Bachelor Project

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of computer graphics and interaction

Tutorials for the Tool for Teaching Transformations (I3T)

Miroslav Müller

Supervisor: Ing. Petr Felkel, Ph.D.
Field of study: Open informatics
Subfield: Computer games and graphics
August 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Müller** Jméno: **Miroslav** Osobní číslo: **474749**

Fakulta/ústav: **Fakulta elektrotechnická**

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

Studijní program: **Otevřená informatika**

Studijní obor: **Počítačové hry a grafika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Tutoriály pro nástroj na výuku geometrických transformací

Název bakalářské práce anglicky:

Tutorials for the Tool tor Teaching Transformations (I3T)

Pokyny pro vypracování:

Nástroj na výuku transformací (I3T) [1, 2] doplňte o systém pro provádění tutoriálů a uvítací obrazovku pro správu tutoriálů a souborů scén. Přitom vyjděte z grafických návrhů na nové uživatelské rozhraní [3, 4] a využijte knihovnu Dear ImGui [5], kterou jste vybral a otestoval ve svém semestrálním projektu.
Proveďte rešerši metod používaných v tutoriálech (od prostého zadávání kroků, přes ověřování splnění kroků a napovídání uživateli, jaký ovládací prvek má použít, po přímé ovládání aplikace z tutoriálu (vložení části scény, úprava hodnot ve scéně)).
Vyberte vhodnou formu výkladu v tutoriálu, aby bylo vyváženo množství obsahu (textu, obrázků) a uživatelských akcí a zvolte vyhovující způsob implementace provázání tutoriálů s aplikací (definice a načítání obsahu tutoriálu, zasílání příkazů do aplikace).
Implementujte ukázkové tutoriály seznamující s ovládáním aplikace a výukový tutoriál na vybrané téma.
Diskutujte i možnost realizace tutoriálu formou jednoduché hry.
Pracujte dle metodiky User Centered Design a řešení vylepšujte na základě spolupráce v řešitelském týmu a na základě testů s uživateli.

Seznam doporučené literatury:

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

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: **10.10.2020** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **30.09.2022**

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

Acknowledgements

I would first like to thank my supervisor, professor Ing. Petr Felkel, Ph.D. for his expert knowledge, positive attitude, and patience. Thank you for leading regular meetings with our working group, they helped us to stay motivated and interconnected between each other's works.

I would also like to thank my parents, which have been a large support throughout this journey. You were always there for me, offering everything I needed during my work.

In addition, I would like to thank my brother and my friends which have kept me sane and smiling in between my working sessions.

Declaration

I declare that the work presented here is my own and that I have listed all the used literature.

Abstract

The interactive tool for teaching 3D transformations (I3T) is an aspiring project developed at the Department of Computer Graphics, FEE, CTU. This tool was missing a solid system for creating and conducting lectures in it. The purpose of this thesis was to follow previous works on I3T and implement a tutorial system for the I3T tool. Not many similar tools exist to this date with such system. I have realized a research on methods used in tutorials, as well as an updated proposal for the structure and visual design of the system. Subsequently, I have implemented the system in I3T and a performed test with users on a demo tutorial. I have made it possible for teachers to create new tutorials. Some of the proposed features have not been implemented yet and further development is recommended. I have provided a discussusion about such features.

Keywords: I3T, tutorial system, tutorials, 3D transformations, computer graphics, interactive transformations, teaching

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

Abstrakt

Interaktivní nástroj I3T určený k výuce 3D transformací je projekt rozvíjený pod Katedrou počítačové grafiky a interakce, FEL, ČVUT. Uvnitř tohoto nástroje dosvad scházel systém pro tvorbu a realizaci lekcí. Cílem této práce bylo následovat předchozí práce na nástroji I3T a implementovat do něj systém pro tutoriály. Není zatím mnoho podobných nástrojů, které by obsahovaly takový systém. Zrealizoval jsem výzkum metod používaných v tutoriálech a aktualizoval stávající návrhy na formu a vizuální styl tutoriálů. Následně jsem implementoval tento systém do I3T a provedl na něm test s uživateli na zkušebním tutoriálu. Některé z návrhů nebyly zatím implementovány a proto doporučuji další rozvoj I3T včetně tohoto systému. K těmto návrhům jsem provedl diskuzi.

Klíčová slova: I3T, systém tutoriálů, tutoriály, 3D transformace, počítačová grafika, interaktivní transformace, výuka

Překlad názvu: Tutoriály pro nástroj na výuku geometrických transformací

Contents

Figures

Tables

Chapter 1
Introduction

In this work, I describe the design and implementation of a tutorial system for the I3T tool. The tutorials are supposed to teach students how to use I3T and topics surrounding 3D transformations.

First, I introduce I3T and specify the reasons for my work. In the next chapter, I do a research on methods used in tutorials. In the central part of my work, I discuss my design for the tutorial system [1]. I proceed with its realization in I3T and the implementation details underlying this system. Furthermore, I go through testing of the tutorials with users. I also discuss my results and future possibilities for I3T's tutorials.

1.1 About I3T

I3T is an educational software which enables the study of 3D transformations and their hierarchy in an illustrative way. Its name "I3T" is an abbreviation of "Interactive Tool for Teaching Transformations". It was originally a result of a master's thesis done by Michal Folta in 2016 [1]. Further development was led by Folta's supervisor Ing. Petr Felkel, Ph.D. under the Department of Computer Graphics and Interaction, FEE, CTU in Prague.

Based on the following works and testing, it was decided to redesign and update its GUI and add additional features like the tutorial system. I will be referring to Michal Folta's version as the "old" version and the one currently in development as the "new" version.

1.1.1 Old version

The tool is composed of two main parts: "space" and "view" [old naming as in 1, Folta]. The space is an extendable subwindow from the bottom where the current "scene" is to be formed. The subwindow cannot be taken out of the main window nor be snapped to another side. View shows the current scene in 3D from an independent camera. These subwindows can be seen in fig. 1.1.

[1] Note that tutorial examples presented in this work include texts that have not yet been translated to English.

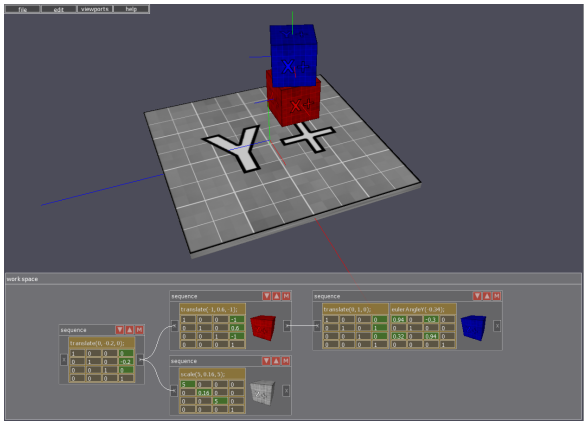

Figure 1.1: Space and view in the old version of I3T

Matrices, sequences, operators, and many other blocks can be added to the space to form a scene graph. The main educational effect is achieved through letting the user interactively change the values of the blocks and observing their behaviour in the view.

Learning materials

The only way for self-study in the old version are preprepared example scene files which can be loaded. These scenes demonstrate various important concepts from the topic of 3D transformations. Most of those scenes contain an additional floating window with their description as shown in Figure 1.2. The description is usually a mix of theory and tasks for the user to try in that scene.

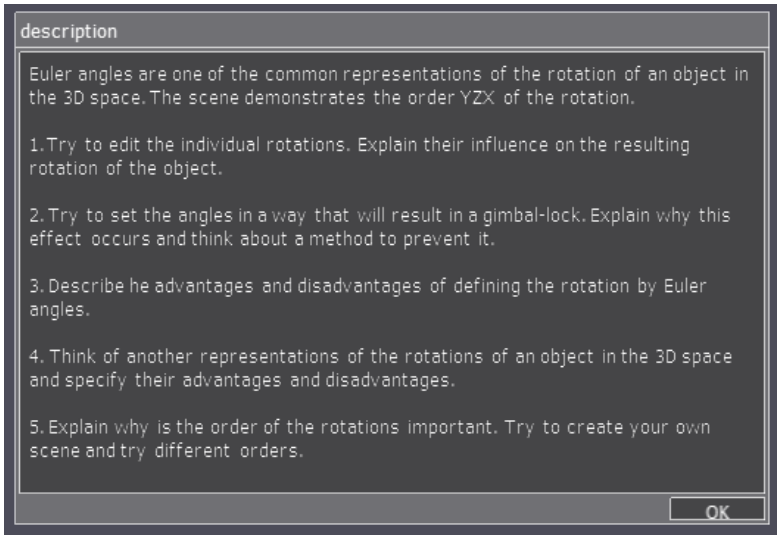

Figure 1.2: Scene description in the old version of I3T

1.1.2 User testing, problems and following designs

Results of various tests were available to me through a shared online directory for I3T. Early tests with users on I3T's teaching effectivity had shown either an improvement or no change in results with I3T. This suggested that while I3T might have been doing well, there might still be a space for improvement in this area. A usability test pointed out a few problems with its GUI and controls, and some of them were fixed. Nevertheless, some problems and possibilities for further evolution of I3T remained. Two important works followed.

Pilka's work

Based on an internal communication with Felkel, Pilka has created a refreshed graphical design for I3T's GUI in 2018 [2]. Apart from new designs of blocks and menus, those designs included a window for selecting scenes or tutorials (fig. 1.3) and a tutorial window (fig. 1.4) for the first time. I will be referring to the tutorial selection window as the start window in my work.

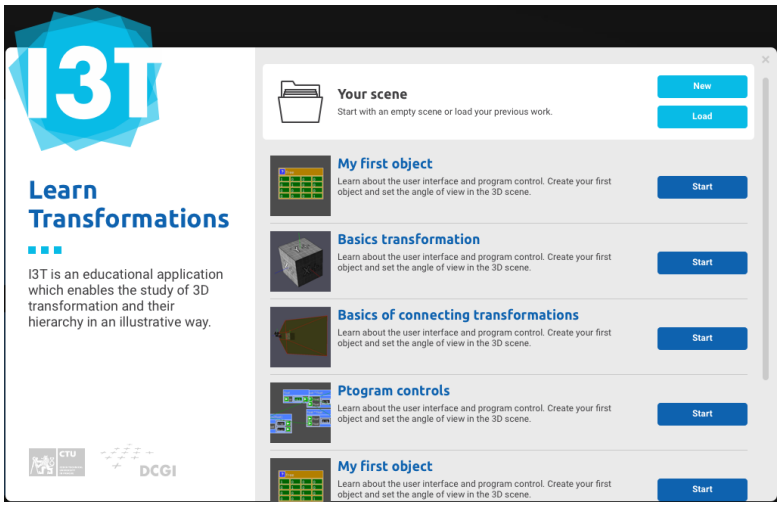

Figure 1.3: Start window design by Pilka [2]

The tutorial was supposed to be divided into thematic lessons as seen in the start window. Together with Felkel, they created 4 example lessons in a word document. This document can be found in appendix A in the directory `/tutorial concepts Felkel and Pilka`. In this document, each lesson was composed of approximately 10 steps. Each step consisted of:

- the step's title
- an explanation
- an assignment
- instructions on how to complete the assignment (optional)

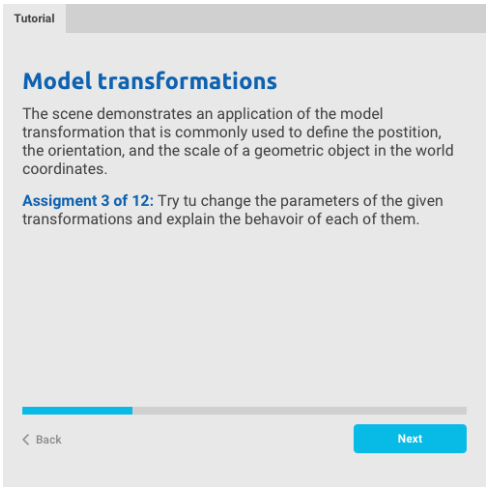

Figure 1.4: Tutorial window design by Pilka [2]

Steps in the lessons had various lengths depending on the topic. Some steps did not even have an explanation or an assignment, although at least one of those two was always present.

Design images that were available to me had shown only a single step of a lesson (fig. 1.4). The step does not portray how instructions for the assignemnt should look. It also does not portray how longer texts should be handled. Multiple steps in the document seemed to me that they would exceed the available space of the window with this design.

Zadina's work

Zadina defended his bachelor's thesis on usability testing of I3T in 2019. He also created an interactive prototype of his own design which he tested the users on.

In his initial analysis, he had pointed out that a lot of users did not even know about the preprepared scenes in I3T and that a step-based approach should be much better than static descriptions. His design thus included a tutorial window similar to Pilka's design. The form of lessons and steps remained, although the steps were slightly different. I will describe the changes on two images from Zadina's work shown in fig. 1.5.

A step in Zadina's design is consisted of:

- the title of the lesson
- the description of the lesson
- an assignment
- hint(s) on how to complete the assignment (alternative design, optional)

This is a change opposed to Pilka's document, where

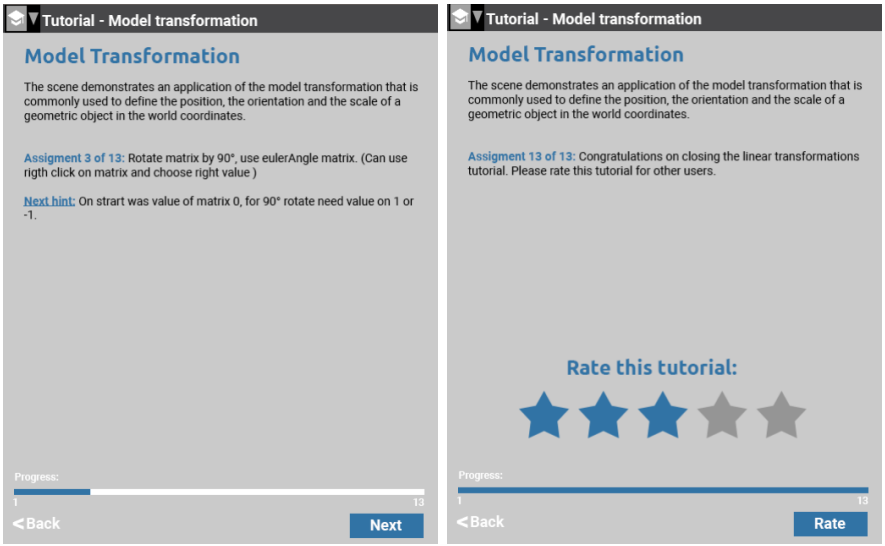

Figure 1.5: Tutorial window designs by Zadina [3]

This image includes hints which were only considered later in his work and were not present in the initial images and testing. Zadina did not mention whether those hints should be expandable on click, but I assumed so.

It seems that Zadina had thought of the lessons as smaller units than Pilka. The theory could then be explained in the description that would always be visible.

One of additional Zadina's features is a blue highlighting of parts that the user is supposed to interact with during the tutorial. That can be seen in fig. 1.6. Another new feature is the rating of tutorials as seen in fig. 1.6.

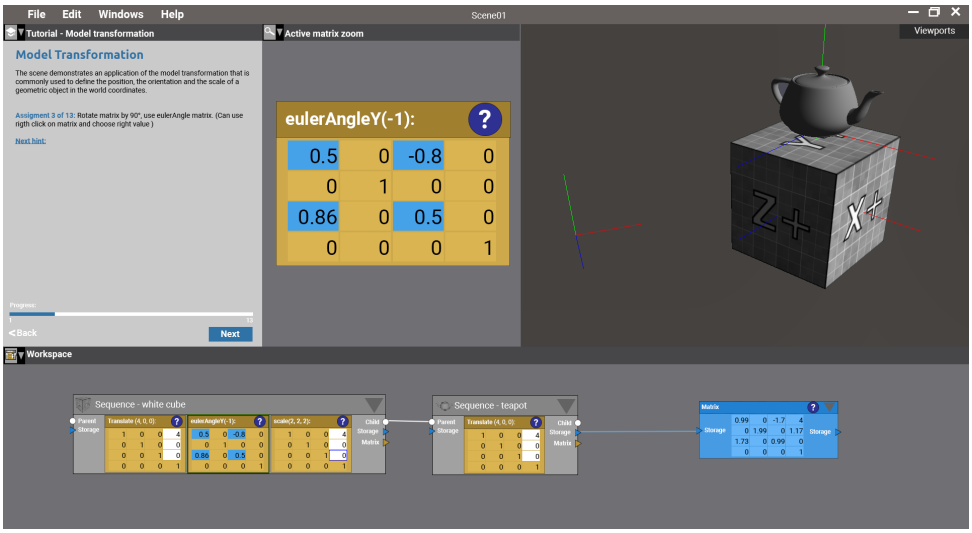

Figure 1.6: I3T design by Zadina [3]

He does not include his own visual aesthetics for the start window and only shows the Pilka's version. Nevertheless, he depicted adding templates to the

Pilka's version (fig. 1.7).

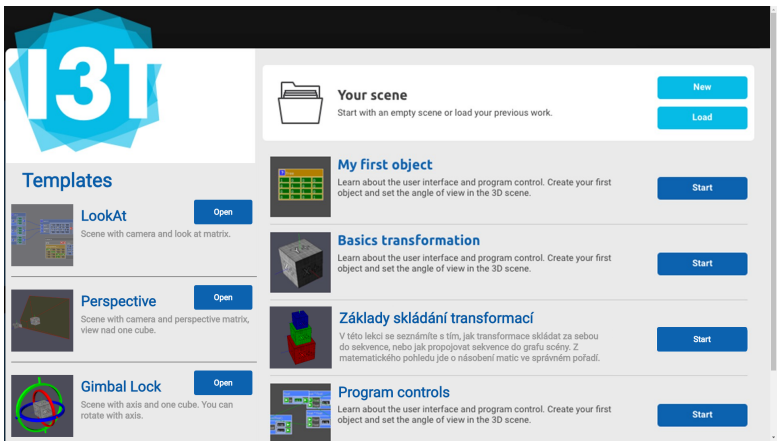

Figure 1.7: Addition of templates to Pilka's start window by Zadina [3]

1.1.3 New version

The project's leader and my supervisor Ing. Petr Felkel, Ph.D. planned a significant update of I3T. The main part to evolve was the GUI. There were already the mentioned proposals on its update. This brought up the need for using an external library to handle it.

Part of the update was the addition of a tutorial system, 3D manipulators, scripting system and changes to the backend and connection to a new external GUI library.

Names of space and view were to change to "worskpace" and "scene vieport".

1.2 My semestral project

In my semestral project [4] preceding this work, my goal was to get familiar with I3T's code and explore GUI libraries suitable for I3T. Based on that, I had to propose a GUI library for the new I3T. Citing from the project, the requirements for the library were:

- Usable with C++ binding
- Enabling dockable windows
- Supporting OpenGL rendering
- Being cross-platform (Windows and Linux at least)
- Allowed to be used in an open-source project

Bonus points were given for:

- Supporting HTML rendering (for example, for a tutorial window)

These requirements narrowed down the possibilities. After considering application frameworks like Qt or WxWidgets, I chose Dear ImGui [5] as the most promising candidate. Finally, I made a successful proof of concept demonstrating the usability of ImGui in I3T.

1.2.1 About Dear ImGui

Compared to other options, ImGui is smaller in size and easier to integrate into an existing project like I3T. At its core lies a single header file accompanied by an implementation file that binds it to a specific platform. The implementation file can be custom made, although ImGui already offers implementations for common utility libraries and rendering APIs. In our case, this is GLFW and OpenGL.

Its name is an abbreviation of "immediate mode graphical user interface'. It uses the IMGUI paradigm for GUI implementation, which differs from the other libraries. This might both be an advantage and a disadvantage depending on the situation. I3T is a big project, which means it might miss a better separation of GUI and code with ImGui in some situations. On the other hand, it might well benefit from ImGui's direct approach and customizability.

1.3 I3T's development in 2020/21

This academic year, the new version was to be implemented. Multiple students joined the the team with their projects.

1.3.1 Preparation works

First, I3T had to be prepared for use with the ImGui library. Two students from the team joined a preceding summer job to do that. The result was the old version of I3T wrapped in ImGui windows and prepared for development.

1.3.2 The team

There were six students including me, who started working on the project. We made regular calls together during the year and discussed various problems and topics. Here is a brief summary of the original goal of each other member:

Martin Herich Restructure and connect the core of the software to the new GUI library. Style the general GUI according to the new designs and enable style editing.

Šárka Švábová Test the new version with users during its development and help to shape its design in a user-centered manner.

Daniel Gruncl Implement 3D manipulators and a scripting system.

Sofia Šašorina Convert the workspace from the old version to an implementation using ImGui. Base it on the new designs.

Jaroslav Holeček Process data from the logger and propose, implement, and test an intelligent assignment suggestion system for teaching purposes.

Members and their goals slightly changed during the year, more on that in chapter 6.

1.4 Goals of this work

Given these points, I would like to summarize the goals of my work. First goal is the analysis of previous designs and general methods used in tutorials. Following is choosing a suitable form of tutorials in I3T and finding a complying way for their implementation. The expected result is a complete implementation of the tutorial system in I3T. This includes a start and a tutorial window created using the ImGui library. The system should offer an appropriate way for defining tutorials and include at a few demo examples on the usage of I3T and teaching topics from 3D transformations. The tutorial system should be created with user-centered design in mind.

Chapter 2

Research of methods used in tutorials

The goal of the research was to find and inspect already existing software that would use some form of tutorial. Consequently, I have used the learned knowledge as an inspiration for my design of the tutorial system described in chapter 3.

Before I list the examples, I would like to make a distinction between teaching the usage of I3T and teaching geometrical transformations through I3T. There are different approaches that might be better for each of the cases. Since I3T tutorials are supposed to cover both cases, I have searched for sources both for teaching the usage of a software and teaching through software in general.

2.1 Educational software

Since I3T is an educational software, it is relevant to look at software trying to achieve a similar goal.

2.1.1 Math tutoring

A paper by Paiva, Ferreira, and Frade describes a web-based software with an intelligent tutorial system [6]. This software uses small self-paced modularized units of educational contents, including tutorial videos, notes, and formative e-assessment with personalized feedback.

Their focus is that students with different backgrounds come into the classes and that the lessons should be individually adjusted. I3T could also account for that.

One option are automatically adjusted lessons like in this paper. This was a future goal of Jaroslav Holeček in our team, so I have taken that into account during implementation.

The other option is letting the user pick lessons which fit him on his own. There could be a drawback in this, since beginner lessons might teach the usage of I3T. An advanced user might thus skip those, but will not be able to properly use the features if there will not be any possibility to get hints about basic things. One solution could thus be properly offering optional

hints throughout the I3T experience wherever possible. The other solution to this could be offering the same lectures in various difficulties.

In summary, the key thing in their design was the division into small modules of teaching materials for self-study.

2.2 Software tutorials

I will explore two examples of software tutorials.

2.2.1 Unreal Editor

Unreal Engine is a popular game engine. Its Unreal Editor contains a tutorial system.

On the first startup, the user is suggested to start a welcome tutorial which teaches about the main parts of the program. After the welcome tutorial ends, the user is presented with a window that allows the selection of further tutorials. This window reminds the proposed start window. Some tutorials are nested inside broader topics which the user can expand.

The tutorials are presented using floating text boxes as seen in fig. 2.1. The tutorial is made of steps and navigated by buttons, similar to our case.

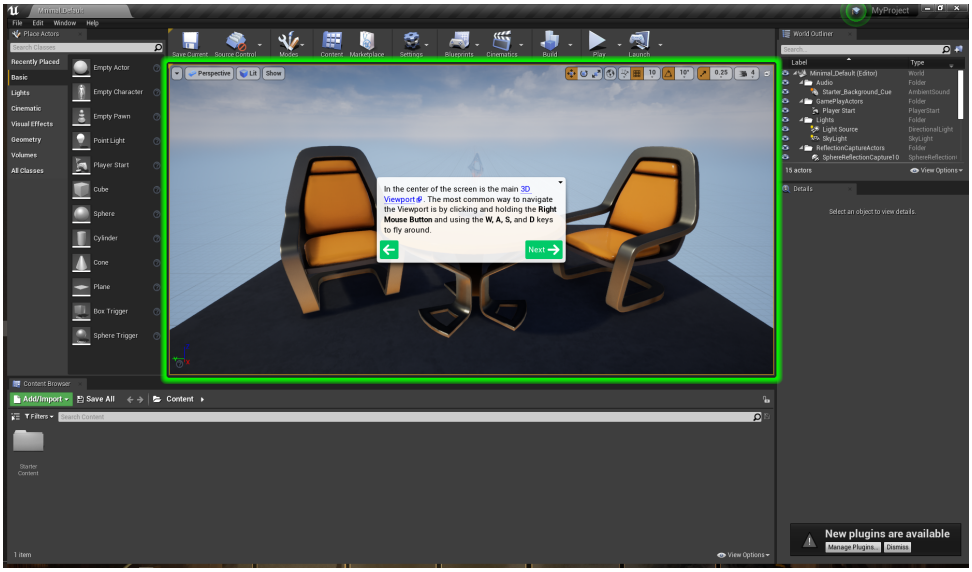

Figure 2.1: Tutorial in Unreal Editor 4.26.2 highliting the 3D viewport

The steps contain only information and no direct tasks for the user. The user is left to try on his own during the tutorial.

When showing specific parts of the GUI, a green highlighting of that part is used. Tutorials can also contain bold text and links to documentation or other sources. The floating box has an animated pop-up effect and the green highlighting is pulsing. These subtle animations make it a little more lively.

In summary, Unreal Editor is a worthy inspiration for I3T in the area of teaching the use of I3T.

2.2.2 GamiCAD

The work by Li, Grossman, and Fitzmaurice presents GamiCAD, an interactive tutorial system for first-time AutoCAD users [7]. Citing from the work, "It uses a software event driven finite state machine to model a user's progress through a tutorial, which allows the system to provide real-time feedback and recognize success and failures."

The tutorial system starts with a "Mission console" which is a top-level interface for choosing "Missions". Each mission has its own page with a description and a set of "Levels" that the user goes through. Each level has a clear goal and the user gets a score after its completion. After a level is started, the level's goal is described in the "Task panel" and can include an image. The user can use the hint feature that reveals step-by-step instructions. These steps are automatically checked and the tutorial provides the user with immediate feedback on his correct actions. In addition, it detects incorrect actions and potentially even suggests the user to restart the level. A work-in-progress tutorial can be seen in fig. 2.2,

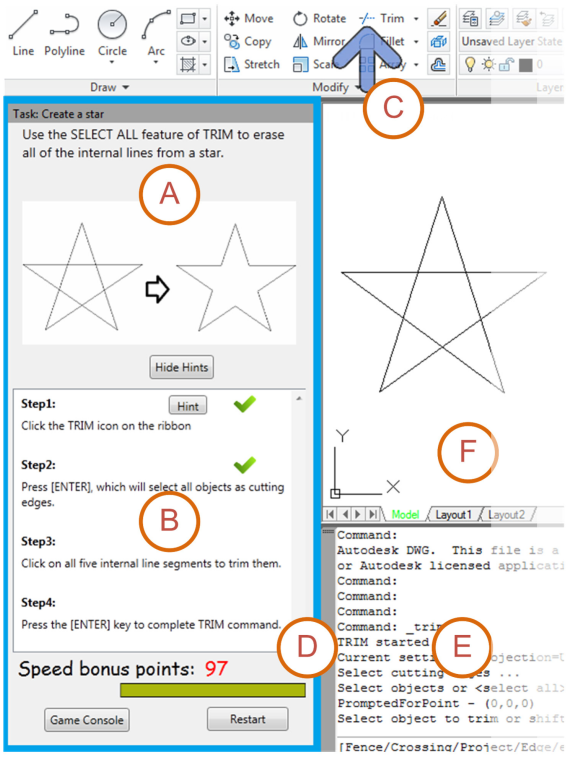

Figure 2.2: GamiCAD's: Task panel (A) Task description; (B) Step-by-step instructions; (C) UI element visual aid; (D) Speed bonus; (E) AutoCAD command line; (F) Drawing area. [7]

All these features serve as a remarkable inspiration for I3T's tutorials and their possible further gamification.

2.3 Game tutorials

Games often contain some form of tutorials. Even though they are a slightly different topic, they were also a perspective inspiration to me. I found the following popular source to be a helpful summary of takeaways from game tutorials.

2.3.1 Tutorials 101

Tutorials 101 - How to Design a Good Game Tutorial is a video from an educational YouTube channel Extra Credits. Based on the experiences and knowledge of people from the game industry, it shows a summary of "rules" that a game tutorial designer should keep in mind. Although these rules are meant for game tutorials, I found them considerably relevant to I3T. I have listed the simplified forms of the rules along with my commentary in table 2.1. Following are two further tips from the video with my commentary in table 2.2.

Less text	I find this rule essential for I3T. The previous user tests have also shown that this is an important factor. For interactivity, the video suggests letting the user play through the actions they are supposed to learn when possible.
No front loading	This rule states that teaching should be gradual and not teach every control and feature at start. This could be applicable in I3T by not teaching everything about I3T in the first tutorial and progressively uncovering features in future lessons when they are needed.
Make it fun	Making a tutorial fun is usually hard to execute and depends on the tutorial's context. In I3T, that might mean creating challenging tasks, letting the user build an interesting or funny scene, speaking in an engaging manner, or making a joke.
Reinforce learning through play	This rule suggests letting the user use his gained knowledge after the tutorial. This should reinforce his knowledge. In I3T, tutorials could use learned concepts from previous steps or tutorials.
Listen to your players	This could be translated to listen to your students for our case. Knowledge about the target demographic should be taken into account and proper testing with users should be made.

Table 2.1: Tutorial 101 rules and commentary

Tutorials should be skippable or should not interrupt the flow of the play	This is meant mostly for game replayability. The takeaway could be - do not force the students to do the tutorials in I3T and let them be optional. During a tutorial, students should feel comfortable to progress and not feel forced to do something they do not want.
Anything you put in should be accessible at all times	This means that any lessons taught in the tutorials should be available to the player later. For I3T, I can imagine a manual window with all controls clearly listed. Blocks and connections could also have their explanations in there.

Table 2.2: Tutorial 101 further tips and commentary

Chapter 3

Designing the tutorial system

Taking inspiration from the designs by Pilka [2] and Zadina [3], I had to prepare the structure for the implementation of the tutorial system. From the user perspective, I tried to make it to meet these points:

- I3T developers, teachers, or interested students can create new tutorials for I3T
- Creating new tutorials should not be overly complicated
- Tutorials should be easily portable
- I3T lists available tutorials and allows the user to choose
- I3T guides the user through the selected tutorial
- The experience should not be boring nor overwhelming

3.1 General idea

I followed the idea of dividing the tutorial system into individual lessons or "tutorials", where each tutorial is a sequence of steps. I prepared two windows for that — a welcome window with a selection of tutorials and a tutorial window displaying a chosen tutorial step-by-step. I based their visual aesthetics on Pilka's design, since Zadina's tutorial window design did not provide enough visual consistency with the welcome window. As for tutorial creation, I decided to create a special folder in I3T where tutorial files would be stored and loaded from. I chose which format will these files use and how will the loading process work. I created demo tutorials to test on. Any refrenced files are located on the enclosed CD whose directory structure is described in appendix A.

3.2 Tutorial window

Individual tutorials will be presented in this window. It was up to me to decide how this will be done. There was one major problem that the users had

with the tutorial in Zadina's testing. The tutorials still contained too much text. Both my research and intuition suggested to focus on minimalizing plain text. I kept that in mind and tried to find alternatives when possible.

3.2.1 Early design

My initial thoughts were focused on what I was able to work with. This meant exploring I3T, previous designs and available libraries. After I got a rough idea on how the tutorials could work in practice, I also started refining these ideas in concept.

I had created a presentation discussing various design possibilities. The full presentation can be found on the CD at `/designs/Navrhy tutorialu.pptx`. The following text in this subsection revolves around the contents of that presentation.

Base of the design

I started with the idea that tutorials consist of explanations of the subject matter, assignments for the user to complete, and instructions on how these assignments can be completed. These will make up the contents of the steps in the tutorial window.

Nor Pilka's nor Zadina's design does account for when the contents of a step are too long. For this reason, I divided the window into two sections — "tutorial contents" and "control panel" as seen in fig. 3.1. The Tutorial

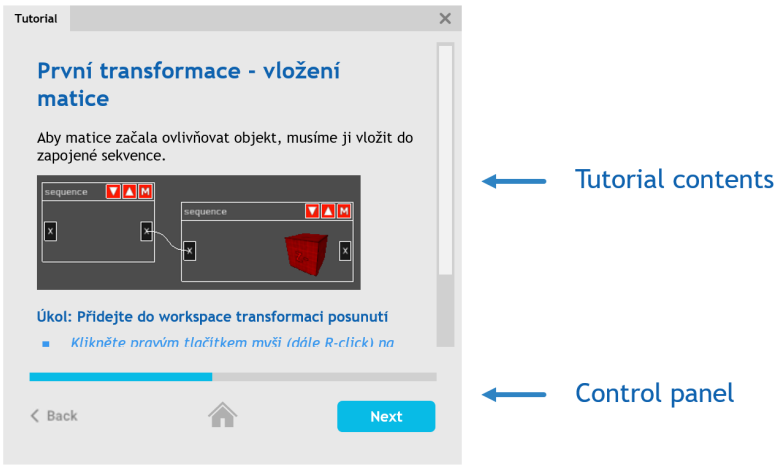

Figure 3.1: My design of the tutorial window with overflow solution

contents section becomes scrollable once its full height is reached. For the first time, pictures are used in it.

Another slight difference to the previous designs may be seen in the control panel. Since users were missing an easy way of getting back to the welcome window in Zadina's testing, I tried adding a "home" button.

Tutorial content features

I discussed the possibilities of what "features" the creators could add to the tutorial.

Headers and text. This is the most basic feature. A description of the step could be written in text, while a differently formatted header would introduce it at the top. For the header, text, and background, I chose the same fonts and colors as Pilka.

Images. By including images, the presentation can be much more lively. Images also allow to further reduce the text by portraying explanations, assignments, and hints visually instead. This is why I presented them as an essential feature.

Assignments and instructions. I followed by including differently highlighted assignments and instructions. I discuss their details in section 3.2.1.

Text formatting. Allowing creators to further format the text on their own could help stress out important parts of the explanation. On the contrary, there is a risk of loss of a clean design with these features. A demonstration of bold, italic, underlined, and colored text as well as a bullet list can be seen in fig. 3.2.

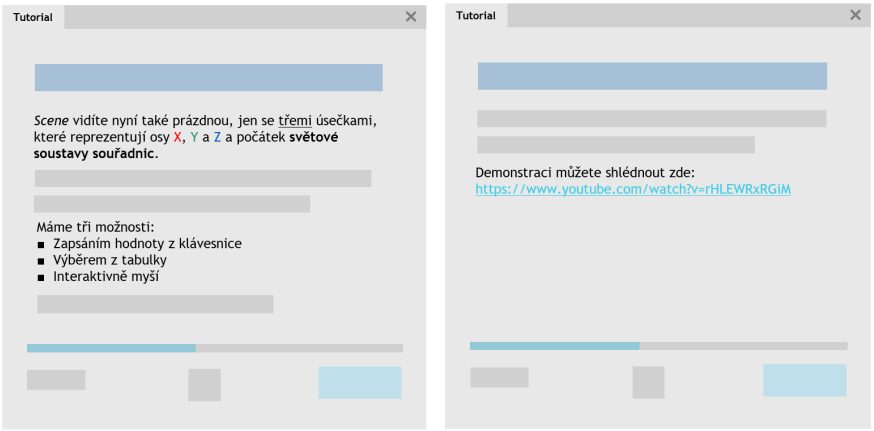

Figure 3.2: Demonstration of text formating (left) and a URL (right)

URLs. Some complex topics may require more space for explanation than a tutorial in I3T can offer. Creators may also want to add references to additional materials. In such cases, links to external URLs may be useful. An example of a link in the tutorial can be seen in fig. 3.2.

Animated images. This is a next level of imagery. Animated images or "GIFs" can portray more complex actions, visualize mathematical concepts, and more. This feature needs more work than regular images, but could definitely be achieved.

Math equations. Since I3T is meant teaching 3D transformations, displaying of math equations might be desired. It is not an easy feature for a custom implementation, but could be probably achieved through the use of external formats and libraries. An alternative solution to this is including the equations as images rendered outside I3T. An example is in fig. 3.3.

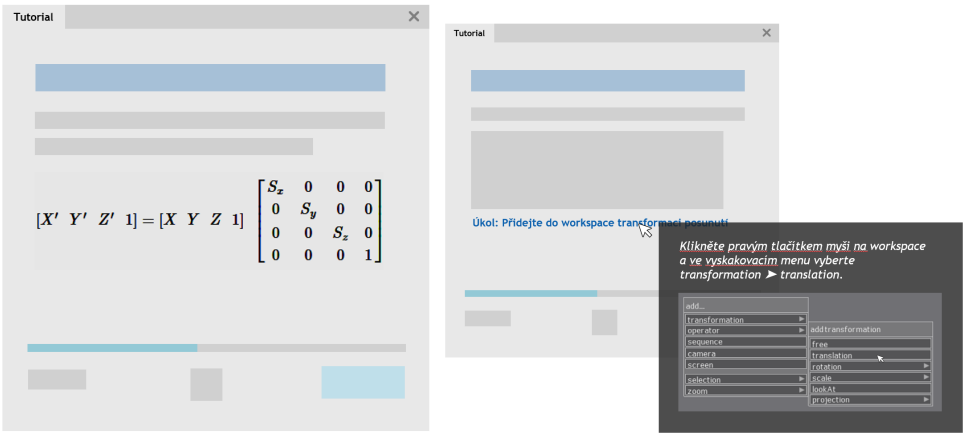

Figure 3.3: Demonstration a math equation (left) and instructions in a tooltip (right)

Tooltip hints. Tooltips are an alternative way of displaying hints and instructions. They add interactivity and information on demand. A tooltip displaying instructions for an assignment can be seen in fig. 3.3.

Advanced features

Next, I discussed more complex features that would interact with other parts of I3T.

Overlay. This feature puts an overlay over other parts of I3T. An overlay could dim unimportant parts, highlight important ones, display an arrow or instructions for the current assignment. An example demonstrating all of these can be seen in fig. 3.4. There are various "levels" of this feature.

The first one would allow the creator to choose a window and possibly accept relative coordinates of where to make the highlight. This approach is limited, but might be enough for some cases.

The next one would correspond to Zadina's blue highlighting of specific fields in specific matrices. The creator would need to specify which blocks, fields, or objects will be highlighted. Zadina discovered a problem that most of the users did not notice a change of a color of matrix fields. This could be fixed by combining the color change with an additional overlay, such as an arrow. This approach requires much more integration and effort from the side of a creator, but could achieve interesting results.

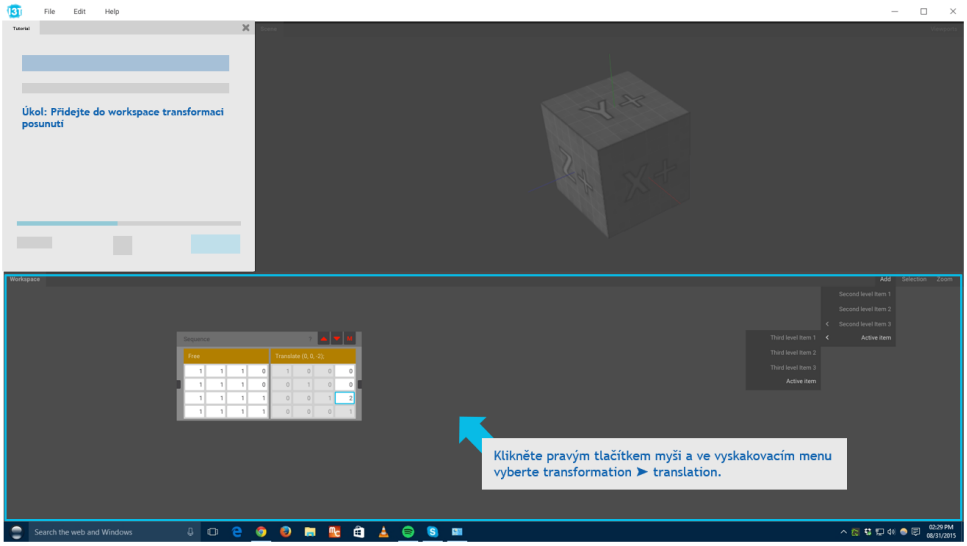

Figure 3.4: Demonstration of a dimmed viewport, highlighted workspace and an arrow with instructions of an assigment

Additional floating window. Adding a second window that floats above I3T as in fig. 3.5 might help to separate the theory from assignments. Although I use the term window, I do not mean a full window in this case. This window would contain only control buttons and an assignment with instructions. Its placement in I3T could stay the same, change depending on what is the assignment's goal, or be adjustable by the user. The regular tutorial window may be used for a deeper explanation of theory.

Control of I3T. Sending commands to I3T for things like changing the scene could be practical. A tutorial could start with a base scene instead of an empty scene every time. Furthermore, these commands would not have to be limited to the start of a tutorial. An example including command buttons can be seen in fig. 3.6. Support for such commands would need to be established in I3T.

Error and completion detection. This feature is one of the hardest to implement, but could have a substantial impact on the tutorials. Zadina's tests have shown that users would like to be informed about their mistakes, for example, through the display of an additional hint or highlighting the wrong value in red. He recommends this feature to be implemented in I3T, at the very least for beginner tutorials. There is also an option for using this feature for checking successfully completed assignments and then indicating that to the user as well.

The problem I saw with this feature was that the definition of tutorials would become much harder. Tutorial creators would need to specify the expected results in the scene. Moreover, the same results can often be achieved in multiple ways in I3T. Creators would either need to be strict on how the user should proceed or capture all possibilities. If a user completed

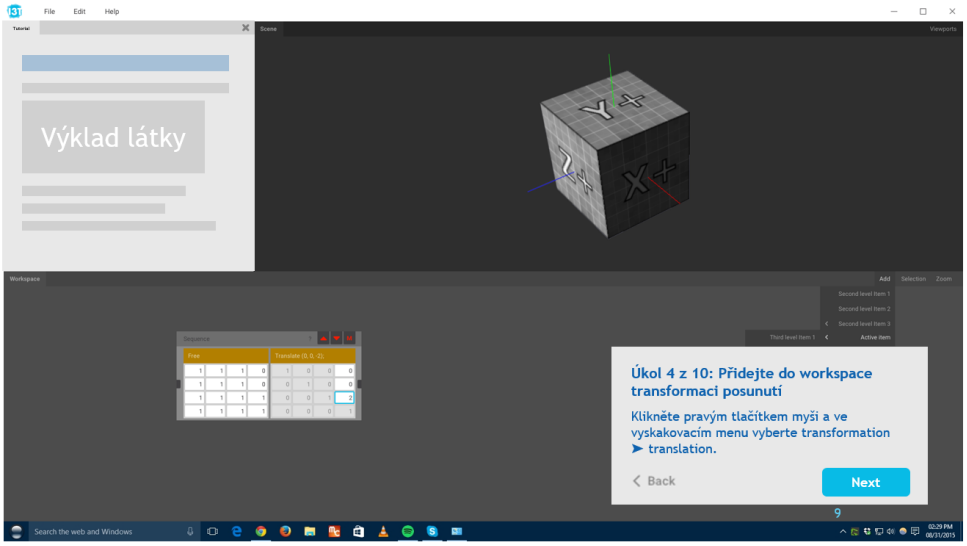

Figure 3.5: Demonstration of a floating window with assignments

a task, but would not get an expected confirmation, then this feature could cause frustration.

On the other hand, a well-done implementation of this feature might make the tutorials more responsive and engaging. It can also allow statistical analyses or other follow-up features.

Structure of step contents

At the point of creation of the presentation, I was still uncertain about the final form of the steps. I found their structure in Pilka's demo tutorial quite satisfying, although I discovered a few problems with it. Nevertheless, I still based the designs in my presentation on the structure of his tutorial, rather than Zadina's tutorial. I will follow with a reasoning for that, and continue with problems mentioned in the presentation.

Tutorial title and description repetition. Zadina did not include step titles nor step explanations and instead displayed the tutorial's title and description in each step. Therefore, all step's explaining had to be done directly in the assignment field. This might be good for simple task-based tutorials, but I believe many tutorials would not benefit from this form. Some topics in 3D transformations might need a more in-depth explanation, which would bloat the assignment with nonassignment text. Image explanations for the steps are also not very suitable in this case, since a major part of the window would already be probably taken by the title and description.

I thus preferred not to repeat both the tutorial's title and description in each step at this point of my work and include a task's title and explanation instead. Zadina mentioned in his work that doing this does indeed brake the rule of minimal design. Nevertheless, he included it in his design for testing, since it had not bothered the users in previous tests.

Figure 3.6: Demonstration of command buttons that change the scene

I could see a reason when it could have a positive effect, and that would be keeping the user in the broader context of the matter all time. For this reason, I suggested displaying the tutorial's title in the header of the window. This way, it would not take any space in the contents of the window, but would still there for reference.

Multiple assignments problem. The demo tutorial from Pilka includes steps which achieved a goal through multiple, smaller tasks. I believe this can sometimes be good in principle, but there has to be a way to define and display that. I considered three possibilities:

1. All assignments will be listed in a single step
2. This will not be allowed, and creators will either need to split the step or merge the assignments
3. There will be a special system of subassignments, where a single explanation is kept on screen while subassignments advance

I ruled out the third option, since it makes the structure more complicated and a similar behaviour can be achieved with the second option. The creator could make more copies of a single step, while changing only the assignments on these copies. The other two were debatable. The second option can be restricting, while it can also be a motivator for creators to create short steps and do not overwhelm the user.

General flexibility. Pilka's tutorials also include steps which are missing an explanation, an assignment, or instructions. I saw that in some cases, this might be desirable. I suggested to allow the ommition of these elements.

Furthermore, I proposed a variant where no firm structure was set. The contents of the step could be freely arranged to form a list of elements. Elements could be one of the following types: title, explanation, image,

assignment, instructions, or any other possible feature. This variant seemed promising for development, since it allowed changes to the step's structure without necessarily changing the code.

3.2.2 Discussion following the early design

I had discussed the designs from the presentation with my supervisor and other members of our team, especially Šárka Švábová – a student of human-computer interaction masters. They gave me valuable insights. For example, the home button in such a place could attract too much attention. I then thought about different solutions for allowing the return to the welcome window at any time, but did not come up with a satisfying one at that time.

One of the main resolutions after that presentation was choosing the flexible variant of the step structure. This way, it would be easier to try various features and it would be less restricting on the creators.

In general, restrictions on the form of tutorials are debatable. On one hand, they can create a more uniform experience from I3T. Users could get used to a specific workflow and become more effective. On the other hand, they depend on the ability of the software designer to design a well-usable form. In the end, I came to the opinion that if I3T should restrict the creator in some way, then it should do it only for very well thought-out reasons. At that time, I was not sure enough about the structure and features, so I chose not to restrict.

3.2.3 Changing the step title to tutorial title

During my own testing of tutorial creation, I noticed that the titles of steps often seem redundant. A step should symbolize a small bit of information or a small action for the user to take to advance in the tutorial, which does not often need an additional title. Since the title changes for every step, it is an additional information for the user to process each step and it makes the steps feel as distinct subtopics rather than a continuous flow of actions aiming for a goal. For comparison, steps in Unreal Editor do not contain a title.

I created a demo tutorial which did not include step titles and instead included only the tutorial's title in the first step. This reduced the presented information in the steps, although it made them feel slightly "hollow" and missing an information with what to connect the newly gained knowledge.

As an experiment, I tried reconsidering the idea from Zadina's design and repeated the tutorial's title in each step instead (this time without the inclusion of tutorial's description). This does not add any new information each step, but it does keep the user in context of the whole tutorial for the whole time as mentioned before. The version with the title felt slighlty more satisfying than the one without the title, judging both by visual and informational point of view.

In conclusion, I chose the version with the tutorial title in each step. Consequently, I suggest creators not to create the tutorials too long, so that

the title does not become too distant from the users actions. This brings them closer to GamiCAD's levels [7] — smaller, clear goals might be preferable. If a better format is later found, then it should not be a problem to change the implementation accordingly.

3.2.4 Resulting looks and features

In this section, I will describe the tutorial window in the actual final form in my implementation. The window is placed at the left top corner in I3T by default. Its screenshot can be seen in fig. 3.7

Figure 3.7: Tutorial window in my implementation

Tutorial Structure

The title of the whole tutorial is located at the top part of the window. In the middle stands the tutorial's content and at the bottom the control panel, as proposed before.

Supported features in contents

- tasks (with special formatting)
- expandable hints
- images
- unordered lists

- ordered lists
- emphases
- links

Tasks. Tasks are currently rendered as a bold black text with a blue square in front. This way, it makes them distinct from the title, while keeping them readable. They can contain text including images, emphasis, links or

Hints. Hints are hidden at first. The user needs to click the blue bold "Hint" text to expand it.

Control panel

Progress bar. The progress bar is thicker than proposed and would need a further tweaking in order to achieve that. It displays a step counter in it.

Buttons. There are two buttons in the control panel. The "Next" button is always visible and changes to "Finish" once the end of the tutorial is reached. This brings the user back to the selection of tutorials. The "Back" button appears after moving forwards. It is smaller and subtle, since it should be needed rarely. The 'X' button closes the tutorial window. It is currently possible to continue the tutorial after reopening the window. An option to return to the start window at any time is missing.

Commands

I have added the support for running scripting commands at the start of each step. This means that the creator of a tutorial can load a scene, add new or remove specific blocks from the scene and many more. This can be very helpful and used with a great potential.

3.3 Tutorial creation and demo tutorials

The presentation of the tutorials has to go hand in hand with what their creators can define in the tutorial files. I had to decide whether to use available libraries or a custom format for storing the content of tutorials. Besides that, I created test and demo tutorials on which I tested the formats.

3.3.1 Using HTML

One possibility for the tutorial window was to use a webview. That would allow the creation of its contents in HTML. This opens a lot of possibilities, but also creates problems.

The first problem is the styling and the question whether the styling should be fully customizable or be done by I3T and how would that be achieved.

The main problem lies probably in the integration with I3T. If interactive elements were to be added, there would need to be a way for communication between the webview and I3T and this would add up an unnecessary complexity.

Therefore, I decided not to use a webview and rather define the tutorials in another format.

3.3.2 Using JSON or YAML

Both of these are human-readable languages for data objects serialization. They are made of attribute-value pairs and arrays. An example of an attribute can be "tutorial title" while the corresponding value could be "Introduction to I3T". The arrays can then be used for a list of steps, where each step is again an object including attribute-value pairs or arrays. An array of elements like "header", "text", or "image" can be used to form the tutorial from top to bottom.

I first looked at the JSON format and experimented with a sample tutorial file. This file can be found at `/desgins/early tutorials/test.json`. It seemed promising, although the frequent use of curly brackets made it less "human-friendly".

After that, I discovered YAML. This format does not use brackets and instead works with indentation. It is much cleaner when looked at and has less additional marks to type. For this reason, I decided to try this format. A first example is located at `/desgins/early tutorials/test.yaml` and a more refined version at `/desgins/early tutorials/test_quite_simple.yaml`.

YAML's problem are linebreaks. If multiline-text is desired, special pipeline character has to be inserted after an attribute. Another problem is that a large indentation is needed to achieve the nested structure of a tutorial.

3.3.3 Adding Markdown

Later, I tried combining YAML and Markdown. Markdown is a lightweight markup language for creating formatted text using plain-text. Available features depend on its implementation. In my case, I used ImGui Markdown [9]. This implementation allows three levels of headers, indentation, unordered lists, links, images and horizontal rules. Emphasis was also added in its newer version this year.

My solution was to define information about the tutorial in a YAML header and then continue with the contents of tutorials in Markdown in a file. I named the extension of such files `.tut`. This showed to be relatively satisfying, although custom signs which marked the ends of individual steps were needed. At first, I used headers, such as `# Adding a cube`, for it. Each header was used as the step's title this way. This meant each step had to have a title.

As mentioned in section 3.2.3, I later removed the need for each step to have a title. I kept the step separator to be the '#' character at the beginning of a line, but removed the need for a title to follow. This way, the first step can have a title while the others not.

3.3.4 Tasks, hints and other features

The need for tasks and hints was solved using custom marks '>' and '?' respectively. I made a distiction between two types of features and their marking — those who are marked at the start of a line (such as tasks and hints) and those anywhere in a text block (direct text formatting using Markdown) For reference of suggested or used marks in the definition at this point, see the `/desgins/Featury obsahu tutorialu v I3T.xlsx` sheet.

With this setup, I created a larger demo tutorial `/desgins/early tutorials/zaklady transformaci.tut` that taught a specific topic of .

3.3.5 Reducing Markdown

From the experience I gained, I realized that combing marks of logical elements with a fixed formatting with Markdown's direct formatting marks creates a slight unwanted inconsistency. This is meant both in terms of definition and the form of presentation. I reduced the features of Markdown that I used only to images, links, lists and later emphasis. Using a more uniform custom marking style could be a better solution, although implementation of it got already out of my time scope.

3.3.6 Resulting form

In the final version, mostly custom marks are used. These are either starting a multiline block of content, or are defined in a single line. A full demonstration of the tutorials features can be found in the final demo tutorial. An example of a line with such a mark is `task: Create a cube` or `script: append("Data/tutorials/1/degreesToRadians.c");`.

3.3.7 Final demo tutorial

In the end, I created a single well made tutorial, rather than creating multiple ones. It uses all implemented features except links. It can be found at

3.4 Start window

I found Pilka's design of this window satisfying so I focused on achieving as similar result as I could get.

3.4.1 Name

I was previously referring to this window as to "Intro window" or "Welcome window". I have later changed it to "Start window" to better reflect the purpose of getting started in general.

I made this change after examining existing software. For example, Visual Studio 2019's window with a similar purpose is also called "Start window".

This window allows the user to select previously opened projects or start new ones using templates.

On the other hand, Blender v2.80 has a “Splash screen”. The screen also allows the user to select previously opened files or templates. I think its creators chose this name since the screen is missing some typical window features. It does not have an ‘X’ button and instead closes when the user clicks anywhere else. It is also non-resizable, rather small and part of it is transparent. Since our window is currently resizable, bigger, and offers an ‘X’ button, I decided to name it “window” rather than “screen”.

As a third example, Autocad 2019 works with tabs and thus opens a "Start tab" on startup.

3.4.2 Result

In fig. 3.8 can be seen a screenshot of my implementation of the start window. I have achieved a very similar result as Pilka designed. All of the buttons

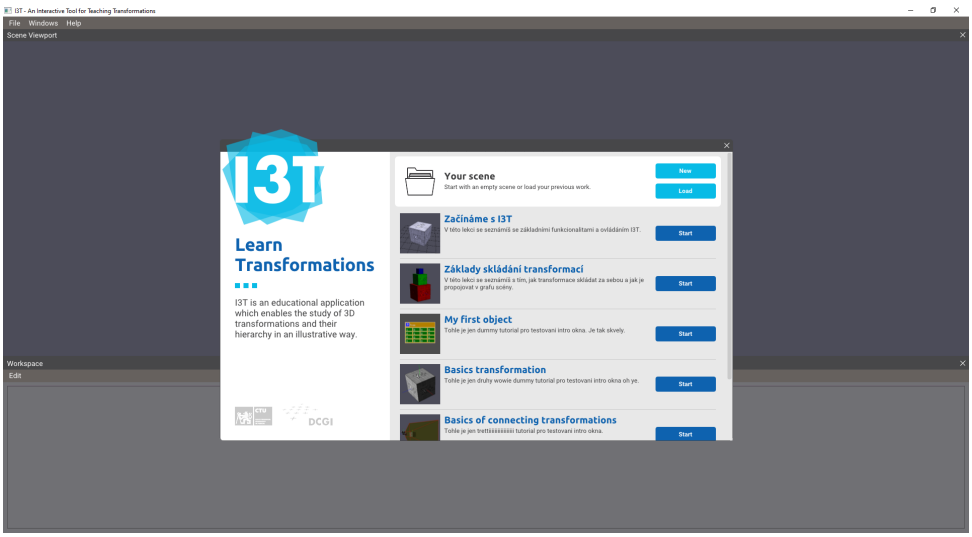

Figure 3.8: Start window in my implementation

work as expected. Starting a tutorial or opening a scene closes this window.

3.4.3 Considered further features

- tutorial rating
- tutorial filtering
- templates and last opened files using tabs
- different tutorial difficulties
- tutorial progress saving

Chapter 4

Implementation to I3T

In this chapter, I will follow up on the details underlying the implementation of the tutorial system. The whole source code can be found in appendix A in the directory `/I3T source`. During the year, we have worked with a shared repository on GitLab. Each of the members had his own branch which he regularly merged into the main development branch.

4.1 Early phase

I will describe the process of how I progressed with the implementation.

4.1.1 Using Dear ImGui and I3T's interface

As mentioned in the introduction, the Dear ImGui library [5] was set up as an underlying GUI base for the new I3T. This allowed me to create new dockable windows, which are one of the main features of the new I3T. It also allowed me to fill these windows with content using ImGui's customizable elements. The GUI is created in an immediate mode paradigm, from top to bottom by default.

Listing 4.1: ImGui GUI creation example

```
ImGui::PushFont(App::get().getFont(FONT_TITLE));
ImGui::Text("Model␣transformations");
ImGui::PopFont();
ImGui::Spacing();
if (ImGui::Button("Back", ImVec2(100, 0))) {
    if (m_current_step != 0) {
      m_current_step--;
    }
}
```

I3T was prepared with the `IWindow` interface for creating windows which I used. This interface has a `void render()` function, which is called in each frame. An implementation of this function can be used to render ImGui content, as in the example above. It also allows to close and open the window through it.

4.1.2 Tutorial window

I started with a `TutorialWindow` class implementing the `IWindow` interface. I filled its render function with a rough version of the contents shown in the example image by Pilka in fig. 1.4 using ImGui elements. This gave me a basic idea of how tutorials could be rendered in the new I3T.

4.1.3 Tutorial class

I continued by making a `Tutorial` class which could hold and display the contents of a tutorial. It contained a vector of `TStep` structs, where each element represented one step. Each step contained a vector of `TWidget` structs. Each widget corresponded to one displayed element in the tutorial step. The widgets contained two strings — one determining its type and one with the contents.

I updated the tutorial window as well, so that it allowed navigation between steps and displaying their contents. The tutorial window held the current tutorial and called the `Tutorial::render(int step)` in each render call. This function looped over all widgets in that step and rendered them based on their string type using an if-else switch. This approach was very primitive and later changed.

4.1.4 Tutorial window upgrade

Steps could be made from elements of the following types: text, H1 header, H2 header, image, task, and hint. I had relaxed the form of the steps so that I could try various combinations of these elements in a single step. All these elements were so far simple texts just in different fonts. The only exception were images, where I used existing I3T features to load and display them in a very basic way.

With this setup, I created a sample tutorial. Its file can be found in appendix A at `example tutorials/early phase/test.yml`.

4.2 Result

Here I will describe the final version of my implementation.

4.2.1 Code structure

The classes I created are the following:

Tutorial is a class for holding information about the loaded tutorials; it also contains classes and structures for parts of the tutorial and tutorial content elements

GUIImage can load images to OpenGL, hold a reference to them and properly destroy them with destroying of this class

TutorialRenderer is an interface of a class that can accept tutorial elements and render them

TutorialLoader is a class with a state machine used for creating Tutorial classes

TutorialWindow is a class that renders the tutorial window including its contents

StartWindow is a class that renders the start window

4.2.2 Tutorial loading and RAII

Tutorial loading is realized through a state machine that returns a shared pointed on a tutorial object. Tutorial objects are created in an RAII principle, since they hold references to loaded OpenGL images. These images are held in a GUIImage class, which also follow this principle. This way, an error during loading is need to be caught through an exception in constructor of these classes.

Chapter 5

Testing

In this section, I will talk about the methods I used for maintaining a user-centered design. At the basic level, I have been brainstorming and consulting the designs with the user in mind. I was also referring to the problems and suggestions mentioned in previously done testing by [1]. I then used usability testing in the later phase to check both the GUI and the way of tutoring in the tutorial itself.

5.1 Usability testing

Usability testing is the practice of evaluating the functionality and design of software by observing users' actions and behavior as they complete specific tasks. I was referring to Steve Krug [10] when designing the form of usability tests in combination with information found online.

5.1.1 Form of testing

I have searched for ways to specify the form of testing. The distinctions I used were:

1. Remote vs. in-person
2. Quantitative vs. qualitative
3. Moderated vs. unmoderated
4. Explorative vs. assessment vs. comparative

1. I have chosen to test remotely. This way, it was convenient for both sides. The user could use his own machine and setup to feel comfortable through the test. The communication setup I used was Microsoft Teams with voice, camera, screen sharing and recording on. An alternative to this was the use of Discord with the exception of recording being done on my side through OBS Studio software. One user was tested in person, but that was only because he was already at the same place as me by the time.

2. I used the qualitative approach, testing only a handful of users, while getting individual insights from them.

3. The test was moderated by me. This allowed me to properly introduce the user, offer a real-time support in unexpected situations and ask specifically tailored follow-up questions.

4. huh

5.1.2 Testing in detail

I have prepared a sample tutorial, which the user would go through on their own. The tutorial was called "Začínáme s I3T", which roughly translates to "Introtuction to I3T". The tasks taught the user about the controls of I3T and about a handful of basic blocks and connections in workspace. I also prapared an introduction to the test with a few entry questions and follow-up questions for the end of the test. The test consisted of:

1. Introduction to I3T
2. Emphasizing that any action the user takes is okay and that the user does not have to worry about making mistakes
3. Specifying the user will be testing the tutorial system
4. Mentioning the incompleteness of the software
5. Asking for screen sharing and a confirmation on being recorded
6. Outlining the test
7. Suggesting the user talk aloud their thoughts if possible during the test
8. Asking a few entry questions about the user to gather information and lowering the tension
9. Asking the user to open the software
10. Asking a simple question about the user's feeling about the intro window
11. Asking the user to start and go through the first tutorial
12. Observing the user finish the tutorial on their own, intervening only during exceptional situations (e.g. the user encountered a known bug)
13. Asking the user various follow-up questions
14. Concluding the test and thanking for participation

The file can be found on the attached CD.

5.1.3 Results of first round

I have tested three users in total, all of which were students. I will be referring to them as personas Vaclav, Francis, and Thomas.

Vaclav knew I3T only from hearsay. He is a first-year student at FEE CTU and will be taking the Programming graphics course next year. He had absolved the linear algebra course. He already had a work experience with OpenGL and shaders.

Francis did not know about I3T from before. He is a first-year student at FEE CTU on a major that does not focus on 3D graphics. He had absolved the linear algebra course.

Thomas did not know about I3T from before. He is a second-year student at The Czech University of Life Sciences Prague and had the least background knowledge surrounding the topic of 3D transformations.

Tutorial system problems

The users encountered these problems during the tutorial lecture:

- Finishing an assignment without realizing it and further wondering what to do (Vaclav)
- Skimming through the text and skipping a task without completing it or missing an important part in the text (Vaclav)
- Task 1 to figure out how to move a cube up with the existing blocks is quite hard
  - Noticing / using the hint quite late (Vaclav)
  - Using the hint straight away, but still struggled for a while (Thomas)
- Missing the “operator” word in step 8 or not realizing it is one of the options in the “Add...” menu (Vaclav and
- Not realizing that the transformation is done by the matrix (
- Taking hint only as a last resort and a sign of user’s failure to do the task as intended (Vaclav)
- Taking very long to recognize what is meant by Scene viewport and Workspace in step 1 (Vaclav, Francis)
- Not realizing the translation was a matrix until step 5 (Thomas)
- Taking a while to find manipulators (Thomas)
- Rotating the view upside down and did not realize for a while (Thomas)
- Not realizing that model block represents the cube (Thomas)
- Using Set values on translation matrix instead (Thomas)

- Having problems to choose the correct operator due to vague task description (Thomas)
- Having to read again to find manipulators (Francis)
- In step 6, the user prefers to do the task as what is on the picture, instead of reading carefully and doing it in a desiref way (Francis)
- Hint was used more as a solution, rather than hint

Other problems

The test also showed other problems with I3T the users had:

- Trying to click the "Add..." label in the "Add..." menu (Vaclav and Thomas)
- Encountering a situation where a wrong submenu would expand in the "Add..." menu when moving the mouse quickly (Vaclav and Thomas)
- Accidentally moving the workspace when trying to rotate the viewport
- Uncertainity about operators and transformations in "Add..." the menu

Other observations and follow-up questions' answers

Overall, the users were mostly positive about the test. Here are a few worthy examples:

- Vaclav mentioned that the lecture has had well informed him about everything and he complimented its step-by-step approach
- Vaclav did not like the idea of the tutorial making an overlay over I3T that would show him where to do what; Thomas and Francis were indifferent
- Vaclav hesitated about the idea of tutorial steps being automatically checked for completion
- Vaclav liked the idea of visualizing tasks (e.g., dragging a matrix into a sequence) through animated pictures
- After the test, when asked to create a new scene, add a model and rotate it by 3 radians:
- Both Vaclav and Francis tried to use a regular matrix and set a level of detail at first, but ended up using an operator instead since they did not properly remember the LOD menu and found the operator way better in the end; in Francis's case, it was due to opening of the sequence menu, instead of the transformation menu

- Thomas intuitively figured out that he could grab the tutorial window and keep it floating at a place he likes right at the start
- Thomas liked the mouse button icons
- All users had no problem with the welcome screen
- Francis would like to have a solution button, in addition to hints
- Francis would like automatic completion checking
- Francis sometimes felt not wanting to read the text, but always read it in the end
- Francis would like an overlay which shows him where to do something

5.1.4 Changes after the first round

I have fixed a lot of errors after this tutorial. The final version of the tutorial can be found on the CD

Chapter 6
Discussion

6.1 Overview of results

The results of my work can already be used for both tutorial creation and their use by students. Students were successfully able to pick and go through a selected tutorial. This demo tutorial includes less text than the previous designs and instead use images to portray actions and concepts. There is added interaction through tooltips, hints, and the possibility of using scripts throughout the tutorials. Although I have not implemented all features that I proposed, my work and designs can be used as the source for their future implementation. I have not created a tutorial in the current version that would teach a topic from 3D trasnformations themselves except for one in the previous version of my work.

6.2 Problems during my work

6.2.1 Changes in the team

Unfortunately, Šárka Švábová left the team very early during this academic year. Therefore, we had to do testing on our own when we needed one, including me.

Jaroslav Holeček's primary goal changed, and he later worked on the original goal of Sofia Šašorina instead. Although I was designing the tutorial system with a fixed order lessons, I added features in the code that support setting a specific step from a specific tutorial. I also prepared a very basic way of defining multiple choice questions or free form questions in each step. This could be later used as a base for potentially achieving his original goal.

6.2.2 Parallel development

Due to each of us having to implement a different a part on the new I3T, it was not possible for me to use features that were not yet implemented in the new version. The first demo tutorials were based solely on old I3T and

previous designs for this reason. Only with the latest tutorial I was able to work with the new I3T.

6.3 Ideas for further development

6.3.1 Tutorial definition and loading

Other different types of definitions can be further tried. A late idea that came to my mind and that I was not able to realize was to form the tutorial definitions in a Latex-like form. This means to use commands such as "Content of the task". Important to note are images, since writing "![Image name](image.png)" is not much coherent with other elements like the tasks or hints. The problematic thing with images is the definition of their size. I suggest further development in this area.

6.3.2 Image rendering

Since images currently do not take an argument for size, their actual size on screen is either adjusted to the remaining width of the current line or their original size, depending on which is smaller. I suggest adding the possibility to specify their behaviour. My proposal is having one parameter for the following. The number -1 means the image will stretch to the width of content even if its original size is smaller. The number 0 means the image will get squished if the width of content is smaller than its original size, otherwise retains its original size. Any other positive number means maximum width and will result in two possibilities. If the available width is smaller, then the image gets squished. If it is larger, then the image will stretch to the specified width.

6.3.3 Visual easthetics

I suggest colors, fonts, and element sizes to be later tweaked by a graphical designer.

6.3.4 Language

The version I worked with combined the use of Czech and English. This should not definitely be the case for the final product. As discussed with my supervisor, an option for the user to choose his preferred language should be added. Therefore, tutorials listed in the start menu could be filtered based on the language.

6.3.5 Tutorial features

Although I have discussed various possible features in chapter 3, I will list few of them here.

Completion and error checking

I was not able to realize this feature in my available time, although I believe this would change the tutorials in a significant way. Zadina mentions this feature should be added on page 55 in his work. I recommend to further consider an implementation of such feature

Progress information

I have not changed the start window, although

Task highlighting

This is another feature previously suggested by Zadina. I agree with th

Animations, effects and sounds

All these features might make I3T more enjoyable when used properly and with moderation. ImGui library does not offer transitions, animations, or effects by default. Nonetheless, I believe some might definitely be possible. I have already implemented a simple animation of the progress bar. Its value gets non-linearly interpolated over a brief period of time after moving between steps. It creates a smooth subtle effect. Here are a few more examples that I was able to think of:

1. Tutorial contents might fade out and fade in when moving between steps. This could be done either by changing the transparency of the contents or by changing the transparency of an overlay with the same color as background.

2. The welcome window might stretch out at startup with an animation of its size.

3. Tutorials listed in the welcome window might get highlighted with a fade-in fade-out background effect when hovered. I have already tried to display a tooltip when a tutorial was hovered and this trigger is currently commented in the code.

4. A simpler version of the previous suggestion might be used for any button.

6.4 Tutorials in a form of a game

Other than individual features, I would like to talk about the possibility of making a game system in I3T. I have already discussed a few approaches that could gamify the experience in chapter 1. Here, I will propose ideas for a gamified tutorial system.

6.4.1 About games

First, we need to specify what is characteristic to games. I will use my notes from the B4B39HRY course at FEE CTU to form the distiction. Games usually:

1. contain conflicts or puzzles
2. have fixed rules
3. let the player solve them
4. reward the user with visual feedback, scores, and more
5. allow the user to get creative

6.4.2 Gamifying I3T

The first point in the previous list is already realized in I3T. Tutorials can contain assignments for the user to solve.

A problem with them is that there is no feedback to the user. Therefore, checking of assignment completion would probably need to be implemented. With it, I3T could give a visual feedback on completion.

Following this feature, various levels of gamification could be achieved. Ideas that could gamify I3T:

I3T mascot. Adding an original character to I3T could make a more personal connection to the user. Tutorials could instead be presented in text bubbles told by this character. This character could "stand" on top of I3T.

An example of a character providing guidance is Microsoft Office's discontinued assistant called Clippit. Although it might not have been perfect, it certainly made a more memorable experience. Another example could be Navi — a fairy from the video game The Legend of Zelda: Ocarina of Time. This fairy helps the player learn the controls and advance in the game.

Including a story. Adding a story to the tutorials, such as GamiCAD's space mission, might also add user engagement.

Chapter 7

Conclusion

The primary goal of this work was to get familiar with the I3T tool and previous works on it, research methods used in tutorials, propose and implement a tutorial system including a welcome screen and a tutorial window. The system should also allow the creation of new tutorials. With a few exceptions, I have realized my proposed goals in this work and implemented a working tutorial system in I3T. I have also included discussion and realized researches that might further help with future development of I3T and its tutorial system.

Bibliography

[1] Michal Folta. "Teaching of Transformations". Master's thesis. FEL ČVUT, 2016. URL: http://hdl.handle.net/10467/64836.

[2] Lukáš Pilka. *Grafické návrhy rozhraní pro nástroj I3T*. Czech. Internal communication. 2018.

[3] Vít Zadina. "Testování užitečnosti nástroje pro výuku transformací". Czech. Bachelor's thesis. FIT ČVUT, 2019. URL: http://hdl.handle.net/10467/83400.

[4] Miroslav Müller. *Windowing system for the I3T Tool*. Semestral project. 2020.

[5] Omar Ocornut. *Dear ImGui*. 2021. URL: https://github.com/ocornut/imgui (visited on 07/13/2021).

[6] Rui Paiva, Milton Ferreira, and Miguel Frade. "Intelligent tutorial system based on personalized system of instruction to teach or remind mathematical concepts". In: *Journal of Computer Assisted Learning* 33 (Mar. 2017), pp. 370–381. DOI: 10.1111/jcal.12186.

[7] Wei Li, Tovi Grossman, and George Fitzmaurice. "GamiCAD: A Gamified Tutorial System for First Time Autocad Users". In: *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*. UIST '12. Cambridge, Massachusetts, USA: Association for Computing Machinery, 2012, pp. 103–112. ISBN: 9781450315807. DOI: 10.1145/2380116.2380131. URL: https://doi.org/10.1145/2380116.2380131.

[8] Extra Credits. *Tutorials 101 - How to Design a Good Game Tutorial*. Youtube. 2012. URL: https://youtu.be/BCPcn-Q5nKE.

[9] Juliettef. *ImGui Markdown: Markdown for dear imgui*. 2021. URL: https://github.com/juliettef/imgui_markdown (visited on 07/13/2021).

[10] Steve Krug. *Web design: Nenuťte uživatele přemýšlet!* Czech. Ed. by Jan Škvařil. Computer Press (CP Books), 2006, pp. 113–136. ISBN: 80-251-1291-8.

Appendix A

CD contents

This is the folder structure of the attached CD.

- `/`
  - `designs` ..... documents and images I have created
    - `images` ..... images that I have created for I3T
    - `early tutorials` ..... not used versions of tutorials I created
    - `demo tutorial` ..... the final version of my demo tutorial
    - `demo tutorial screenshots` ..... images of the demo tutorial
  - `I3T release` ..... an executable version of I3T including my work
  - `I3T source` ..... source code of I3T including my work
    - `Data`
      - `tutorials` ..... folder for storing tutorials
    - `Tutorial` ..... classes that I created for tutorials
    - `GUI`
      - `Elements`
        - `Windows` .... classes for the start and tutorial window that I created
      - `Theme.cpp and .h` .... class that I edtied to style the tutorials
  - `usability testing` .. demo tutorial versions from testing with users
  - `tutorial concepts Felkel and Pilka` ... documents with tutorials that were available to me
  - `thesis` ..... the LaTeX source code of this paper