.

Když uživatel vybere výukový program ke spuštění, aplikace:

1. Vyhledá příslušný soubor .tut.
2. Přečte jeho obsah pomocí parseru YAML.
3. Extrahuje hlavní pole (název, jazyk, scéna, rozložení...).

Pokud je přítomno pole „layout“ a obsahuje cestu ke konfiguračnímu souboru, aplikace může použít příslušné rozložení.

Kapitola 7

Profilování a optimalizace aplikací I3T.

V rámci další práce na aplikaci I3T (An Interactive Tool for Teaching Transformations) bylo nutné aplikaci optimalizovat a provést profilování - analyzovat činnost aplikace s cílem zjistit její výkonnost. Hlavním úkolem bylo identifikovat úzká místa aplikace, to jest části kódu, které spotřebovávají nejvíce počítačových prostředků.

7.1 Metody profilování.

Před zahájením práce na optimalizaci aplikace I3T je nutné nejprve analyzovat fungování aplikace, to jest profilování - a zjistit konkrétní místa, kde by bylo možné aplikaci dále optimalizovat.

Existuje řada způsobů a hotových nástrojů, jak provádět profilování aplikací. K prvotnímu prozkoumání a identifikaci částí kódu, které zabírají nejvíce času, je možné použít i nástroje napsané vlastními silami, které jsou integrovány přímo do aplikace. Příkladem takového nástroje je i nástroj Statistics popsaný v následující sekci a jeho funkce pro vytváření časovačů, které je možné sledovat prostřednictvím samostatného okna v samotné aplikaci I3T.

Pro podrobnější analýzu fungování aplikace je nutné využít existující nástroje speciálně určené pro profilování. Pro studii byla vybrána sada nástrojů NVIDIA Nsight [3] [4], která patří k nejpopulárnějším a plně vyhovuje potřebám. Z celé sady nástrojů nabízených společností NVIDIA budou použity nástroje Nsight Graphics [5] a Nsight Systems [6].

Nsight Graphics byl vybrán proto, že se jedná o nástroj přímo zaměřený na profilování grafických aplikací. Aplikace umožňuje analyzovat volání API grafických knihoven jednak vizualizací v textovém zobrazení (seznam volaných funkcí), jednak umožňuje zobrazit kroky při vykreslení snímku aplikace. Vzhledem k tomu, že aplikace I3T je založena na OpenGL, je Nsight Graphics ideální pro profilování na GPU.

Nsight Systems byl vybrán proto, že se jedná o rozšířenou aplikaci pro profilování na GPU i CPU. Nástroj umožňuje zaznamenat požadovaný běh aplikace a poté na grafu podrobně studovat zatížení jader CPU v určitém okamžiku, vliv dalších procesů operačního systému a systémových volání. Hlavním důvodem pro výběr aplikace Nsight Systems byla možnost zahrnout do profilovaného kódu aplikace přímo časovače, jejichž výsledky je možné

zobrazit v celkovém grafu. Princip je podobný fungování časovačů používaných v nástroji Statistics popsaném dále v textu. Přestože se nástroj Statistika dobře hodí pro zobrazení doby běhu jednotlivých částí kódu v aplikaci, není možné jej použít pro sledování doby provádění funkce, která je v jednom snímku volána několikrát, protože hodnota je pokaždé přepsána. To bylo uděláno proto, že časy časovačů jsou v okně zobrazeny jako samostatné percentilové sloupce, nikoli jako celkový graf. V opačném případě by se v okně statistik (Statistics) nacházelo obrovské množství časovačů pod sebou, což nedává představu o fungování aplikace. Nsight Systems naproti tomu umožňuje zobrazit v jednom grafu všechna volání téže funkce najednou, což je nutné v hlubších částech kódu.

7.2 Nástroj Statistics a okno se statistikou.

Nové okno zobrazuje základní statistiky o výkonu a zdrojích systému. Kromě toho mohou vývojáři vytvářet vlastní časovače měřící dobu provádění určitých fragmentů kódu a výsledky zobrazovat v přehledné formě ve stejném okně. Tato funkce umožňuje vývojářům sledovat klíčové metriky (FPS, čas snímku, využití GPU atd.) a hlouběji analyzovat výkon jednotlivých modulů.

7.2.1 Účel a cíle nástroje Statistics.

- Zobrazení klíčových metrik: okno by mělo obsahovat grafy a indikátory:
  - Aktuální FPS (počet snímků za sekundu) a historie jeho změn (ve formě grafu).
  - Čas vykreslování snímků a historie jeho změn (ve formě grafu).
  - Volnou a využitou paměť na GPU.
- Podpora nastavitelných časovačů:
  - Možnost spustit a zastavit časovač s jedinečným názvem kdekoli v kódu aplikace.
  - Zobrazení výsledku v podobě ukazatele průběhu v procentech vzhledem k celkovému času snímku.
  - Zjednodušení postupu vytváření a mazání těchto časovačů díky statickým metodám.

7.2.2 Struktura a implementace třídy Statistics.

Pro organizaci sběru a ukládání výkonnostních dat byla vytvořena speciální třída Statistics se statickými metodami a poli. Obsahuje veškerou potřebnou logiku související se sledováním FPS, časem snímků a správou časovačů. Tato architektura umožňuje centralizovat práci se statistikou a zjednodušit její připojení v libovolném místě kódu.

Třída Statistics

Hlavní funkce a pole třídy:

- FPS

  Statické pole FPS odráží aktuální počet snímků za sekundu. Je aktualizováno v metodě update(), která je volána každý snímek.

- Frame Time (DeltaTime)

  Statické pole DeltaTime uchovává dobu, která uplynula mezi jednotlivými snímky. Tuto hodnotu lze použít k vizuálnímu zobrazení délky trvání snímku.

- Timers

  Statický slovník (mapa) Timers, kde klíčem je řetězec s název časovače a hodnotou je inteligentní ukazatel na objekt TimerPtr. Při volání metod startTimer(name) a stopTimer(name) (viz Kód 7.1) se spustí a zastaví příslušný CPU časovač. Při volání metod startGPUTimer(name) a stopGPUTimer(name) (viz Kód 7.2) se spustí a zastaví příslušný GPU časovač.

- Vnořené třídy Frame a GPU.

  - Třída Frame obsahuje informace o aktuálním snímku (ID, čas strávený na CPU a GPU).
  - Třída GPU sleduje stav videopaměti (plná a volná paměť, množství alokované paměti atd.).

Volání statických metod a polí nevyžaduje inicializaci instance třídy při každém volání: například Statistics::update() nebo Statistics::FPS.

```
1  void Statistics::startTimer(const std::string& name)
2  {
3      // Získá nebo vytvoří nový časovač a spustí jej.
4      if (!CPUTimers.contains(name))
5          {
6                  CPUTimers[name] = std::make_shared<CPUTimer>(name);
7          }
8          CPUTimers[name]->start();
9  }
10
11 void Statistics::stopTimer(const std::string& name)
12 {
13     if (CPUTimers.contains(name))
14         {
15                 CPUTimers[name]->stop(); // Stop the timer
16         }
17 }
```

Kód 7.1: Funkce spuštění a zastavení CPU časovače

```
void Statistics::startGPUTimer(const std::string& name)
{
    // Získá nebo vytvoří nový GPU časovač a spustí jej.

        if (!GPUTimers.contains(name))
        {
                GPUTimers[name] = std::make_shared<GPUTimer>(name);
        }
        GPUTimers[name]->start();

}

void Statistics::stopGPUTimer(const std::string& name)
{
        if (GPUTimers.contains(name))
        {
                GPUTimers[name]->stop(); // Stop the timer
        }
}
```

Kód 7.2: Funkce spuštění a zastavení GPU časovače

Třída Timer.

Základní abstraktní třída Timer slouží k měření doby provádění jednotlivých částí kódu. Nabízí následující metody:

- start(bool sync = false) - spustí časovač.
- stop(bool sync = false) - zastaví časovač.

  Parametr `sync` určuje, zda se má před zahájením nového měření synchronizovat s GPU. Když sync == true provede se funkce glFinish(). Toto volání zablokuje vlákno CPU, dokud GPU nedokončí všechny dříve odeslané příkazy.
- get() - získá čas posledního intervalu.
- getAverage() - průměrný čas všech spuštění časovače.
- getCounter() - počet zaznamenaných intervalů.
- reset() - vynuluje nashromážděné statistiky.

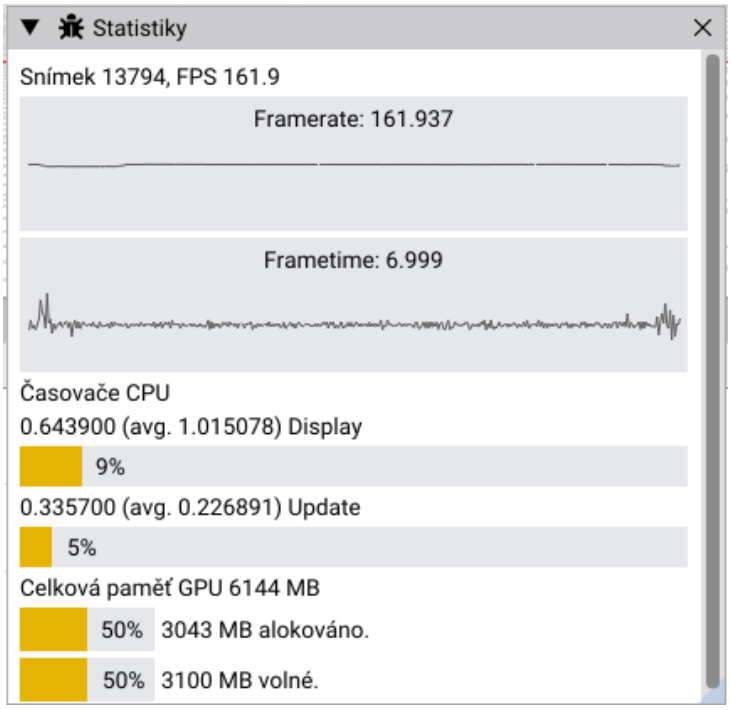
Třída CPUTimer implementuje toto rozhraní pro měření procesoru pomocí interních funkcí měření času. V případě potřeby je možné přidat další varianty časovače (např. na bázi GPU, systémových událostí atd.).

Třída GPUTimer implementuje toto rozhraní pro měření času stráveného zpracováním na grafickém procesoru (GPU). K tomu slouží funkce glCreateQueries(GL_TIMESTAMP, 1, &queryStart), která inicializuje štítek v příkazovém streamu GPU, a glQueryCounter(query, GL_TIMESTAMP), která nastavuje štítek v příkazovém streamu. Příkaz glQueryCounter(...,

GL_TIMESTAMP) je zařazen do fronty spolu s ostatními příkazy GPU. Když provádění GPU dosáhne tohoto místa, zapíše se do objektu queryStart aktuální hodnota interních časovačů GPU. Tímto způsobem je možné sledovat čas provádění úseků kódu přímo na GPU.

7.2.3 Zobrazení statistik v novém okně.

Pro zobrazení statistických informací o výkonu aplikace I3T bylo vytvořeno nové okno „Statistiky“, které zahrnuje sledování frekvence aktualizací aplikace a sledování vytvořených časovačů (viz Obrázek 7.1).



Obrázek 7.1: Okno Statistics

Okno obsahuje následující grafické prvky:

- Graf FPS.

  Zobrazuje historii změn počtu snímků za sekundu za posledních několik snímků, což umožňuje sledovat nárůsty a poklesy výkonu.

- Čas snímků (CPU)

  Ve formě grafů zobrazuje aktuální zatížení CPU a GPU během zpracování každého snímku.

- Úroveň využití paměti GPU.

  - Celková kapacita paměti.
  - Volná kapacita.

- Časovače pro vývojáře.

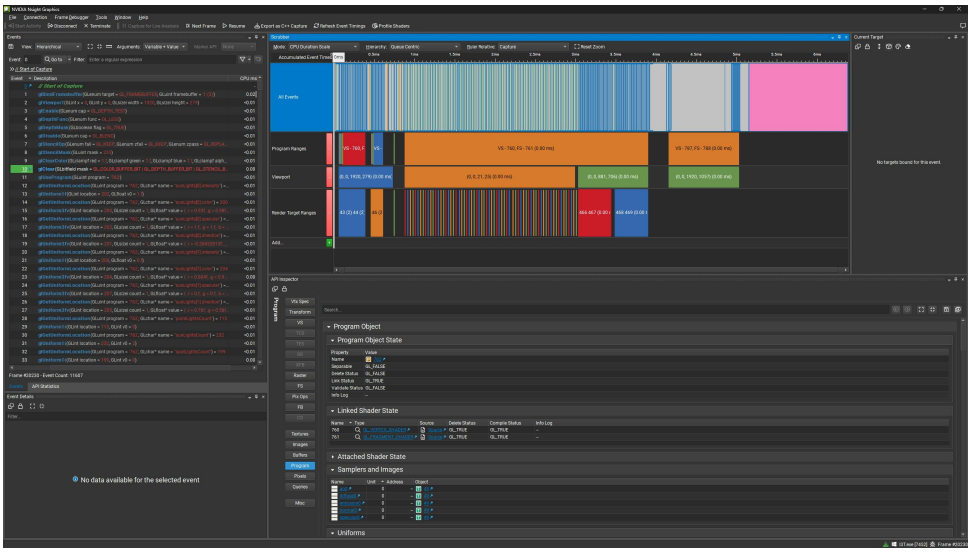
  Časovače v okně statistik jsou zobrazeny jako horizontální sloupce indikátorů umístěné jeden pod druhým, které ukazují procento trvání daného fragmentu kódu ve vztahu k celkovému času snímku. To značně zjednodušuje detekci „úzkých míst“ v aplikaci.

7.3 Profilování.

Důvodem provedení profilování bylo zjistit, co způsobuje snížení výkonu aplikace při otevírání scén obsahujících velký počet uzlů v pracovním prostoru (okno Workspace). Bylo nutné objektivně zaznamenat, která místa v aplikaci spotřebovávají nejvíce prostředků, aby bylo možné formulovat hypotézy o úzkých místech a stanovit priority další optimalizace.

7.3.1 Profilování pomocí nástroje Nsight Graphics.

Pro určení možné příčiny snížení výkonu aplikace bylo nutno zjistit, zda nejvíce času snímku zabírá přímé vykreslování uzlů v pracovním prostoru na GPU. Ke zjištění se dobře hodí nástroj pro profilování aplikací na GPU - Nsight Graphics. Nsight Graphics umožňuje zaznamenat a analyzovat libovolný snímek v aplikaci (viz na Obrázku 7.2).



Obrázek 7.2: Příklad výsledku práce Nsight Graphics

Účelem profilování pomocí nástroje Nsight Graphics v aplikaci I3T bylo zjistit, zda se ve snímku nevyskytují zbytečná volání funkcí rozhraní OpenGL API, která by měla za následek prodloužení celkové doby trvání snímku a stanovení celkového času potřebného k vykreslení snímku přímo na GPU.

Zkoumání získaných výsledků provedené na scéně obsahující 300 uzlů v pracovním prostoru ukázalo, že:

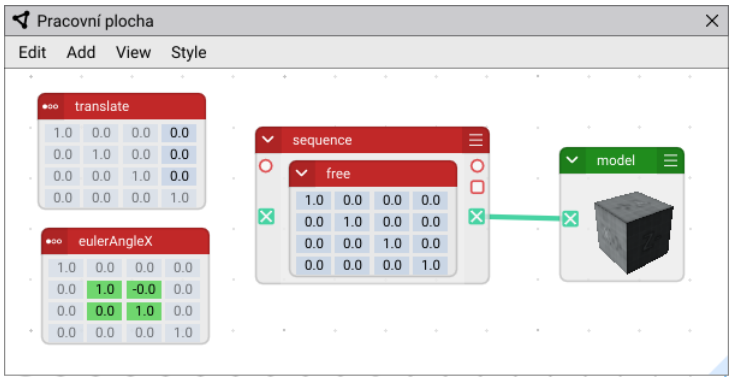
- Aplikace neobsahuje chybná nebo zbytečná volání funkcí OpenGL API, která způsobují snížení výkonu při vykreslování grafického uživatelského rozhraní pomocí knihovny Dear ImGui.
- Celkový čas strávený vykreslováním přímo na grafické kartě činil 25% (6,5 ms z celkové doby trvání snímku 26 ms).

Z analýzy výsledků profilování grafického procesoru získaných pomocí nástroje Nsight Graphics vyplývá, že vykreslování grafického rozhraní aplikace I3T není místem, které zabírá nejvíce času. Z toho byl vyvozen závěr, že je nutné provést další profilování na procesoru.

7.3.2 Profilování pomocí nástroje Statistics.

Vzhledem k tomu, že dalším krokem profilování aplikace I3T byla počáteční analýza výkonu procesoru, byl pro tento účel zvolen ručně psaný nástroj Statistics, protože umožňuje snadno přidávat časovače do kódu I3T a okamžitě pozorovat výsledky ve vizuální podobě.

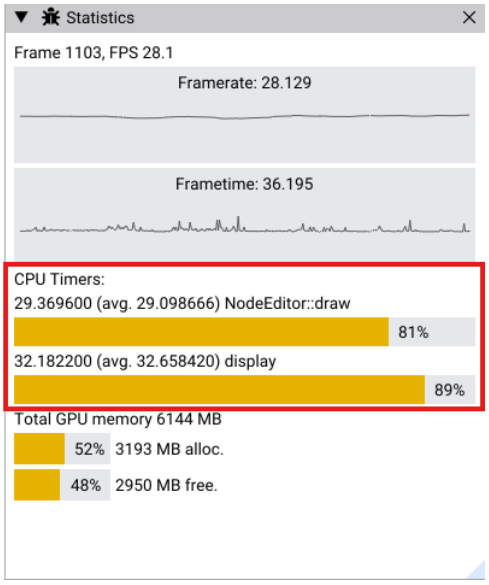
Pro určení částí kódu, které zabírají nejdelší čas snímku, bylo provedeno postupné přidávání časovačů s postupným pronikáním do kódu aplikace. Výsledky ukázaly, že takovým místem je funkce NodeEditor::draw(), která zajišťuje sestavení všech uzlů v pracovním prostoru (viz Obrázek 7.3 s příkladem uzlů v pracovním prostoru). Ačkoli název této funkce obsahuje slovo „draw“ (nakreslit), ve skutečnosti v ní k žádnému přímému vykreslování na grafickou kartu nedochází. Zvláštností grafické knihovny Dear ImGui je totiž to, že všechny grafické prvky jsou nejprve sestavovány pomocí funkcí poskytovaných knihovnou (např. ImGui::Checkbox a další), a teprve na konci snímku jsou odesílány ke zpracování na GPU prostřednictvím volání funkcí ImGui::Render() a ImGui_ImplOpenGL3_RenderDrawData(ImGui::GetDrawData()).



Obrázek 7.3: Uzly v okně pracovního prostoru

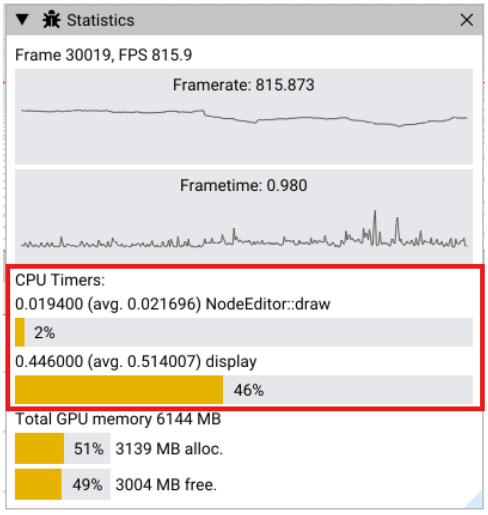
Podrobné zkoumání výsledků získaných ze scény, která obsahuje 300 uzlů v pracovním prostoru, ukazuje, že provedení funkce NodeEditor::draw() zabere 81% celkového času snímku (29,1 ms z 36,2 ms času snímku) (viz Obrázek 7.4 časovač NodeEditor::draw). Funkce display(), která následně volá funkci NodeEditor::draw(), zabere 89% celkového času snímku (viz Obrázek 7.4

časovač display). Tato funkce provádí sestavení všech grafických oken a obsahu aplikace I3T. Z toho vyplývá, že sestavení ostatních oken, jako jsou Viewport, StartWindow, TutorialWindow, zabere pouze 8% celkového času snímku a není slabým místem aplikace. Zbývajících 11% času snímku zabírá příprava aplikace na spuštění snímku, aktualizace logické části a vykreslování na grafické kartě.



Obrázek 7.4: Zobrazení funkcí v okně Statistics ve velké scéně

V případě, že scéna nebude obsahovat žádné uzly v pracovním prostoru, zabere funkce NodeEditor::draw() pouze 2% času celého snímku (viz Obrázek 7.5 časovač NodeEditor::draw). A vykreslení všech oken aplikace (všech grafických prvků) ve funkci zobrazení zabere 46% (viz Obrázek 7.5 časovač display).



Obrázek 7.5: Zobrazení funkcí v okně Statistics v prázdné scéně

Počáteční analýza provozu aplikace I3T ukázala, že část kódu, která zabírá nejvíce času snímku, je vykreslování všech uzlů v pracovním prostoru. Další hloubkové zkoumání funkce NodeEditor::draw() není možné pomocí nástroje Statistics, protože není možné sledovat celkový čas provádění funkce, pokud je volána více než jednou za snímek. Funkce NodeEditor::draw() volá funkci drawDiwne(), která je rekurzivně volána pro všechny objekty, které jsou v pracovním prostoru a tvoří všechny uzly. Pro další analýzu funkce drawDiwne() je nutné se obrátit na nástroje pro profilování aplikací třetích stran.

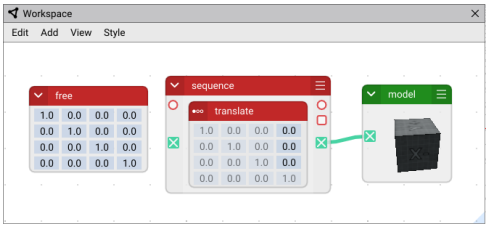
7.3.3 Profilování pomocí nástroje Nsight Systems.

V předchozí fázi profilování aplikace bylo zjištěno, že sestavení všech uzlů v prostoru Workspace zabere nejvíce času a je úzkým místem. K podrobné identifikaci problematických úseků kódu byl použit nástroj Nsight Systems, který profiluje aplikaci na procesoru a zobrazuje výsledky v přehledném grafu. Nsight Systems umožňuje přidávat do profilovaného kódu aplikace vlastní objekty časovače, což bylo využito.

Jak už bylo uvedeno, téměř veškerý čas snímku zabere funkce NodeEditor::draw(), která začne rekurzivně skládat všechny objekty Workspace pomocí funkce drawDiwne(). Vytvořením časovačů pomocí objektů nvtx3::scoped_range z knihovny nvtx3 dodávané s Nsight Systems se dalo hlouběji ponořit do analýzy funkce drawDiwne().

Obrázek 7.6: Graf celého snímku aplikace I3T

Obrázek 7.6 ukazuje příklad práce systému Nsight Systems, který v grafu zobrazuje časovače nvtx3::scoped_range přidané do kódu I3T. V tomto příkladu, který obsahoval čtyři uzly scény - free, model, sequence a translate (viz Obrázek 7.7) - zabírá téměř veškerý čas snímku skládání středového obsahu uzlu v TransformationBase::centreContent(). TransformationBase::centreContent() je funkce, která skládá obsah všech základních uzlů v aplikaci.

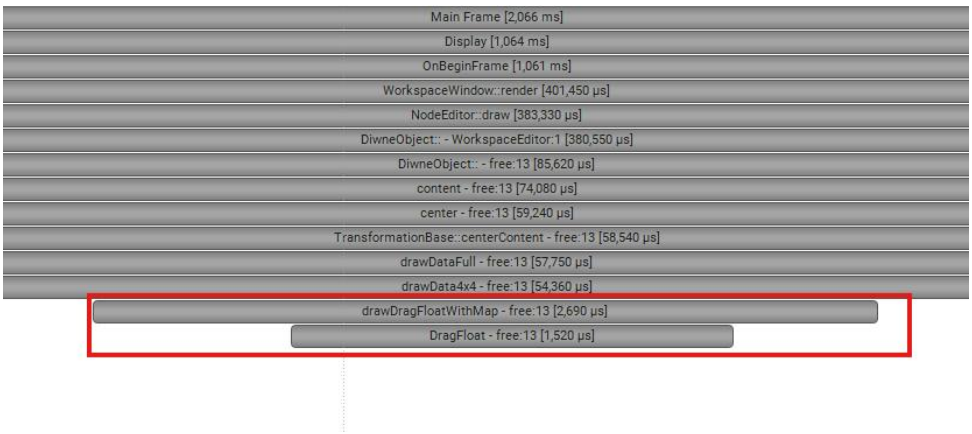


Obrázek 7.7: Příklad profilované scény se 4 uzly

Při podrobném zkoumání konkrétního uzlu je vidět, že celý čas snímku je tvořen voláním funkce drawDragFloatWithMap(), která je určena k vykreslování obdélníků s hodnotami v matici takových uzlů, jako jsou free, translate, scale atd. (viz Obrázek 7.8).

Obrázek 7.8: Graf jednoho uzlu v aplikaci I3T

Dále bylo zjištěno, že 56% (1,52 mikrosekundy z 2,69 mikrosekund) času funkce drawDragFloatWithMap() zabírá volání funkce DragFloat(), která vytváří přetahované pole hodnot z knihovny Dear ImGui (viz Obrázek 7.9). Zbytek času zabere definování vizuální reprezentace buňky s hodnotou a zpracování logiky při změně hodnoty. Vzhledem k tomu, že jde o tak malé časové intervaly, jako jsou mikrosekundy, dá se očekávat, že i funkce z knihovny Dear ImGui zabere velké procento času provádění funkce. Vezmeme-li v úvahu, že pro jeden uzel free je třeba nakreslit matici 4x4, což dává 16 volání funkce DragFloat() jen pro jeden uzel, je snadno pochopitelné, že ve scéně obsahující mnoho uzlů bude takových volání obrovské množství.



Obrázek 7.9: Graf pro funkci drawDragFloatWithMap()

7.3.4 Závěry z provedeného profilování I3T.

Bylo provedeno komplexní profilování aplikace I3T, které zahrnovalo analýzu výkonu vykreslování na GPU i výkonu na CPU pomocí nástrojů třetích stran a nástrojů napsaných vlastními silami. Výsledky ukázaly následující:

- Aplikace pracuje s GPU optimálně. Vykreslování pomocí GPU není úzkým místem aplikace.
- Většinu času snímku zabere sestavení uzlů v pracovním prostoru. Úzkým místem je volání funkce drawDragFloatWithMap(), která je zodpovědná za zobrazení buňky s hodnotou v uzlech obsahujících matice.

Získané výsledky umožňují další optimalizaci aplikace I3T snížením času potřebného k provedení funkce drawDragFloatWithMap().

7.4 Optimalizace aplikace I3T.

Po profilování aplikace I3T byl dalším krokem pokusit se aplikaci optimalizovat, aby se zvýšil výkon a snížilo zatížení procesoru a grafické karty.

7.4.1 Optimalizace vykreslování uzlů.

Hlavním cílem bylo snížit počet volání funkce DragFloat() z knihovny Dear ImGui, která zabírala 56% času provádění funkce drawDragFloatWithMap(). Způsoby, jak toho je možné dosáhnout:

1. Nevykreslovat uzly, které se nacházejí mimo viditelnou oblast.

   Vzhledem k tomu, že pracovní plochu je možné procházet kliknutím pravým tlačítkem myši a zvětšováním/zmenšováním pomocí kolečka myši, je mnoho uzlů při práci v aplikaci mimo zorné pole uživatele. Logika ignorování takových uzlů je však již implementována v I3T, což umožňuje zvýšit FPS při zvětšování pracovního prostoru.

2. Nevolat funkci ImGui::DragFloat(), pokud je vzdálenost v pracovním prostoru taková, že hodnoty v uzlech již nejsou vidět a nelze s nimi manipulovat (viz Obrázek 7.10a a Obrázek 7.10b). Tím se sníží počet volání drahé funkce, aniž by se ztratil vzhled aplikace. Tato metoda je vhodná i proto, že výkonnostní problémy začínají, když uživatel pracovní plochu hodně oddálí a zachytí mnoho uzlů, z nichž každý je třeba zpracovat. Tato optimalizace bude přímo řešit zlepšení výkonu v případě velkého oddálení pracovního prostoru.

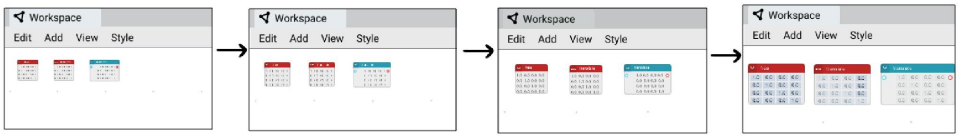
Bylo rozhodnuto implementovat druhý způsob optimalizace I3T - odmítnutí volání funkce ImGui::DragFloat(), když je pracovní plocha daleko. Nestačilo však odmítnout volání funkce, protože v takovém případě by výplň uzlů zmizela, což by je donutilo se sbalit. Bylo rozhodnuto místo funkce ImGui::DragFloat() volat funkci ImGui::Text(), když je vzdálenost pracovního

prostoru menší než 0,45f (hodnota byla zvolena experimentálně na základě kvality a čitelnosti výsledného obrázku), kde 0.25f je maximální možná vzdálenost a 4.0f je minimální. (viz Kód 7.3, Obrázek 7.10)

```
bool DataRenderer::drawDragFloatWithMap_Inline(...)
{
    if (WorkspaceModule::g_editor->canvas().getZoom() < 0.45f)
    {
       ImGui::Text("%.*f", numberOfVisibleDecimals, value);
       return false;
    }
    ...
}
```

Kód 7.3: Zjednodušená funkce drawDragFloatWithMap()

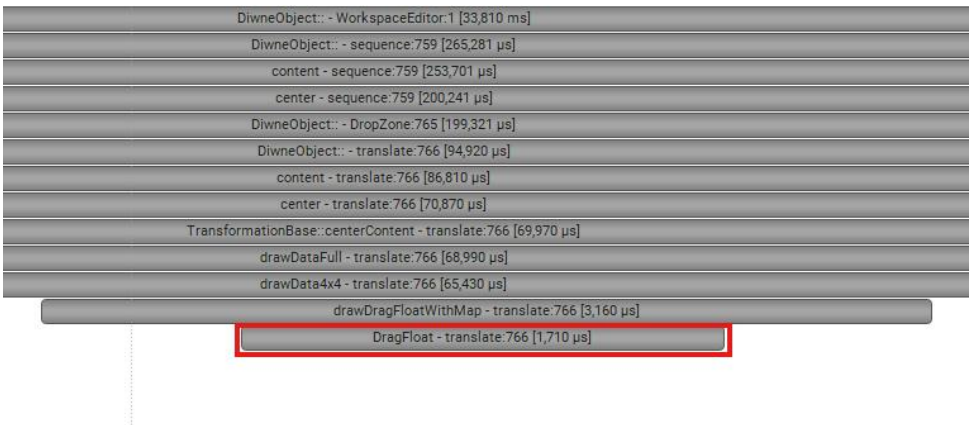
(a) : Přibližování uzlů bez optimalizace



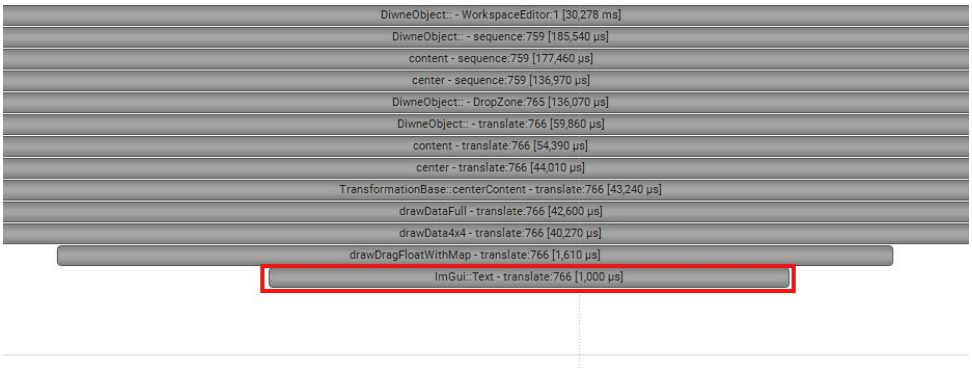
(b) : Přibližování uzlů s optimalizací

Obrázek 7.10: Přibližování bez a s optimalizací

Funkce ImGui::Text() byla vybrána jako náhrada funkce ImGui::DragFloat(), protože její doba provádění je 1 mikrosekunda, zatímco doba provádění funkce ImGui::DragFloat() je 1,7 mikrosekundy. (viz Obrázek 7.11 a Obrázek 7.12)

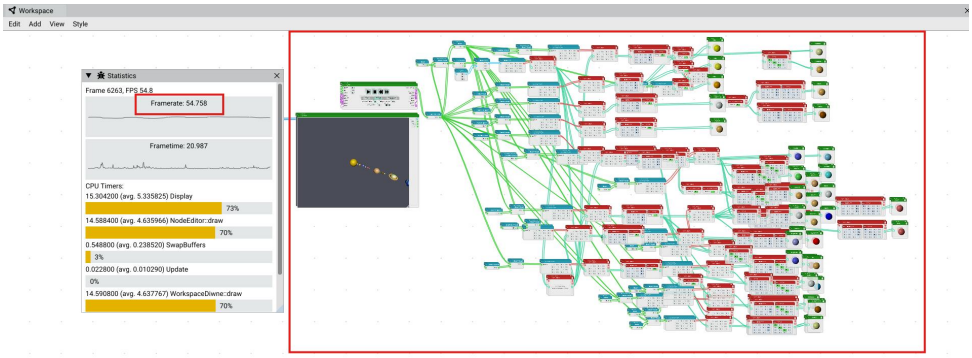


Obrázek 7.11: Graf funkce ImGui::DragFloat()

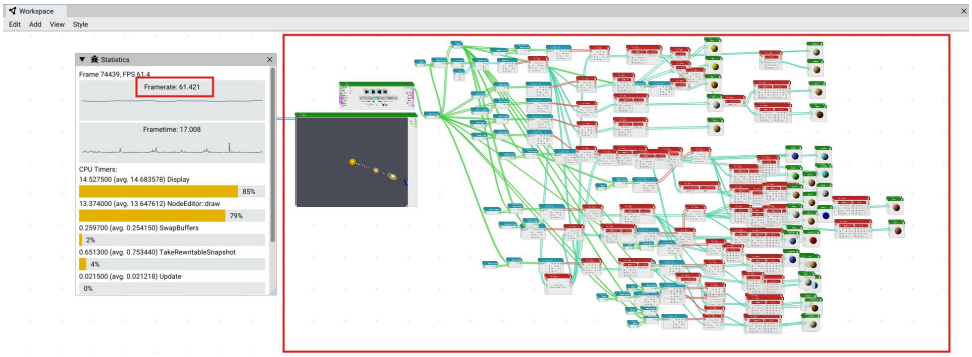


Obrázek 7.12: Graf funkce ImGui::Text()

Tímto způsobem se podařilo dosáhnout zvýšení výkonu aplikace. V náročné scéně obsahující 300 uzlů v pracovním prostoru došlo k nárůstu z 54 snímků za sekundu na 61 snímků za sekundu, což odpovídá 12% nárůstu. (viz Obrázek 7.13)

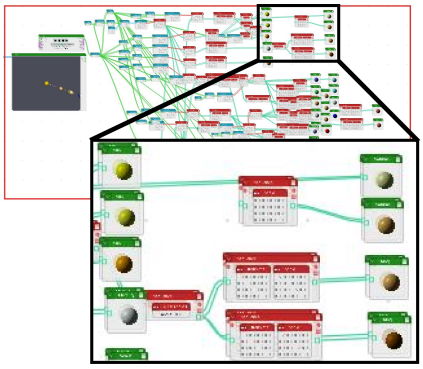


(a) : Scéna před optimalizací

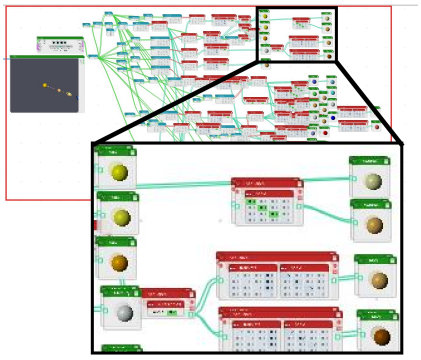


(b) : Scéna po optimalizaci

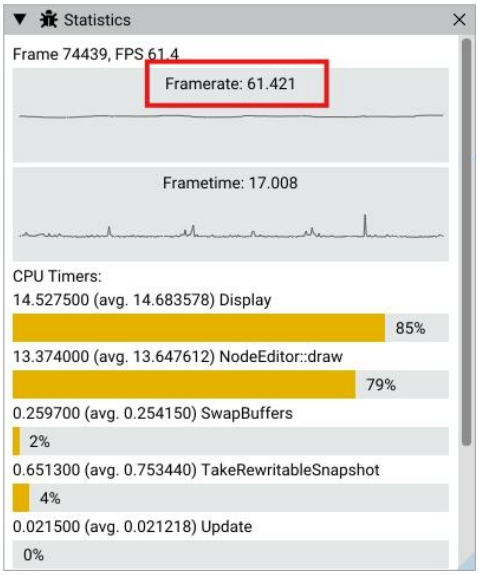
Obrázek 7.13: Scéna před a po optimalizaci



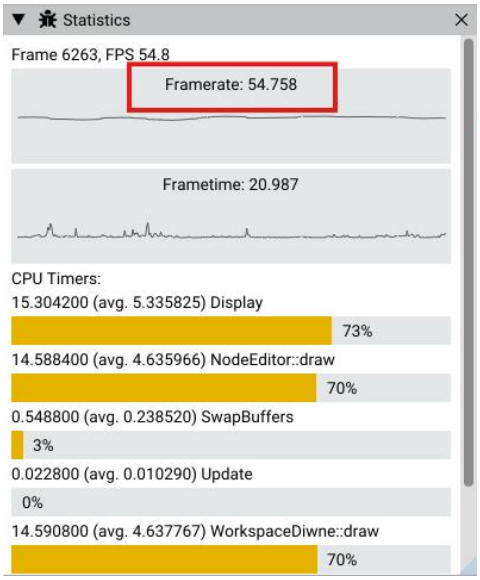
(a) : Uzly v Workspace po optimalizaci



(b) : Uzly v Workspace před optimalizací



(c) : FPS po optimalizaci



(d) : FPS před optimalizaci

Obrázek 7.14: Zoom scény před a po optimalizaci

7.4.2 Optimalizace hlavní smyčky aplikace.

Dalším krokem optimalizace aplikace I3T byla optimalizace hlavní smyčky aplikace. Aplikace využívá technologii VSync, která synchronizuje obnovovací frekvenci aplikace s frekvencí monitoru. Toto řešení je užitečné v tom, že eliminuje trhání obrazu a šetří prostředky počítače tím, že brání aplikaci běžet na maximální výkon (v případě, že maximální počet FPS je vyšší než obnovovací frekvence monitoru). Použití funkce VSync však zvyšuje latenci aplikace, čímž se snižuje odezva rozhraní. Další nedostatek hlavní smyčky byl v tom, že aplikace běžela vždy se stejnou obnovovací frekvencí bez ohledu na to, zda s ní uživatel interagoval.

Proto bylo k vyřešení těchto problémů rozhodnuto vyvinout nástroj odpovědný za správu hlavní aplikační smyčky. Byla vytvořena nová třída AppLoopManager, která se stala jedním z objektů hlavní třídy Application. AppLoopManager uchovává informace o tom, zda se má použít Vsync, po-

žadovaný limit snímkové frekvence, zda má být omezen, a požadovaný limit snímkové frekvence pro období, kdy uživatel s aplikací neinteraguje, zda má být omezen. Třída se zabývá sledováním toho, kolik času uplynulo od posledního snímku, a umožňuje uspat vlákno na požadovanou dobu pomocí funkce sleepUntilNextFrame(). AppLoopManager je navíc zodpovědný za výpočet doby, na kterou by mělo být vlákno uspáno, pokud uživatel přestal s aplikací komunikovat, pomocí funkce getFrameDurationOnIdle().

7.4.3 Manuální omezení FPS.

Třída Application obsahuje funkci run(), tj. funkci, která přímo obsahuje hlavní smyčku aplikace. Funkce frame() připraví a vykreslí celý snímek. Po vykreslení snímku byl přidán blok kódu, který na potřebnou dobu uspí vlákno, pokud je nutné jej uspat. Funkce shouldLimitFPS() vrací true, pokud je vypnuta funkce VSync a je povoleno manuální omezení FPS. (viz Kód 7.4)

```
1  int Application::run()
2  {
3      int exitCode = 0;
4      ...
5      while (!m_shouldClose)
6      {
7          if (!frame())
8              break;
9
10         if (m_appLoopManager.shouldLimitFPS())
11             m_appLoopManager.sleepUntilNextFrame();
12     }
13     return exitCode;
14 }
```

Kód 7.4: Zjednodušená funkce Application::run() po optimalizaci

```
1  int Application::run()
2  {
3      int exitCode = 0;
4      ...
5      while (!m_shouldClose)
6      {
7          if (!frame())
8              break;
9      }
10     return exitCode;
11 }
```

Kód 7.5: Zjednodušená funkce Application::run() bez optimalizace

Pro udržení stabilního času snímku ve funkci sleepUntilNextFrame() se provede funkce std::this_thread::sleep_until(m_nextFrameTime), která vlákno uspí do zadaného času, a poté se k m_nextFrameTime přičte doba trvání snímku vypočtená z cílového FPS. Tato metoda umožňuje zachovat plynulý časový plán snímků, i když se doba provedení některého ze snímků prodlouží. (viz Kód 7.6)

```
void AppLoopManager::sleepUntilNextFrame()
{
    std::this_thread::sleep_until(m_nextFrameTime);

    m_nextFrameTime += m_frameDuration;

    // Catch up if we are behind schedule
    while (clock::now() > m_nextFrameTime)
    {
       m_nextFrameTime += m_frameDuration;
    }
}
```

Kód 7.6: Funkce sleepUntilNextFrame()

Při manuálním omezení obnovovací frekvence aplikace pomocí funkce std::this_thread::sleep_until() se na platformě Windows vyskytoval problém s rozlišením časovače (timer resolution), který neumožňoval uspat vlákno na dobu kratší, než je minimální rozlišení časovače. Ve výchozím nastavení je rozlišení časovače systému Windows nastaveno na 16,5 ms, takže pro odstranění tohoto omezení při inicializaci AppLoopManager nastaví minimální rozlišení časovače na 1 ms pomocí funkce timeBeginPeriod(1) [9], což umožní uspání vlákna na 1 ms, což umožní dosáhnout maximální možné obnovovací frekvence 1000 snímků za sekundu. Destruktor třídy vrátí rozlišení časovače na výchozí rozlišení pomocí funkce timeEndPeriod(1).

7.4.4 Omezení FPS, když uživatel neinteraguje s aplikací.

Hlavní myšlenkou této optimalizace bylo snížit obnovovací frekvenci aplikace v době, kdy uživatel s aplikací neinteraguje (Idle), a ušetřit tak prostředky počítače.

Jeden cyklus aplikace je reprezentován funkcí frame(), která přijímá všechny události zadané uživatelem, zpracovává je a vykresluje snímek aplikace. Na samém začátku je volána funkce startFrame(), která aktualizuje čas uplynulý od posledního snímku (delta time). Původně funkce frame() používala funkci glfwPollEvents(), která aktualizovala údaje o událostech bez ohledu na to, zda se změnily, nebo ne. Tato funkce byla nahrazena funkcí glfwWaitEventsTimeout(sec), která čeká, až přijde událost od uživatele, a pokud žádná událost nepřijde, provede se po uplynutí doby uvedené v argumentech funkce (viz Kód 7.7). Argumenty této funkce obsahují výsledek funkce getFrameDurationOnIdle(), která vrací čas v sekundách, který je třeba čekat, aby bylo dosaženo cílového času snímku. Funkce getFrameDurationOnIdle() implementuje možnost čekat po ukončení interakce s uživatelem určitou dobu, přičemž vrací hodnotu rovnou 0 sekundám. To znamená, že funkce glfwWaitEventsTimeout() nebude čekat a provede se okamžitě. To má zabránit okamžitému omezení obnovovací frekvence po skončení interakce s uživatelem, což by vedlo k trhané animaci.

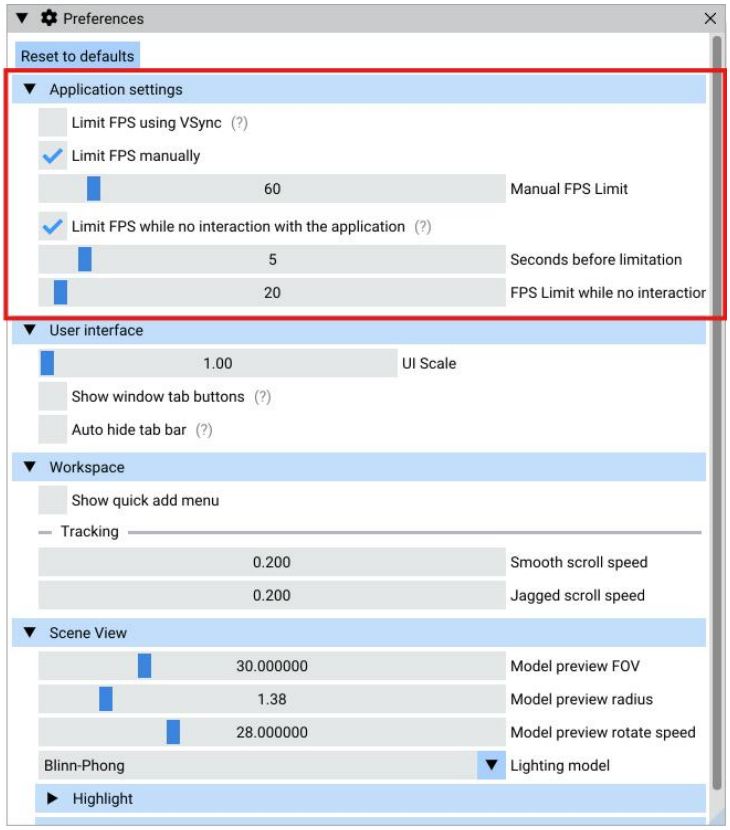
```
bool Application::frame()
{
```

```
3       m_appLoopManager.startFrame();
4
5       glfwWaitEventsTimeout(m_appLoopManager.getFrameDurationOnIdle());
6
7       ...
8       update();
9       display();
10      ...
11      return true;
12  }
```

Kód 7.7: Zjednodušená funkce Application::frame()

7.4.5 Uživatelské rozhraní pro ovládání obnovovací frekvence.

Aby bylo možné plně přizpůsobit frekvenci aktualizací aplikace, bylo nutné připravit a implementovat uživatelské rozhraní. To by umožnilo přizpůsobit aplikaci vlastním potřebám nebo v případě problémů vypnout funkčnost. Nastavení FPS bylo přidáno do stávajícího okna nastavení aplikace, protože již obsahuje základní nastavení aplikace, jako je například nastavení měřítka rozhraní (viz Obrázek 7.15).



Obrázek 7.15: Ovládání FPS v okně „Nastavení“ v I3T

Položky nabídky umožňují:

- Kliknutím na položku „Limit FPS using VSync“ je možné povolit nebo zakázat použití technologie VSync.
- Povolit a ručně konfigurovat omezení obnovovací frekvence aplikace kliknutím na položku nabídky „Limit FPS manually“.
- Povolit a nakonfigurovat omezení obnovovací frekvence aplikace, když nedochází k interakci s uživatelem, kliknutím na položku nabídky „Limit FPS while no interaction with the application“, včetně nastavení cílového FPS i počtu sekund před zahájením omezení.

Všechna změněná nastavení se mezi relacemi aplikace ukládají. Kliknutím na tlačítko „Reset to defaults“ se obnoví výchozí nastavení. Výchozí nastavení je možné změnit v souboru ApplicationSettings.h.

Výchozí nastavení hlavní smyčky aplikace:

- Omezení pomocí VSync - zapnuto.
- Manuální omezení FPS - vypnuto.
- Cílové FPS pro manuální omezení je 60 FPS.
- Omezení FPS, když uživatel neprovádí interakci s aplikací - zapnuto.
- Doba před zahájením omezení je 5 sekund.
- Cílové FPS, když uživatel s aplikací neinteraguje - 20 FPS.

7.5 Výsledky profilování a optimalizace aplikace.

1. Výsledkem profilování byla úspěšná identifikace míst, která v aplikaci zabírají nejvíce času.
2. Další optimalizace identifikovaných míst vedla k 12% nárůstu výkonu aplikace pro scény s velkým počtem uzlů.
3. Hlavní smyčka aplikace byla úspěšně optimalizována s možností kontroly práce v již zkompilované aplikaci. Bylo implementováno manuální omezení obnovovací frekvence a automatické omezení obnovovací frekvence aplikace, pokud s ní uživatel neprovádí žádnou interakci. To zlepšuje odezvu aplikace a šetří prostředky počítače v době nečinnosti.

Kapitola 8

Inventarizace a úprava mechanismu historie akcí (Ctrl-Z / Ctrl-Y).

I3T se ve výuce používá k demonstraci trojrozměrných transformací a studenti často během několika minut provedou desítky pokusů a způsobí mnoho chyb. Chybějící spolehlivá historie činí vrácení změny zpět problematickým a snižuje význam nástroje jako experimentální platformy.

Kromě toho je správně implementovaná historie základem pro případné pozdější přehrávání akcí krok za krokem při rozboru domácích úkolů.

8.1 Stav před změnou kódu.

V aplikaci I3T je za serializaci scén zodpovědný modul StateManager. Každý funkční modul dědí rozhraní IStateful a implementuje metody `saveScene()` a `loadScene()`, čímž vytváří vlastní snímek (dále Memento) to jest objekt rapidjson::Document obsahující data, která potřebuje každý modul uložit, např. informace o umístění oken pro modul UIModule. StateManager agreguje objekty memento do jediného JSON a jako všechno je zapisuje do souboru.

Od začátku se předpokládalo, že stejný princip bude použit i pro mechanismus Undo/Redo, kde při každé interakci uživatele se vytvoří nový snímek (Memento) spuštěním funkce saveScene() pro každý modul, ale místo ukládání do souboru bude vložen do vhodné datové struktury (např. fronty) a příkazy pro Undo/Redo budou měnit pouze aktuální index v datové struktuře.

Byly zjištěny tyto nedostatky původní implementace:

- **Porušení principu větvení historie.**

  Při provedení funkce undo() se snížil aktuální index, ale při další uživatelské akci takeSnapshot() se bezpodmínečně přidal nový objekt Memento na konec fronty, aniž by se zrušená větev smazala. V důsledku toho bylo možné se vrátit do neexistujícího stavu.

- **Neomezený nárůst paměti.**

  Neexistoval mechanismus pro zkracování historie na nastavitelnou velikost.

Implementace byla označena za nefunkční a musela být kompletně přepracována.

8.2 Úprava logiky ukládání historie.

Pro implementaci správné logiky ukládání historie bylo rozhodnuto implementovat vlastní datovou strukturu vhodnou pro současné potřeby, protože std (standardní knihovna C++) neobsahuje datovou strukturu vhodnou pro splnění požadavků (zejména kruhovou vyrovnávací paměť).

8.2.1 Datové struktura CircularVector.

Požadavky na datovou strukturu pro uchovávání historie:

1. Uložení indexu aktuálního prvku, aby bylo možné se v prvcích posouvat dopředu a dozadu.
2. Rychlý přístup k aktuálnímu prvku a jeho sousedům.
3. Možnost rychle odstranit položky, které jdou po aktuálním, aby bylo možné provést čištění historie.
4. Omezená kapacita s cyklickým přepisováním nejstarších dat, čímž se udržuje konstantní množství paměti (kruhová vyrovnávací paměť).

Pro splnění těchto požadavků byla vyvinuta vlastní datová struktura `CircularVector<T>` - kruhová vyrovnávací paměť s pevnou kapacitou (viz Kód 8.1). Kromě pozice prvního prvku (head) je v ní uložena i pozice aktuálního prvku (current). Funkce pushBack(value) vkládá prvek na samý konec vektoru (konec vektoru se počítá podle jeho skutečné velikosti, nikoli podle pozice aktuálního prvku). Funkce insertAfterCurrent(value) vkládá prvek hned za aktuální prvek a posouvá všechny následující prvky. Funkce popCurrent() vrací aktuální prvek a posouvá pozici aktuálního prvku na předchozí prvek (pokud existuje). Funkce revertCurrent() posouvá pozici aktuálního prvku na následující prvek (v případě, že existuje) a vrací tento prvek. Funkce truncateFromCurrent() odstraní všechny prvky následující za aktuálním prvkem zmenšením aktuální velikosti vektoru.

```
1  template<typename T>
2  class CircularVector {
3
4      std::vector<T> buffer; // interní úložiště
5      size_t fullCapacity; // maximální počet prvků
6      size_t actualSize; // aktuální počet položek
7      size_t head; // index nejstaršího prvku
8      int current; // index aktuální prvku
9
10 public:
11
12     explicit CircularVector(size_t cap);
```

```
13      size_t length() const;
14      bool empty() const;
15      void pushBack(T&& value);
16      void insertAfterCurrent(T&& value);
17      T& popCurrent();
18      T& revertCurrent();
19      void truncateFromCurrent();
20      ...
21  };
```

Kód 8.1: Zjednodušená třída CircularVector

8.2.2 Operace Undo/Redo s ohledem na CircularVector.

Implementovaný objekt datové struktury CircularVector byl přidán do třídy StateManager pro ukládání snímků aplikace během uživatelské interakce. Dále je popsán způsob, jakým třída pracuje s CircularVector při volání funkce pro vytvoření snímku a provádění operací Undo a Redo.

Při **vytváření nového snímku scény** pomocí funkce StateManager:: takeSnapshot() se provede následující logika:

1. Vytvoří se nové snímek scény ve formátu Memento (objekt rapidjson::Document).
2. Pokud aktuální index neukazuje na poslední prvek, je zavolána funkce truncateFromCurrent() - všechny stavy následující za aktuálním prvkem jsou odstraněny. Ve skutečnosti tato funkce trvá konstantní dobu provádění, protože nemaže prvky, ale snižuje hodnotu aktuální délky vyrovnávací paměti.
3. Funkce pushBack() přidá na konec vyrovnávací paměti Memento, na které se přesune index aktuálního prvku.

Pro navigaci na pracovní ploše (posun a zvětšení) do modulu StateManager byla přidána metoda takeRewritableSnapshot(). Pokud byl předchozí snímek vyvolán stejnou metodou a nedošlo k žádné nové akci uzlu, snímek přepíše aktuální stav bez zvýšení hloubky historie. Tímto způsobem se při posouvání a změně velikosti pracovní plochy uloží do historie pouze její finální pozice a velikost.

Při provádění operace **Undo** pomocí funkce StateManager::undo() se provede následující logika:

1. Získá aktuální memento pomocí funkce popCurrent(), pokud vyrovnávací paměť CircularVector není prázdná.

   Ve funkci popCurrent() se index aktuálního prvku přesune na předchozí.
2. Pomocí funkce loadScene(memento) načte se předchozí scéna.

Při provádění operace **Redo** pomocí funkce StateManager::redo() se provede následující logika:

1. Získá další prvek za aktuálním indexem pomocí funkce revertCurrent(), pokud již není posledním prvkem.

   Ve funkci revertCurrent() se index aktuálního prvku přesune na následující prvek.

2. Pomocí funkce loadScene(memento) načte se následující scéna.

Když je vyrovnávací paměť plná (length()==kapacita), metoda pushBack() automaticky přesune index úplně prvního prvku na následující prvek, čímž se okno dostupné historie přemístí bez zvětšení paměti.

8.2.3 Shrnutí změn kódu.

Nová implementace umožnila deterministickou a předvídatelnou navigaci v historii akcí a snížila spotřebu paměti díky pevné velikosti vyrovnávací paměti.

- Opraveny logické chyby: Undo/Redo správně ničí nepotřebné snímky po větvení.
- Implementována datová struktura (kruhová vyrovnávací paměť) s omezenou kapacitou, aby se zabránilo neomezenému nárůstu paměti.
- Do historie se ukládá až výsledná pozice a velikost pracovního prostoru.
- Uživatel získá předvídatelné a stabilní chování kombinací Ctrl-Z / Ctrl-Y ve všech běžných scénářích I3T.

8.3 Místa pro vytvoření snímku.

Úplné prozkoumání kódu aplikace odhalilo všechny akce uživatele, které mění stav scény. V každém z těchto bodů je volána funkce takeSnapshot() (nebo takeRewritableSnapshot() pro navigaci):

- Změna číselné hodnoty ve všech uzlech obsahujících editovatelné pole s touto hodnotou.
- Výběr jednoho nebo více uzlů (selekce).
- Ukončení pohybu uzlu po pracovním prostoru (přesun uzlu).
- Odstranění uzlu (delete).
- Přidání nového uzlu (insert).
- Připojování a odpojování drátů na piny uzlů (link).
- Poslední posouvání pracovního prostoru (panning).
- Poslední přiblížení pracovního prostoru (zoom).

8.4 Přidání scény k existující scéně (funkce *Append scene*).

Další možností, která byla implementována v rámci vylepšení funkcionality I3T, bylo doplnění možnosti přidat existující scénu do aktuálně otevřené scény. Tím by se zjednodušilo vytváření nových scén, možnost rozdělit scény na funkční bloky s možností jejich okamžitého importu do scény.

Nezbytné požadavky:

- Přidání podscény do skutečné otevřené scény.
- Uložení obsahu aktuální scény.
- Žádné volání k otevření modálního okna „Uložit aktuální scénu“.
- Zrušení výběru všech aktuálně vybraných uzlů a výběr všech přidaných uzlů.
- Přidání všech importovaných modelů z přidané scény do aktuální scény.

8.4.1 Detaily implementace.

Pro implementaci této funkce bylo možné vyjít z již existující funkce loadScene(), která je volána pro všechny objekty dědící ze třídy IStateful a přidána do StateManageru jako originátor.

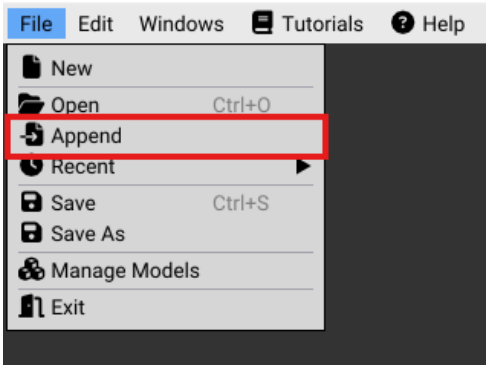
Pro splnění potřebných požadavků bylo nutné pochopit implementaci funkce loadScene() pro WorkspaceModule a ResourceManager.

- WorkspaceModule při volání funkce loadScene() vyčistí scénu a přidá všechny uzly uložené v souboru scény.
- ResourceManager vyčistí dočasný adresář pro modely a přidá všechny importované vlastní modely, které byly do scény načteny uživatelem.

Po prostudování implementace metody loadScene() bylo rozhodnuto přidat do IStateful novou virtuální metodu appendScene().

- Modul WorkspaceModule implementující tuto metodu neprovádí čištění scény, ale přidává uzly ze ctěné scény přímo do stávající scény, a označuje je jako vybrané.
- ResourceManager neprovádí čištění dočasného adresáře a přidá všechny importované modely do skutečné scény.

Na základě provedených změn byla implementována funkce přidání libovolné existující scény k aktuální scéně (viz Obrázek 8.1).



Obrázek 8.1: Umístění funkce "Append Scene"v I3T.

Kapitola 9

Inventarizace a úprava mechanismu logování uživatelské interakce.

Dalším krokem práce na aplikaci I3T (An Interactive Tool for Teaching Transformations) byla inventarizace a případná implementace mechanismu pro logování interakce uživatelů.

Logování interakcí uživatelů je důležitým a nezbytným nástrojem v každé aplikaci. Umožňuje vývojářům sledovat, jak uživatel s aplikací interaguje, a je užitečným nástrojem pro analýzu všech implementovaných funkcí aplikace.

9.1 Analýza existujícího stavu mechanismu logování.

Starší verze I3T, rovněž vyvinutá na katedře počítačové grafiky a interakce, již měla možnost sledovat a zaznamenávat interakci uživatele s aplikací do souborů. Tato funkce byla přidána v rámci bakalářského projektu [Logovací systém pro nástroj na výuku transformací I3T, Filip Uhlík, BP 2020] [8].

Po prozkoumání kódu aktuální verze I3T bylo zjištěno, že hlavní kód a makra používaná k implementaci nástroje pro protokolování byla přenesena ze staré verze. Funkce přeneseného kódu však nebyly využity a nebyly kompatibilní se současnou implementací I3T.

Co bylo přeneseno:

- Singleton třídy Logger, která provádí základní logiku logování a používá knihovnu `spdlog`.
- Makra pro logování určený pro vývojáře.

  Umožňuje vypisovat do konzoly zprávy s různou úrovní důležitosti.

```
1 #define LOG_TRACE(...) ...
2 #define LOG_DEBUG(...) ...
3 #define LOG_INFO(...) ...
4 #define LOG_WARN(...) ...
5 #define LOG_ERROR(...) ...
6 #define LOG_FATAL(...) ...
```

- Makra pro logování uživatelské interakce.

```
#define LOG_EVENT_MOUSE_POS(mouseX, mouseY) ...
#define LOG_EVENT_MOUSE_CLICK(button, mouseX, mouseY) ...
#define LOG_EVENT_MOUSE_RELEASE(button, mouseX, mouseY) ...
#define LOG_EVENT_OPEN_POP_UP(clickedTab) ...
#define LOG_EVENT_CLOSE_POP_UP(tab) ...
#define LOG_EVENT_CLOSE_POP_UP_IN(tab) ...
#define LOG_EVENT_BUTTON_CLICK(tab) ...
#define LOG_EVENT_CONNECT(outputPin, inputPin) ...
#define LOG_EVENT_DISCONNECT(outputPin, inputPin) ...
#define LOG_EVENT_TAB_ADDED_AT_INDEX(index, addedTab, addedToTab)
    ...
#define LOG_EVENT_TAB_ADDED(addedTab, addedToTab) ...
#define LOG_EVENT_TAB_REMOVED(removedTab, removedFromTab) ...
#define LOG_EVENT_OBJECT_ADDED(objectName, tab) ...
#define LOG_EVENT_OBJECT_REMOVED(tab) ...
#define LOG_EVENT_MATRIX_VALUE_UPDATE(tabGroup, matrix, index,
    newVal) ...
#define LOG_EVENT_MATRIX_VALUE_UPDATE_DRAG(matrix, index, newVal)
    ...
#define LOG_EVENT_TUTORIAL_STEP(tutorial, step, name) ...
```

Co bylo vhodné pro použití bez nutnosti změn:

- Třída Logger byla plně kompatibilní s aplikací.
  - Jediné, co bylo třeba opravit, byla logika zápisu zpráv do souborů. Protože v současné implementaci se při zapnutém logování interakce uživatele záznamy vypisovaly jak do konzole, tak do souboru `UserInteraction.log` a `Mouse.log`.
- Makra pro logování určená pro vývojáře byla plně kompatibilní a v aplikaci se již používala.

Co bylo třeba přepracovat pro pozdější použití v aplikaci:

- Makra pro logování uživatelských interakcí bylo nutné přepracovat, protože se změnila vnitřní struktura ukládání dat v aplikaci. Bylo nutné odstranit makro pro sledování zavírání menu v aplikaci, protože taková událost již neexistuje a přidat makra pro sledování kliknutí na tlačítko a výběru objektu z menu.

Další zkoumání kódu ukázalo, že v aplikaci byla použita pouze makra `LOG_EVENT_MOUSE_POS`, `LOG_EVENT_MOUSE_CLICK` a `LOG_EVENT_MOUSE_RELEASE`. Jsou zodpovědná za zaznamenávání polohy a kliknutí myší v aplikaci.

9.2 Oprava mechanismu logování uživatelské interakce.

Po prozkoumání současné implementace logování byly provedeny následující změny:

- Třída Logger byla upravena tak, aby se logování uživatelské interakce ukládalo pouze do souboru `UserInteraction.log`, pokud je zapnuto.
- Třída Logger byla upravena tak, aby se zaznamenávání polohy myši a kliknutí ukládalo pouze do souboru `Mouse.log`, pokud je zapnuto.
- Logování uživatelské interakce je nyní zapnuto pro režim sestavení aplikace `Debug` a vypnuto pro režim `Release` (je možné zapnout níže uvedenými tlačítky).

  Podle původní implementace logování byla veškerá logika rozdělena do čtyř částí:

  1. **Logování otevíraných nabídek v aplikaci (pop-up)**
     Zapnutí klávesou `F1`
  2. **Logování logiky v aplikaci (přidávání, odstraňování nových uzlů atd.)**
     Zapnutí klávesou `F2`
  3. **Logování změn hodnot ve všech uzlech obsahujících editovatelné pole s touto hodnotou.**
     Zapnutí klávesou `F3`
  4. **Logování pohybu myši**
     Zapnutí klávesou `F4`

- Makra pro logování uživatelské interakce byla přizpůsobena současné struktuře aplikace.

  V původní verzi byl některým makrům pro logování předáván celý objekt s názvem `tab`, což byl libovolný obdélník z GUI.

  Příklad: Makro `LOG_EVENT_CLOSE_POP_UP(tab)`. Z něj se volala funkce `tab->getNameableParentNameId()`, která získávala informace o názvu okna nebo objektu. Tato implementace není pro současnou verzi aplikace vhodná z důvodu nepřítomnosti této funkce pro objekty. Proto se nyní při každém takovém volání získává přímo požadovaný název objektu jako řetězec.

  Bylo odstraněno makro `LOG_EVENT_CLOSE_POP_UP`, které se v původní verzi aplikace volalo při zavření nabídky na pracovním prostoru (Workspace). Důvodem odstranění bylo to, že grafická knihovna `Dear ImGui` tuto událost negeneruje.

  Bylo přidáno nové makro `LOG_EVENT_MENU_CLICK(label)` pro logování uživatelské interakce s nabídkou v horní části oken a makro `LOG_EVENT_SELECT(label)` pro logování konečného výběru určitého uzlu z nabídky nebo okna.

  Vzhledem ke grafické knihovně `Dear ImGui` nebylo nutné rozlišovat, byla-li hodnota v matici uzlů změněna tažením myší nebo ručním zadáním z klávesnice. Proto makra `LOG_EVENT_MATRIX_VALUE_UPDATE` a `LOG_EVENT_MATRIX_VALUE_UPDATE_DRAG` byla sloučena do jednoho makra `LOG_EVENT_MATRIX_VALUE_UPDATE`.

Název makra	Ukládá se
LOG_EVENT_MOUSE_POS (mouseX, mouseY)	Aktuální poloha myši
LOG_EVENT_MOUSE_CLICK(button, mouseX, mouseY)	Poloha myši v okamžiku kliknutí
LOG_EVENT_MOUSE_RELEASE(button, mouseX, mouseY)	Poloha myši v okamžiku uvolnění

Tabulka 9.1: Makra pro logování s popisem (S výstupem do souboru `Mouse.log`)

Název makra	Ukládá se
LOG_EVENT_MOUSE_POS (mouseX, mouseY)	Aktuální poloha myši
LOG_EVENT_MOUSE_CLICK(button, mouseX, mouseY)	Poloha myši v okamžiku kliknutí
LOG_EVENT_MOUSE_RELEASE(button, mouseX, mouseY)	Poloha myši v okamžiku uvolnění
LOG_EVENT_OPEN_POP_UP(label)	Název otevřené kontextové nabídky v pracovním prostoru
LOG_EVENT_MENU_CLICK(label)	Název kliknuté nabídky v horní nabídce oken
LOG_EVENT_BUTTON_CLICK(label)	Název tlačítka, na které bylo kliknuto
LOG_EVENT_SELECT(label)	Název uzlu, který byl vybrán v nabídce
LOG_EVENT_CONNECT(outputPin, inputPin)	Název pinu, který byl připojen, a název pinu, ke kterému byl připojen
LOG_EVENT_DISCONNECT(outputPin, inputPin)	Název pinu, který byl odpojen, a název pinu, od kterého byl odpojen
LOG_EVENT_NODE_ADDED_AT_INDEX(index, addedNode, addedToNode)	Název uzlu, který byl přidán k uzlu sekvence, s názvem uzlu a pozicí, na kterou byl přidán
LOG_EVENT_NODE_ADDED(addedNode, addedToNode)	Název přidaného uzlu a název objektu, ke kterému byl přidán
LOG_EVENT_NODE_REMOVED(removedNode, removedFromNode)	Název odstraněného uzlu a název objektu, ze kterého byl odstraněn
LOG_EVENT_OBJECT_ADDED(objectName, node)	Název uzlu modelu přidaného do pracovního prostoru
LOG_EVENT_MATRIX_VALUE_UPDATE(matrix, index, newVal)	Název uzlu, index v matici a hodnota, která byla nastavena v matici uzlu
LOG_EVENT_TUTORIAL_STEP(tutorial, step, name)	Název tutoriálu a číslo tutoriálového kroku

Tabulka 9.2: Makra pro logování s popisem (S výstupem do souboru `UserInteraction.log`)

```
: >>> Interaction logger initialized! <<<
: P: Right clicked "WorkspaceEditor:1" to show the pop up menu
: P: Selected "Matrix Sequence"
: L: "sequence:12" added to "WorkspaceEditor:1"
: P: Right clicked "WorkspaceEditor:1" to show the pop up menu
: P: Selected "free"
: L: "free:19" added to "WorkspaceEditor:1"
: L: "free:19" added to "sequence:12" at index "0"
: L: Connecting "pin 1 of Sequence #1" output to "pin 1 of Sequence
  #9" input
: P: Opened the "Edit" menu
: P: Opened the "Selection" menu
: P: Selected "Select all"
: P: Opened the "Windows" menu
: P: Selected "Tutorial window"
: P: Clicked the "Open Start Menu" button
: P: Clicked the "Start##Data/Tutorials/1TUT\1TUT.tut" button
: T: (1. Začínáme s I3T).0: V této lekci se seznámíš s I3T.
: P: Clicked the "Next" button
: T: (1. Začínáme s I3T).1: V této lekci se seznámíš s I3T.
: >>> Interaction logger ending! <<<
```

Kód 9.1: Záznam uživatelské interakce v logu (soubor `UserInteraction.log`)

9.3 Doplnění aplikačního kódu pro podporu logování.

Vzhledem k tomu, že ze staré verze aplikace byla přenesena pouze třída Logger, byl prozkoumán celý kód aplikace a na potřebná místa byla přidána všechna příslušná makra.

Pro podporu logování otevření menu a stisků tlačítek byly zapouzdřeny příslušné funkce z knihovny `Dear ImGui`:

- Byl vytvořen nový jmenný prostor `I3TGui`. Obsahuje všechny potřebné funkce z `ImGui` pro podporu logování.
- Funkce `ImGui::MenuItem` byla zabalena do `I3TGui::MenuItemWithLog`, aby bylo možné zaznamenávat kliknutí na položku v menu.

  V případě potřeby je možné v argumentech funkce zadat požadovanou funkci logování.
- Funkce `ImGui::BeginMenu` byla zabalena do funkce `I3TGui::BeginMenuWithLog`, která podporuje logování kliknutí na menu.
- Funkce `ImGui::Button` byla zabalena do `I3TGui::ButtonWithLog`, aby podporovala logování kliknutí na tlačítko.

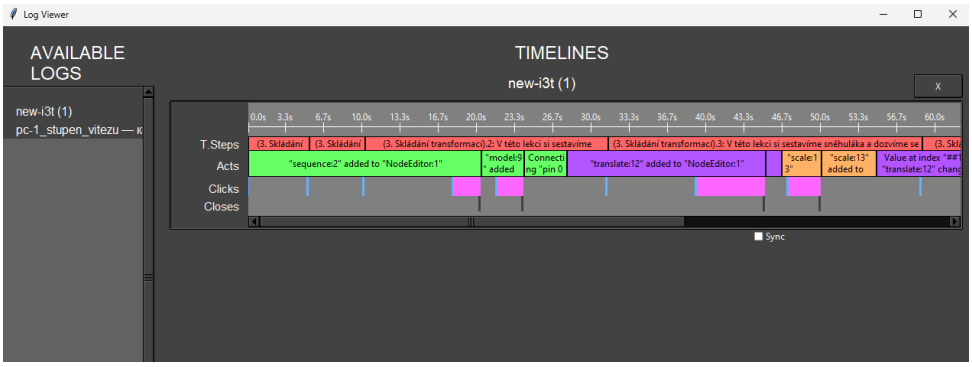
To umožnilo implementovat do nové verze `I3T` plnohodnotné logování interakce uživatele bez nutnosti výrazného zkomplikování kódu.

9.4 Integrace se stávající aplikací pro prohlížení logů.

V rámci bakalářské práce Filipa Uhlíka [8], ve které bylo implementováno logování starší verze aplikace I3T, byla vytvořena i aplikace LogViewer, která umožňuje grafické zobrazení logů vytvořených v I3T (viz Obrázek 9.1).

Kvůli přizpůsobení loggeru aktuální verzi I3T a konkrétně odstranění makra `LOG_EVENT_CLOSE_POP_UP` pro sledování zavírání nabídek, přidání makra `LOG_EVENT_BUTTON_CLICK`, které sleduje kliknutí na tlačítko a přidání makra `LOG_EVENT_MENU_CLICK` pro sledování kliknutí na menu, nebylo možné použít k prohlížení logů stávající aplikaci LogViewer.

Po prozkoumání zdrojového kódu aplikace a odstranění výše uvedených nekompatibilit byla obnovena možnost načítat a prohlížet soubory logů nové verze I3T (viz Obrázek 9.1). Nyní však aplikace nezaznamenává událost zavření nabídky v menu a nezobrazuje ji na grafu.



Obrázek 9.1: Zobrazení protokolů z nové verze I3T v aplikaci LogViewer

9.5 Možnosti logování ve skriptování.

V aplikaci I3T lze nově psát tutoriály s podporou skriptování [Skriptování v nástroji I3T a automatizace zveřejňování na GitHubu, Martin Herich, DP 2025] [7]. To rozšiřuje možnosti tutoriálů tím, že umožňuje vytvářet tutoriály s různými podmínkami, které je třeba splnit pro další postup, nebo vytvářet dotazníky a testy přímo v tutoriálu.

V této fázi vývoje logování nepodporuje možnost zaznamenávat, na co uživatel v tutoriálu kliknul.

Při potřebě rozšířit funkcionalitu pro podporu logování uvnitř tutoriálů se skripty bude postačovat:

- Vytvořit nová makra potřebná pro tutoriály.

  Například: `LOG_EVENT_CHECKBOX` pro zaznamenání, které volby provedl uživatel v testu tutoriálu. `LOG_EVENT_RESULT` pro zaznamenání, zda uživatel vybral správnou odpověď.

- Doplnit soubor `TutorialWindow.cpp` pro správné sledování událostí o potřebná makra.
- V případě přidání makra `LOG_EVENT_CHECKBOX` je nejlepší vytvořit obal `I3TGui::CheckboxWithLog()` kolem `Dear ImGui` funkce `ImGui::Checkbox()` ve jmenném prostoru `I3TGui` pro zjednodušenou podporu logování.

Aktuální implementace skriptovaných tutoriálů a implementace logování umožňuje v případě potřeby jednoduše přidat požadovanou funkcionalitu. Tím se přidá podrobnější sledování akcí uživatele, což je při práci s tutoriály důležité.

Kapitola 10

Implementace podpory vícejazyčných rozhraní v I3T.

V rámci projektu bylo nutné do aplikace An Interactive Tool for Teaching Transformations (I3T) přidat možnost přepínat jazyk uživatelského rozhraní mezi nejméně třemi jazyky.

10.1 Požadavky na provádění.

Řešení musí splňovat následující požadavky:

- **Nezávislost na platformě.**

  Aplikace se kompiluje pro Windows a Linux.

- **Minimální externí závislosti.**

  Není žádoucí připojovat těžké frameworky třetích stran.

- **Dynamická změna jazyka během provozu.**

- **Snadná rozšiřitelnost.**

  Práce na jazykových lokalizacích by neměla vyžadovat rekompilaci celého projektu.

10.2 Rešerše běžných lokalizačních metod.

Před implementací této funkcionality je nutné prozkoumat již existující metody implementace lokalizace v aplikacích.

Následující tabulka uvádí některé existující metody implementace lokalizace s výhodami a nevýhodami.

Metoda	Princip	Výhody	Nevýhody
Vložené řetězcové literály	Text je uložen přímo ve zdrojovém kódu	Jednoduchost, žádné externí soubory	Překompilování při jakýchkoli změnách; není rozdíl mezi vývojáři a překladateli
Soubory zdrojů (*.rc, *.resx, XML/JSON)	Řádky jsou přesunuty do externích tabulek načtených prostřednictvím rozhraní API	Výměna zdrojů za chodu; nezávislost na kódu	Vlastní systém parsování; nutnost spravovat několik souborů
GNU Gettext (libintl)	Řádky jsou označeny makrem _(...); msgid je extrahováno z kódu, překlad je uložen v *.po / *.mo; vyhledávání se provádí pomocí hashovací tabulky v runtime	Podpora plurálových forem, kontextu, kompozitů; vyspělý ekosystém; profesionální nástroje CAT.	Nutná knihovna třetí strany; binární katalogy komplikují ladění
Qt Linguist (tr())	Generátor souborů *.ts + převod do *.qm	Integrace s Qt; vizuální editor	Těsné propojení s Qt; hlavní závislost
ICU MessageFormat	BCP 47 Parametrizované zprávy	Síla formátování a pluralizace	Složitost API, velká velikost knihovny

Tabulka 10.1: Metody implementace lokalizace

V ekosystému otevřeného zdrojového kódu C++ se de facto standardem stal gettext z knihovny libintl. Z tohoto důvodu byl pro analýzu vybrán **Prusa Slicer** [11], který má otevřený zdrojový kód a používá knihovnu libintl.

10.3 Analýza řešení na bázi gettextu (příklad Prusa Slicer).

Nástroj gettext vychází z myšlenky jediného identifikátoru zprávy (msgid), který se rovná anglickému originálu. V procesu:

1. Nástroj xgettext prohledá zdrojový kód a vygeneruje soubor messages.pot.
2. Překladatelé naplní adresáře xx.po, kde každá položka odpovídá dvojici klíč-hodnota.

```
msgid "Open"
msgstr "Otevřít"
```

3. Zkompilované binární soubory *.mo jsou za běhu připojeny pomocí volání setlocale, bindtextdomain, gettext.

Výhody:

- Podpora plurality.
- Kontextový mechanismus msgctxt.
- Pohodlné grafické editory (Poedit, Weblate).

Nevýhody pro I3T:

- Nutnost připojovat libintl ke všem cílovým souborům.
- Komplikuje CMake (Je nutné stanovit vytvoření souborů .po a .mo) a multiplatformní sestavení

10.4 Důvody odmítnutí externí závislosti (gettext).

Po prototypování na libintl bylo zjištěno, že:

- Připojení gettextu přidá každému spustitelnému souboru přibližně 350 KB.
- Je třeba podporovat generování *.mo pro každou cílovou platformu.
- Pod Windows neexistuje systémový libintl, je nutné statické linkování nebo dodání DLL.

Vzhledem k tomu, že I3T potřebuje pouze jednoduchý substituční mechanismus bez množného čísla slov, bylo rozhodnuto implementovat lehký interní mechanismus inspirovaný gettextem, ale nezávislý na libintl.

10.5 Návrh vlastního lokalizačního mechanismu.

Pro vytvoření vlastního lokalizačního mechanismu založeného na principech gettextu je třeba definovat následující:

- **Formát souboru překladu.**

  V jakém formátu a s jakou strukturou budou lokalizační soubory uloženy.
- **Architektonický princip**, který bude použit pro implementaci lokalizace.

10.5.1 Formát lokalizačních souborů

- Každý jazyk je uložen v souboru `Data/Lokalization/<název>.txt` (např. `cz.txt`).
- Syntaxe je dvojice **klíč=hodnota**:

```
1        Open File=Otevřít soubor
```

10.5.2 Třída Localization (Singleton)

Vzhledem k tomu, že přístup k lokalizaci může být vyžadován v kterékoliv části aplikace, bylo rozhodnuto vytvořit třídu Localization pomocí vzoru singleton (inicializace se provede jednou při spuštění), aby bylo možné přistupovat k jednomu objektu třídy odkudkoli v aplikaci prostřednictvím funkce `Localization::instance()`.

Vytvořená třída Localization se skládá z následujících komponent (viz Kód 10.1):

- Pole `m_languages` obsahující informace o všech jazycích používaných v aplikaci.
- Pole `m_translations`, které je mapou ukládající prvky ve tvaru klíč=hodnota. V poli `m_translations` jsou uloženy všechny klíče a překlady pro aktuálně vybraný jazyk aplikace.
- Funkce `std::string& translate(const std::string& key)`, která vyhledá v datové struktuře unordered_map `"m_translations"` zadaný klíč a vrátí překlad.
- Funkce `loadLanguage(const std::string& langName)`, která načte překlady ze zadaného názvu jazyka do `m_translations`.

```
1  class Localization
2  {
3  public:
4      static Localization& instance();
5      const std::string& translate(const std::string& key) const;
6
7      bool loadLanguage(const std::string& langName);
8
9      ...
10
11 private:
12     std::unordered_map<std::string, std::string> m_translations;
13
14     std::vector<AppLanguage> m_languages{
15         {.id = 0, .name = "English", .filePath = "Data/Localization/
               en.txt"},
16         {.id = 1, .name = "Čeština", .filePath = "Data/Localization/
               cz.txt"},
17         ...
18     };
19
20     ...
21 };
```

Kód 10.1: Zjednodušená třída Localization

10.5.3 Makra pro usnadnění vyvolání.

Bylo rozhodnuto zabalit tato volání do makra pro snazší použití:

```
#define _ts(key) Localization::instance().translate(key)
#define _t(key) _ts(key).c_str()
#define ICON_T(icon,key) (std::string(icon) + _ts(key)).c_str()
```

- _ts - vrací std::string;
- _t - vhodné pro API očekávající const char* (např. Dear ImGui);
- ICON_T - spojí ikonu s překladem (používá se, pokud je ikona uložena ve fontu FontAwesome).

To umožňuje stručně začlenit lokalizaci do rozhraní Dear ImGui, jehož API očekává const char* (viz Kód 10.2).

```
SelectLayoutDialog::SelectLayoutDialog()
: IWindow(ICON_T(ICON_I3T_GRID " ", "Select Layout")) {}

void SelectLayoutDialog::showSelectLayoutMenu()
{
        ImGui::TextWrapped(_t("Manage window layouts."));

        ImGui::TextWrapped(
        (_ts("You can switch between existing layouts") + "\n"
          + _ts("or create new ones.")).c_str());
    ...
}
```

Kód 10.2: Volání maker pro překlad

10.6 Hodnocení vlivu na výkon.

Při implementaci lokalizace bylo důležité zajistit, aby zvolené řešení nemělo znatelný dopad na výkon.

Obavy vzbuzovalo vyhledávání v datové struktuře „`unordered_map`“ - kontejneru, který uchovává dvojice klíč-hodnota, na každém snímku a v každém výskytu funkce.

K ověření hypotézy o nákladnosti vyhledávání `unordered_map` byl proveden mikrobenchmark na počítači Ryzen 5 2600, 16 GiB RAM, GTX 1660 TI (sestavení Debug, Windows 11, VSync vypnuto):

- Počet unikátních klíčů - 5000.
- Doba trvání snímků základní aplikace - 2,6 ms ($\approx$ 420 FPS).

Počet volaní funkce translate() na snímek	Čas snímku (ms)	Čas funkce (ms)	% snímku	Snížení FPS
100	2.632	0.138	5	−40 (do 380 FPS)
1000	4.000	1.378	35	−170 (do 250 FPS)
5000	9.524	6.283	68	−315 (do 105 FPS)
10000	16.393	13.589	82	−359 (do 61 FPS)

Tabulka 10.2: Vliv volání translate() na výkon

Závěr:

Ačkoli výsledky testování ukázaly silný pokles výkonu aplikace, bylo vzato v úvahu, že ve výchozím nastavení je v aplikaci omezení obnovovací frekvence (VSync) a reálné využití překladové funkce nepřekročí 100-200 volání v jednom snímku (pokud budou otevřena všechna okna). Počet volání funkce translate() nezávisí na počtu uzlů v pracovním prostoru, protože v nich není volána.

Vzhledem k tomu, že zvláštností DearImGui je, že pokud se okno na obrazovce nezobrazí, kód, který ho tvoří, se nevolá, nebude se volat ani překladová funkce. S ohledem na tuto skutečnost nebude počet volání překladové funkce při běžném používání aplikace vyšší než 50.

Po zohlednění výše uvedených skutečností a po schválení vedoucím projektu bylo rozhodnuto tento přístup k implementaci lokalizace zachovat, protože splňoval požadavky na výkon.

10.6.1 Alternativní přístupy bez vyhledávání na mapě.

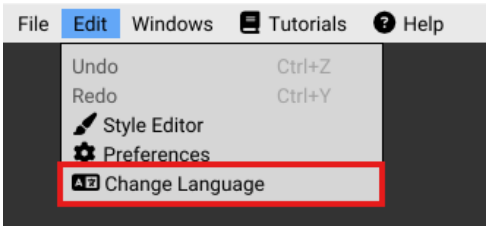
Přestože byl zvolený způsob implementace lokalizace přijat jako vhodný, bylo rozhodnuto prozkoumat alternativní způsoby ukládání a vyhledávání.

- Vygenerovaný „switch-statement“ (klíč -> enum).
  + Transparentní pro překladač.
  - Vyžaduje rekompilaci při přidávání řetězců.
- Perfect Hash (gperf).
  + O(1) bezkolizní.
  - Externí generátor.
  - Komplikovaný CMake.
- Konstantní index (std::array tabulka, přímý přístup podle předpočítaného ID).
  + Nulové náklady na volání.
  - Nedostatečná flexibilita.
  - Nepříjemnosti pro lokalizátory při přidávání a práci s jazyky.

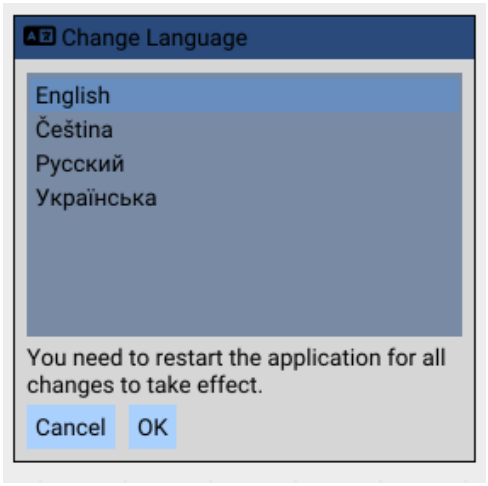
Tyto možnosti byly zamítnuty kvůli nutnosti překompilovat aplikaci při přidávání nebo úpravě překladů a kvůli nepříjemnostem pro lokalizátory.

10.7 Uživatelské rozhraní pro výběr jazyka.

Pro plné využití potenciálu lokalizačního nástroje bylo nutné vytvořit v aplikaci nové okno, které by uživatelům umožnilo změnit jazyk aplikace (viz Obrázek 10.2 a Obrázek 10.1).



Obrázek 10.1: Umístění okna "Change Language"



Obrázek 10.2: Nové modální okno "Změnit jazyk"

- Modální okno obsahuje seznam (ListBox) dostupných jazyků a tlačítka „OK“/„Cancel“.
- Po potvrzení se provede funkce Localization::loadLanguage(code).

  Načte se nový lokalizační soubor a aktualizují se všechny texty v aplikaci kromě názvů oken. Téměř všechny texty v aplikaci jsou lokalizovány ihned po potvrzení, s výjimkou názvů oken v aplikaci. To je způsobeno tím, že jejich název je nastaven jednou při spuštění aplikace. Řešením tohoto problému by bylo znovu vytvořit všechna okna, jak je to implementováno v aplikaci PrusaSlicer, nebo změnit přístup k vytváření oken.

  Bylo rozhodnuto tuto funkci neimplementovat, protože ke změnám v

jazyce aplikace nedochází často během standardního používání aplikace a její opětovné spuštění nepředstavuje velký problém.

- Index zvoleného jazyka se ukládá do souboru UserData/Global.json a načte se při příštím spuštění (viz Kód 10.3).

```
{
    "themeName": "ProjectorLightMode",
    "recentFiles": [
        ...
    ],
    "recentCommands": [
        ...
    ],
    "language": 1 // index 1 odpovídá jazyku čeština
}
```

Kód 10.3: Uložení indexu jazyka do souboru UserData/Global.json

10.8 Podpora čtyř jazyků a migrace do tabulky CSV.

Byly vytvořeny 4 lokalizační soubory (en.txt, cz.txt, ru.txt a ua.txt). Bylo přidáno plné pokrytí klíčů pro češtinu, ruštinu a ukrajinštinu.

- Pro angličtinu je překladem samotný klíč, což vývojářům usnadňuje práci neboť přidáním makra _t("Open File") okamžitě získají text "Open File".

  Tato implementace má své výhody i nevýhody [10]. Výhodou je, že pro přidání nového anglického textu jej stačí přidat do kódu jako klíč, ale přináší to takové nevýhody, jako je nemožnost snadno měnit text, aniž by se musel měnit klíč pro všechny jazyky, a nemožnost přidat dva překlady pro jeden klíč, pokud to například v jiných jazycích bude potřeba v závislosti na kontextu.

  Bylo rozhodnuto tuto implementaci zachovat, protože je v současné době pohodlnější pro vývojáře, kteří nelokalizují, a v případě potřeby lze snadno přejít na jinou implementaci, a to pouhým přidáním překladů pro angličtinu.

10.8.1 Tabulka localization.csv

S rostoucím počtem řádků se udržování čtyř stejných souborů stalo nepohodlným. Bylo rozhodnuto ukládat data do jediné tabulky localization.csv 10.3, kde první sloupec jsou klíče, ostatní sloupce jsou překlady (viz Obrázek 10.3). Vkládání jazyků je díky jedné společné tabulce jednodušší ze dvou důvodů. Prvním je, že překladatel vidí i překlady do jiných jazyků. Má větší kontext a snáze pak zvolí odpovídající výraz. Druhým důvodem je snazší práce s jedním

souborem než s několika, a jednodušší kontrola, že jsou ve všech jazycích přeloženy všechny pojmy.

	Key	Group na…	en	cz	ru
1		Main Menu Bar			
2		File Menu			
3	File			Soubor	Файл
4	New			Nová scéna	Новая Сцена
5	Append			Přidat scénu	Добавить Сцену
6	Open			Otevřít	Открыть
7	Recent			Nedávné	Недавние
8	Save			Uložit	Сохранить
9	Save As			Uložit jako	Сохранить Как
10	Manage Models			Správa modelů	Управление Моделями
11	Exit			Ukončit	Выход
12		Edit Menu			
13	Edit			Úpravy	Правка
14	Undo			Zpět	Отменить
15	Redo			Vpřed	Вернуть
16	Style Editor			Editor vzhledu	Редактор стиля
17	Preferences			Nastavení	Настройки
18	Change Language			Změnit jazyk	Изменить Язык
19		Windows Menu			
20	Windows			Okna	Окна
21	Start window			Úvodní okno	Начальное окно
22	Tutorial window			Okno tutoriálu	Учебное окно
23	Scene view window			Okno náhledu scény	Окно просмотра сцены
24	New Scene view			Nové okno náhledu scény	Новое окно просмотра сцены
25	Workspace window			Okno pracovní plochy	Окно рабочего пространства
26	Console window			Konzolové okno	Консольное окно
27	Log window			Logovací okno	Журнал
28	Layouts			Rozložení	Компоновка
29	Layouts As Submenu			Rozložení jako podnabí…	Компоновка Как Подменю
30		Tutorial Menu			
31	Tutorials			Tutoriály	Туториалы
32	Reload			Znovu načíst	Перезагрузить

Obrázek 10.3: Lokalizační tabulka localization.csv

Kromě této tabulky byl napsán skript v programovacím jazyce Python ConvertCSVToLangFiles, který analyzuje tabulku a na základě hodnot z ní generuje lokalizační soubory (en.txt, cz.txt, ru.txt, ua.txt).

Bylo rozhodnuto přidat automatické spouštění tohoto skriptu při kompilaci programu v případě změny lokalizační tabulky. Za tímto účelem byl do souboru CMakeLists.txt přidán vlastní příkaz, který je zodpovědný za konfiguraci sestavení aplikace a který se automaticky spustí při změně tabulky nebo skriptu.

Tím bylo dosaženo co nejpohodlnější a nejefektivnější operace, která vývojářům, kteří se budou podílet na lokalizaci, umožňuje co nejjednodušší práci pouze změnou lokalizační tabulky.

10.9 Kontrola úplnosti lokalizace.

Dalším důležitým úkolem bylo zajistit lokalizaci aplikace a pohodlné zobrazení dosud nelokalizovaných klíčů. Proto byly vytvořeny dva skripty v jazyce Python:

1. **CheckLangFilesKeyCoverage.py**

   Zajišťuje, že všechny klíče, které se vyskytují v *.cpp/*.h, jsou přítomny v tabulce CSV.

2. **CheckUnusedKeysInLocalization.py**

   Zjišťuje všechny překlady, které jsou přítomny v tabulce CSV, ale nejsou použity v souborech *.cpp/*.h.

Pro informování vývojářů o stavu lokalizačního pokrytí bylo rozhodnuto přidat testy do GitLab CI. Bylo rozhodnuto, že testy budou nepovinné k absolvování, protože jinak by to vývojářům, kteří nemají s lokalizací nic společného, nutilo udržovat úplné lokalizační pokrytí pro všechny jazyky.

10.10 Závěr.

Vyvinutý mechanismus vnitřní lokalizace splňuje všechny formální požadavky problému:

- Byl vyvinut vlastní odlehčený lokalizační mechanismus bez závislostí na třetích stranách.
- Bylo provedeno objektivní testování: při reálném počtu volání (< 100 na snímek) nejsou náklady znatelné.
- Uživatelské rozhraní umožňuje změnit jazyk dvěma kliknutími; nastavení se ukládá mezi relacemi.
- Vestavěné nástroje (CSV + skripty) usnadňují rozšíření seznamu jazyků a kontrolu pokrytí překladem.

Kapitola 11

Testování implementovaného uživatelského rozhraní.

Poslední částí práce bylo testování implementovaného uživatelského rozhraní s uživateli. Během práce na aplikaci I3T (An Interactive Tool for Teaching Transformations) byla v aplikaci vytvořena řada nových oken, to jest nových prvků uživatelského rozhraní. Aby bylo možné získat představu o intuitivnosti a použitelnosti vytvořených oken a zjistit nedostatky implementace, bylo nutné je otestovat na skutečných uživatelích.

Testování bylo provedeno jak na lidech, kteří mají vztah ke grafice a mají zkušenosti s aplikací I3T, tak na lidech, kteří aplikaci použili poprvé. Konkrétně se testu zúčastnilo 5 osob, které jsou studenty ČVUT a absolvují (nebo absolvovaly) předmět Programování grafiky (B0B39PGR), během kterého studenti pracují s aplikací I3T. A navíc 2 osoby, které jsou studenty ČVUT, ale nikdy nepracovaly s aplikací I3T. Celkem bylo implementované uživatelské rozhraní testováno na 7 studentech.

11.1 Cíle testování.

Během práce na aplikaci I3T byl implementován systém více jazyků aplikace a bylo vytvořeno nové okno pro změnu jazyka aplikace (okno „Change Language“). Bylo nutné otestovat, jestli je pro uživatele snadné okno „Změnit jazyk“ najít a jak intuitivní je jeho obsah. Úkol pro testující zněl následovně:

1. „Změňte jazyk celé aplikace z angličtiny na češtinu.“

Dalším nově přidaným grafickým uživatelským rozhraním bylo okno pro výběr, vytváření a odstraňování rozložení oken (okno „Layouts“). Bylo nutné otestovat, jestli je pro uživatele snadné najít okno pro správu rozložení oken, jestli uživatel chápe, k čemu toto okno slouží. Dále bylo třeba otestovat intuitivnost okna „Layouts“ pro vytváření a mazání nově vytvořených rozložení oken. Úkoly byly následující:

1. „Zvolte nějaké existující rozložení oken v případě, že je aktuální rozložení oken rozbité.“
2. „Vytvořte si vlastní nové rozložení oken.“

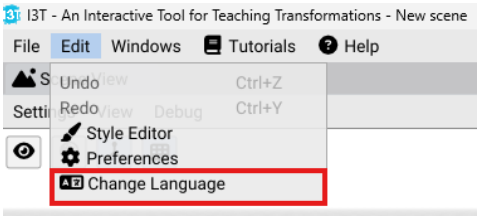
3. „Odstraňte právě vytvořené rozložení oken.“

Posledními novými prvky uživatelského rozhraní bylo přidání několika položek do stávajícího okna „Nastavení“ (Preferences) pro ovládání hlavní smyčky aplikace a vytvoření nového okna „Statistika“ (Statistics) pro sledování informací o výkonu I3T. Bylo nutné zkontrolovat, jestli jsou názvy a popisy v položkách nastavení pro uživatele srozumitelné a jestli je okno „Statistika“ snadno k nalezení. Úkoly byly následující:

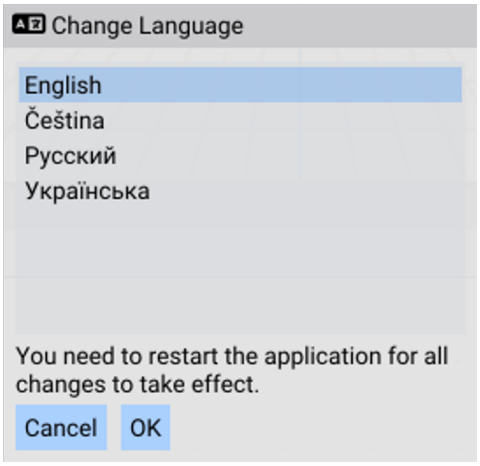
1. „Nastavte obnovovací frekvenci aplikace na 40 a sledujte změny na grafu v okně „Statistika“ (Statistics).“
2. „Nastavte obnovovací frekvenci aplikace, když uživatel neinteraguje s aplikací, na 15.“
3. „Zapněte VSync a zakažte omezování obnovovací frekvence aplikace, když uživatel s aplikací neinteraguje.“

11.2 Stav uživatelského rozhraní v době testování.

Podnabídka, která otevírá okno pro změnu jazyka, se nachází v hlavní horní nabídce „Edit“. Ona je v seznamu nabídek poslední, hned za oknem „Nastavení“ (viz Obrázek 11.1a). To je modální okno se seznamem dostupných jazyků a tlačítky „Cancel“ (Zrušit) a „OK“ (Potvrdit) pro zrušení, resp. potvrzení změny jazyka (viz Obrázek 11.1b). Je zde uvedeno upozornění na nutnost restartovat aplikaci pro přijetí všech změn.



(a) : Umístění podnabídky okna „Změnit jazyk“

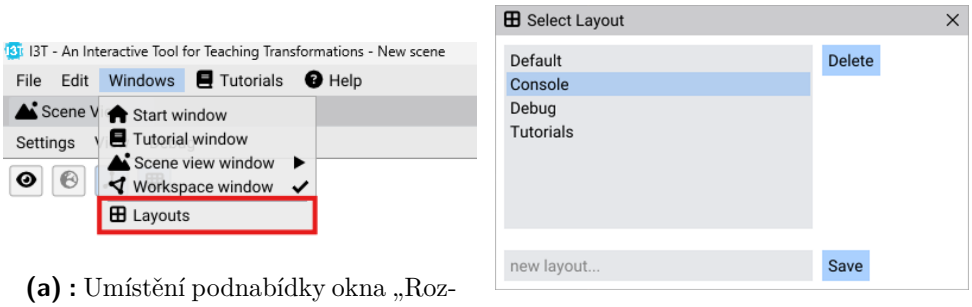


(b) : Okno „Změnit jazyk“

Obrázek 11.1: Umístění podnabídky a vzhled okna „Změnit jazyk“ během testování

Podnabídka, která otevírá okno pro správu rozložení oken (Layouts), se nachází v hlavní horní nabídce „Windows“. Ona je poslední v seznamu všech ostatních podnabídek hlavních oken dostupných v aplikaci (viz Obrázek 11.2a).

Okno obsahuje seznam existujících rozložení oken. Část z nich jsou standardní rozložení pro obvyklé použití aplikace. Ta jsou pevně nastavena autory I3T. Další jména reprezentují rozložení, která si vytvořil uživatel I3T. Okno má textové pole s možností pojmenovat nové rozložení a uložit je (viz Obrázek 11.2b). Přepínání mezi rozloženími oken probíhá výběrem požadovaného rozložení ze seznamu. Pokud je možné dané rozložení odstranit, zobrazí se vedle něj tlačítko „Delete“.



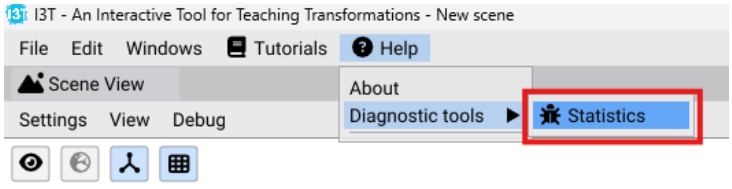
(a) : Umístění podnabídky okna „Rozložení“

(b) : Okno „Rozložení“

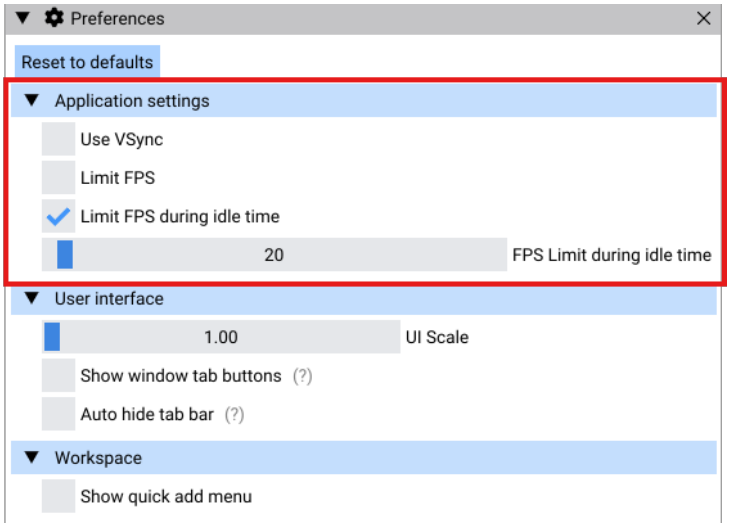
Obrázek 11.2: Umístění podnabídky a vzhled okna „Rozložení“ během testování

Položky v okně „Nastavení“ (Preferences) jsou seznamem položek pro ovládání hlavní smyčky aplikace (viz Obrázek 11.4). Položky jsou buď zaškrtávací políčka s názvem toho, za co odpovídají, nebo posuvník s názvem, který umožňuje nastavit konkrétní hodnotu. Pokud je povolena položka „Use VSync“, která je zodpovědná za zapnutí technologie synchronizace obnovovací frekvence aplikace s frekvencí monitoru, položka „Limit FPS“ se nezobrazuje a nelze ji aktivovat. Pokud je položka „Use VSync“ vypnuta a položka „Limit FPS“ (Omezit FPS) zapnuta, zobrazí se posuvník, který umožňuje určit omezení FPS. Je-li zapnuta položka „Limit FPS during Idle time“ (Omezit FPS v době nečinnosti), která je zodpovědná za omezení FPS v době, kdy uživatel nepracuje s aplikací, zobrazí se posuvník, umožňující nastavit toto omezení.

Podnabídka, která otevírá okno se statistikami aplikace (Statistics), se nachází v hlavní horní nabídce „Help“. Tlačítko, které okno otevírá, se nachází v podnabídce „Diagnostic tools“ (viz Obrázek 11.3).



Obrázek 11.3: Umístění podnabídky okna „Statistika“ během testování



Obrázek 11.4: Konfigurace cyklu aplikace v nastavení během testování

11.3 Výsledky testování.

Po testování se podařilo zjistit nedostatky v současné implementaci prvků uživatelského rozhraní.

Změna jazyka aplikace v okně „Změnit jazyk“ nezpůsobila žádnému z testujících problémy. Jeden z testujících nemohl nalézt okno v nabídce, protože narazil na podnabídku „Nastavení“ (Preferences), která se nachází před podnabídkou „Změna jazyka“, a přepnul na ni svou pozornost.

Výběr, vytváření a odstraňování rozložení oken způsobovalo některým testovaným uživatelům problémy. Ačkoli se rozhraní okna ukázalo jako jednoduché a srozumitelné – každý z testovaných si bez problémů poradil s výběrem, vytvořením i odstraněním rozložení – u dvou z nich nastal problém s pochopením, že okno „Rozložení“ (Layouts) je právě to, co hledají. Testovaní uživatelé nejprve nepochopili účel tohoto okna a pokračovali v hledání dál.

Problémy testujícím způsobovalo také omezení obnovovací frekvence aplikace v nastavení. Čtyři testující nemohli pochopit, jak povolit manuální omezení FPS, protože se tato položka nezobrazovala, když bylo povoleno „Use VSync“. Uživatelé neznají terminologii a nejsou schopni pochopit vztah mezi použitím VSync a omezením FPS.

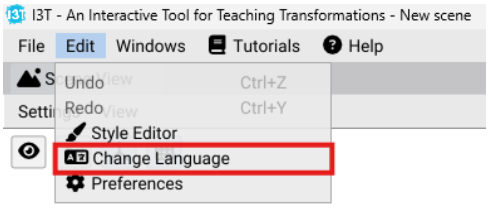
Všichni testující měli problémy s nalezením okna „Statistika“, protože se nacházelo v nabídce „Help“ (Nápověda), což si uživatelé nespojují s tím, že se tam taková okna mohou nacházet. Nastavení omezení FPS, když uživatel s aplikací neinteraguje, a zapnutí technologie VSync nezpůsobilo testovaným uživatelům žádné problémy. Po zobrazení obnovovací frekvence v okně „Statistika“ všichni testující pochopili, za co přesně jsou položky v okně nastavení zodpovědné.

11.4 Změny po testování.

Provedené uživatelské testování ukázalo nedostatky současné implementace uživatelského rozhraní a jasně identifikovalo aktuální problémy.

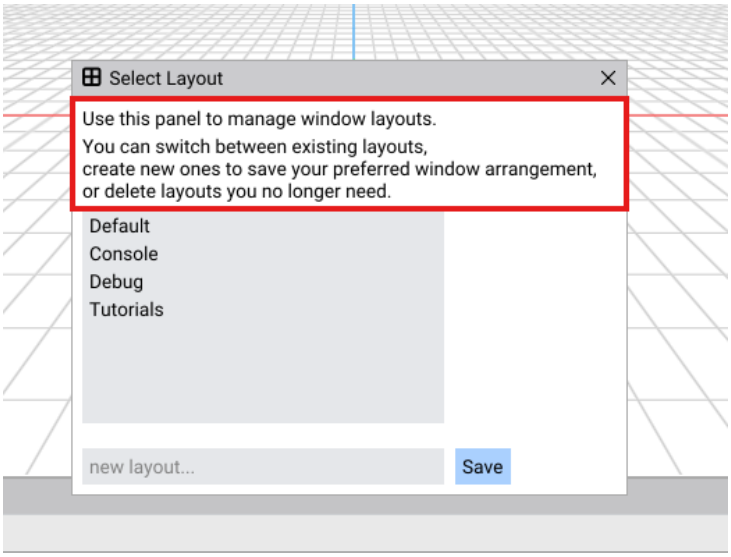
Co bylo opraveno:

- Podnabídka „Změnit jazyk“ (Change Language) byla přesunuta výše. Nyní se podnabídka „Nastavení“ (Preferences) nachází níže, což zabrání uživateli přesměrovat svou pozornost na ni dříve, než si všimnou okno „Změnit jazyk“ (viz Obrázek 11.5).



Obrázek 11.5: Upravené umístění okna „Změnit jazyk“

- Do okna „Rozložení“ (Layouts) bylo přidáno vysvětlení, k čemu toto okno slouží. (viz Obrázek 11.6)

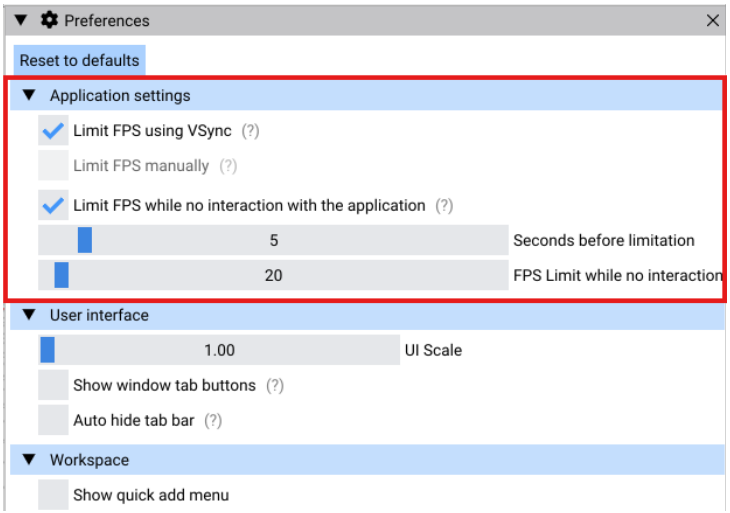


Obrázek 11.6: Upravené okno „Rozložení“

- Nastavení hlavní smyčky aplikace bylo přepracováno (viz Obrázek 11.7). Byly přidány informační bubliny, které zobrazují uživateli krátký popis jednotlivých nastavení při najetí myší na ikonu „(?)“. To pomáhá uživateli pochopit význam jednotlivých voleb v případě, že to není zcela jasné z jejich názvů. Názvy jednotlivých položek byly upraveny: položka „Use

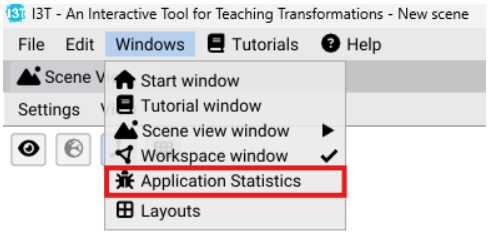
VSync“ se nyní nazývá „Limit FPS using VSync“, „Limit FPS“ byla přejmenována na „Limit FPS manually“ a „Limit FPS during Idle time“ nyní nese název „Limit FPS while no interaction with the application“, což zvyšuje jejich srozumitelnost a intuitivnost pro uživatele.

Položka „Limit FPS manually“ se nyní zobrazuje i v případě, že je aktivní „Limit FPS using VSync“, ale je v takovém případě neaktivní (zešedlá) a nelze s ní interagovat. Díky tomu uživatel vidí, že tato volba existuje, a informační bublina mu napoví, co je potřeba deaktivovat, aby ji bylo možné upravit.



Obrázek 11.7: Přepracované nastavení

- Okno „Statistics“ bylo přesunuto z nabídky „Help“ do nabídky „Windows“ za účelem zvýšení konzistence a intuitivnosti pro uživatele. Název byl změněn z „Statistics“ na „Application Statistics“ (viz Obrazek 11.8).



Obrázek 11.8: Upravené umístění okna „Statistika“

Tímto způsobem jedno kolo testování aplikace I3T, které proběhlo na 7 uživatelích, umožnilo jednoduše identifikovat nedostatky v implementaci grafického uživatelského rozhraní a odstranit je, čímž se stalo pro uživatele intuitivnější.

Kapitola 12

Závěr.

Cílem této bakalářské práce bylo plně rozšířit funkčnost interaktivního nástroje I3T. K dosažení tohoto cíle bylo provedeno pět úkolů, z nichž každý zlepšil použitelnost a spolehlivost aplikace.

1. **Ukládání a načítání pozic dokovatelných oken aplikací.**

   Ukládání a načítání pozic dokovatelných oken aplikací. Implementován mechanismus, který umožňuje ukládat a obnovovat pozice dokovatelných oken. Přidáno samostatné okno pro správu rozložení, ve kterém může uživatel vybírat přednastavená rozložení nebo vytvářet a odstraňovat vlastní. Nyní se poslední rozložení ukládá spolu se scénou a v tutoriálových souborech je možné nastavit rozložení, se kterým bude aplikace spuštěna. To vše výrazně usnadnilo práci s nástrojem.

2. **Profilování a optimalizace výkonu aplikací.**

   Profilování umožnilo pochopit slabá místa aplikace, zatímco optimalizace vedla k 12% nárůstu výkonu a optimalizaci hlavního cyklu aplikace s cílem ušetřit počítačové zdroje.

3. **Korektní fungování historie akcí (Undo/Redo).**

   V aplikaci byl obnoven a vylepšen mechanismus uchovávání historie: operace Undo a Redo jsou nyní prováděny stabilním a předvídatelným způsobem, což uživateli poskytuje jistou kontrolu nad změnami.

4. **Logování aktivity uživatele.**

   Byl zaveden systém logování interakcí, který zaznamenává akce uživatelů. Shromážděná data lze nyní použít k další analýze.

5. **Vícejazyčné rozhraní.**

   Aplikace je nyní plně lokalizována do tří jazyků. Bylo vytvořeno okno pro změnu jazyka, aby uživatel mohl přepínat mezi lokalizacemi bez nutnosti restartu.

Pro ověření implementované funkčnosti byla provedena fáze testování uživatelského rozhraní, jejíž výsledky pomohly včas identifikovat a odstranit problémy.

Cíle tak byly beze zbytku splněny: nástroj I3T se stal produktivnějším, flexibilnějším a uživatelsky přívětivějším. Provedená vylepšení otevírají možnosti dalšího rozvoje projektu, včetně rozšíření souboru lokalizací, hloubkové analýzy uživatelských scénářů a další optimalizace výkonu.

Literatura

[1] I3T (An Interactive Tool for Teaching Transformations). *Github* [online]. [cit. 23.05.2025]. Dostupné z: `https://github.com/i3t-tool/i3t`

[2] Aymar Cornut. Dear ImGui Bloat free Graphical User interface for C++. *Github* [online]. [cit. 23.05.2025]. Dostupné z: `https://github.com/ocornut/imgui`

[3] First look at profiling tools (C#, Visual Basic, C++, F#). *Microsoft* [online]. [cit. 23.05.2025]. Dostupné z: `https://learn.microsoft.com/en-us/visualstudio/profiling/profiling-feature-tour?view=vs-2022&pivots=programming-language-dotnet`

[4] NVIDIA Profiling Tools. *Nvidia* [online]. [cit. 23.05.2025]. Dostupné z: `https://developer.nvidia.com/tools-overview`

[5] NVIDIA Nsight Graphics. *Nvidia* [online]. [cit. 23.05.2025]. Dostupné z: `https://developer.nvidia.com/nsight-graphics`

[6] NVIDIA Nsight Systems. *Nvidia* [online]. [cit. 23.05.2025]. Dostupné z: `https://developer.nvidia.com/nsight-systems`

[7] Martin Herich. „Skriptování v nástroji I3T a automatizace zveřejňování na GitHubu“. Diplomová práce. FEL ČVUT. 2025*Handle* [online]. [cit. 23.05.2025]. Dostupné z: `http://hdl.handle.net/10467/120310`

[8] Filip Uhlík. „Logovací systém pro nástroj na výuku transformací I3T“. Bakalářská práce. FEL ČVUT. 2020 *Handle* [online]. [cit. 23.05.2025]. Dostupné z: `http://hdl.handle.net/10467/87647`

[9] Sleep function (synchapi.h). *Microsoft* [online]. [cit. 23.05.2025]. Dostupné z: `https://learn.microsoft.com/en-us/windows/win32/api/synchapi/nf-synchapi-sleep`

[10] Localization Keys vs. Direct Text Keys. *Dev* [online]. [cit. 23.05.2025]. Dostupné z: `https://dev.to/spyke/localization-keys-vs-direct-text-keys-5i`

[11] Prusa Slicer. *Github* [online]. [cit. 23.05.2025]. Dostupné z: `https://github.com/prusa3d/PrusaSlicer/releases`

[12] Blender. *Github* [online]. [cit. 23.05.2025]. Dostupné z: `https://github.com/blender/blender`

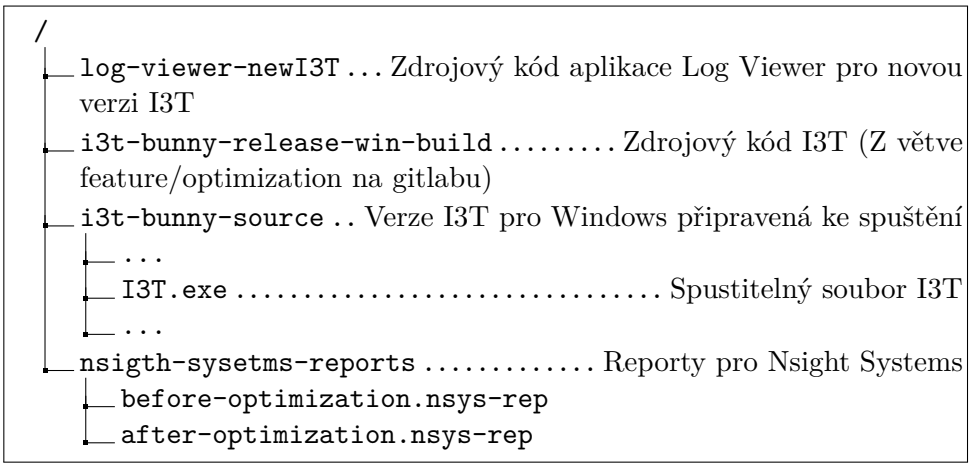
[13] Fórum „What to use instead of glfwSleep?“. *Fórum GLFW* [online]. [cit. 23.05.2025]. Dostupné z: `https://discourse.glfw.org/t/migrating-to-3-0-what-to-use-instead-of-glfwsleep/577/3?page=2`

[14] Hello ImGui. *Github* [online]. [cit. 23.05.2025]. Dostupné z: `https://github.com/pthom/hello_imgui`

[15] Python Software Foundation, Documentation Python. *Python Docs* [online]. [cit. 23.05.2025]. Dostupné z: `https://www.python.org/doc/`

Příloha A

Obsah přiloženého CD



Obrázek A.1: Struktura přiloženého CD