BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

PRESENTATIONS MAKING BLENDER ADD-ON
BLENDER ADD-ON PRE TVORBU PREZENTÁCIÍ

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR — RONALD TELMANIK
AUTOR PRÁCE

SUPERVISOR — Ing. TOMÁŠ CHLUBNA
VEDOUCÍ PRÁCE

BRNO 2021

Brno University of Technology
Faculty of Information Technology

Department of Computer Graphics and Multimedia (DCGM) Academic year 2020/2021

Bachelor's Thesis Specification

24081

Student:	**Telmanik Ronald**
Programme:	Information Technology
Title:	**Presentations Making Blender add-on**
Category:	Computer Graphics

Assignment:

1. Learn how to work with the 3D modelling program Blender (version 2.8 and higher)
2. Go through materials about scripting Python API in Blender
3. Look for already existing or conceptual solutions
4. Design and propose the add-on that extends Blender, providing an easy way to create presentation slides
5. Implement the proposed add-on and demonstrate its usage with various types of templates
6. Write a documentation and publish the add-on in the community
7. Create a video describing your work

Recommended literature:

- Blender Python API Official Documentation

Requirements for the first semester:

- Items 1 to 4, experiments leading to item 5

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Chlubna Tomáš, Ing.**
Head of Department:	Černocký Jan, doc. Dr. Ing.
Beginning of work:	November 1, 2020
Submission deadline:	May 12, 2021
Approval date:	November 10, 2020

Abstract

The goal of this thesis is to create a tool that allows Blender users to create slide-based presentations. This process can be time-consuming caused by the complexity of Blender. Hence, to eliminate this issue an add-on is proposed. This add-on implements a set of tools to support the creation and later rendering the presentation in real-time. The final source code for the add-on is publicly available on GitHub.

Abstrakt

Cieľom tejto preace je vytvoriť nástroj, ktorý umožní používateľom program Blender vytvoriť prezentáciu vo forme snímkov. Tento proces s je zvyčajne časovo náročný spôsobené najme komplexnosťou program Blender. Preto na uľahčenie tohto problém je vytvorené rozšírenie pre program Blender. Toto rozšírenie implementuje sadu nástrojov, ktoré uľahčia tvorbu a neskôr aj samotnú prezentáciu v reálnom čase. Finálny zdrojový kód rozšírenia je voľne dostupný na platforme GitHub.

Keywords

Blender, Add-on, Computer graphics, Slides, Presentation, Real-time, Python

Klíčová slova

Blender, Rozšírenie, Počítačová grafika, Snímky, Prezentácia, Reálny čas, Python

Reference

TELMANIK, Ronald. *Presentations Making Blender add-on.* Brno, 2021. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Tomáš Chlubna

Presentations Making Blender add-on

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Ing. Tomáš Chlubna. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

. . . .
Ronald Telmanik
May 10, 2021

Acknowledgements

I would like to thank my supervisor Ing. Tomáš Chlubna, for support and guidance throughout the thesis.

Contents

Chapter 1

Introduction

Presentations are increasingly getting popular as tools to give information. These are then used in various areas ranging from schools to businesses to inform, educate and motivate.

Combining Blender with slides might seem like a strange combination, but Blender can show images, videos, and text like any other presentation software. Unlike any other slide show software, Blender does 3D graphics well, and it definitely can create complex animations. Most importantly, Blender is free. However, all of this comes at a cost. Blender's learning curve is steep, and the lack of tools makes this process time-consuming.

This project aims to implement an add-on, that will help users create and later present the presentation. This add-on is integrated into open-source software Blender. It supports tools to help with designing slides and real-time fullscreen rendering. These tools include various tools for creating text objects, designing slides through templates, and finally, an option to export the presentation.

Chapter 2 describes tools used during development, introduces Blender, how it works, and its API. Existing solutions are presented in chapter 3. Not only does it shows solutions from traditional presentation software but also web applications for experienced users. Chapter 4 proposed design of add-on. Lastly, chapter 5 shows some interesting implementation details and how were some issues solved.

Chapter 2

Theory

This chapter aims to introduce all the basics that are used throughout this thesis. It describes tools used for development, basics of Blender and short examples of Blender Python API.

2.1 Presentation

Presentation software is a piece of software designed to showcase a piece of information to a group of people. This kind of software also provides tools to make the process of creating such presentations easier. It is made of two parts:

- Environment to create and edit text, and insert graphics.
- Tools to display the actual slide show.

These software help user generate a presentation or a slideshow. Presentations contain various elements including text, images or videos. Final presentation represents a series of images that are shown in a sequence. These images can be switched automatically or manually by a presenter.

The main reason of using a presentation is to save presenter from describing details about subject while it can be said by a single image. Presentations are used in various field from business world to instructional purposes. Some types of presentation are used for just for artistic purposes as a screensaver.

2.2 Development tools

This sub-chapter introduces Python, a programming language used for development. It also describes PEP8 and the version control system. A short introduction to Visual Studio Code is given.

Python

Python[1] is a general-purpose, interpreted, dynamically typed and object-oriented programming language[2]. It supports multiple programming paradigms such as procedural and

[1]Python: https://www.python.org/
[2]What is Python?: https://www.python.org/doc/essays/blurb/

functional[3]. Python excels in its syntax, it is clear and resembles written English like none other language does. Python syntax comes close to written English, it is simple and clear. Even though negatively perceived by its performance, it is now a strong alternative to any other programming language. Today Python is used in various fields ranging from small scripts, GUIs[4], numeric and scientific programming to robotics. The standard implementation of Python runs on all big platforms counting Linux and Unix systems and Microsoft Windows. Listing 2.1 show a Python sample code, that prints names of friends with indexes.

```
friends = ['John', 'Pat', 'Gary', 'Michael']
for idx, name in enumerate(friends):
    print (f'Position {idx} is {name}!')
```

Listing 2.1: Sample Python code

Python was not chosen by mistake nor preference, it is the tight integration with Blender. It is used for writing add-on, simple scripts, creating panels and menus for user interface[5]. All of the features will be explained later in the documentation.

PEP8

Python Enhancement Proposal #8[6], otherwise known as PEP8, is a style guide for Python code. Python does not force you to write in a certain style if it is syntactically correct, but following PEP8 recommendations, it makes the final code more readable. PEP8 describes various aspects of code including naming conventions, blank lines, maximum line length, or even comments. Even though these rules seem pedantic, following them can and will improve code quality, especially when it comes to sharing the code with somebody else. Code in this project was written with these rules in mind and to make sure it complies with PEP8, the `black`[7], PEP 8 compliant opinionated formatter is used.

Fake bpy

The bpy acronym stands for `blender python`, which is a module, that Blender provides to programmers.

Programming Blender add-on in Blenders own text editors is a viable option, but not for a long. This text editor is just missing more advanced features that are standard in any other source code editor. But leaving Blenders text editor comes with one downside, that is text auto-completion. Luckily, thanks to the Pythons community, someone already created a Python package[8], that brings Blender API to the editor of your choice.

```
>>> bpy.context.object.name
'Cube'
>>> bpy.context.scene.objects["Torus"].data.vertices[0].co.x
'1.0'
```

Listing 2.2: Accessing attributes with bpy

[3]Python FAQ: https://docs.python.org/3/faq/general

[4]GUI - Graphical User Interface

[5]Blender development: https://www.blender.org/get-involved/developers/

[6]PEP8: https://www.python.org/dev/peps/pep-0008/

[7]black: https://pypi.org/project/black/

[8]fake-bpy-module: https://pypi.org/project/fake-bpy-module-2.91/

Listing 2.2 showcases the usefulness of having such help in more complicated or unexplored `bpy` modules.

Visual Studio Code

Visual Studio Code is a lightweight source code editor with endless customization options. It does not come fully loaded with all the tools, but thanks to its easy customization everyone can make it fit their needs. VS Code contains dozens of extensions to help with any task. Featuring clean user interface, support for dozens of languages and many keyboard shortcuts.

Creating a Blender add-on outside of Blender application can be an unpleasant experience. There is a solution to this and that is VS Code extension for Blender development[9]. This extension helps managing add-on development outside Blender. It provides tools to create new add-on with template, start Blender with add-on enable or hot reload on code change. It also implements a nice way of registering add-on but more on this later.

Git and GitHub

`Git`[10] is a version control system, that locally stores file copies at various stages of project development while `GitHub`[11] is a Git hosting service that adds a bunch of features like bug tracking, continuous integration, and wikis.

Both technologies are used throughout this project development. There is the whole history of the project, release version, and wiki page with description and manual.

2.3 Blender

Blender is the free and open-source 3D creation software. It supports many different areas in computer graphics ranging from modeling, rigging, animations to rendering[12]. In other words, Blender enables users to create characters, props, or environments. Users can set them to life and use a final object in a video game or use it as footage in the video.

Blender is a cross-platform solution available to download on Linux, macOS, as well as Windows systems. Portable version is also available to download. Blenders front-end is consistent throughout platforms thanks to implementation using OpenGL. Users can customize how Blender behaves through various settings built-in or extend existing with add-ons.

Blender is somehow in a complex open-source software. It comes with great power and is free to use, while many other alternatives are much more expensive. It is not only free price-wise, but the source code is available for anyone interested.

Usually, 3D graphics software has a steep learning curve, while Blender is not an exception. This section will introduce the key concepts that will be used throughout this project, and mainly so that anyone can build their presentation in Blender after reading this, with no prior knowledge. Most of this information is taken from official Blenders documentation manual[2] and API[3], Blender for dummies[6] and Blender Python API[4].

[9]Blender Development: https://github.com/JacquesLucke/blender_vscode
[10]Git: https://git-scm.com/
[11]GitHub: https://github.com/
[12]Blender: https://www.blender.org/

User Interface

The default User Interface (UI) looks the same throughout platforms thanks to OpenGL, though it can be easily customized and seem different. The Blender window contains multiple areas, and these areas are populated by editors. Areas are resizable rectangular boxes. A window can have one or multiple of these areas. Areas can be resized by dragging their side, and joined or split by pulling a corner. These areas hold screen space for an editor and each area can hold only one of these editors. The editor is a way to access and modify data by the user. Blender has multiple types of editors, just to name a few:

- 3D Viewport provides a preview of a project.
- Timeline is used for manipulating keyframes and changing the time.
- Python console executes commands in Python through Blender Python API.
- Outliner lists all types of data stored inside `.blend` file.

Many editors are divided into tabs and panels. Tabs create overlapping sections. Each tab usually contains multiple panels. These panels can show or hide their contents by either expanding or collapsing. Panel position is modified by dragging it to the desired location. The default Blender User Interface can be seen in figure 2.1.

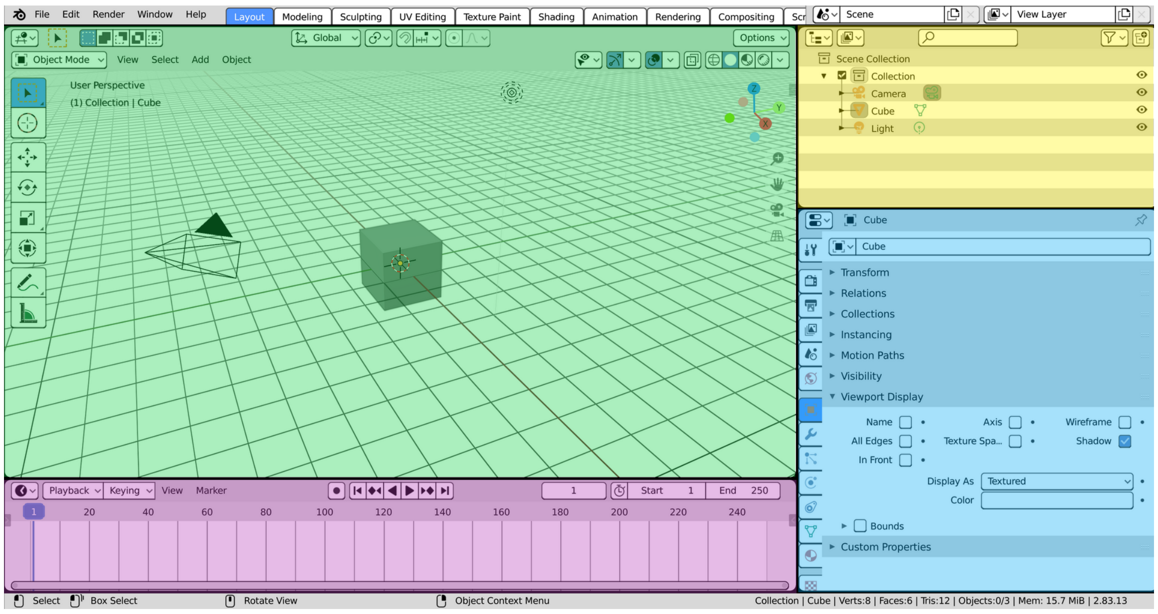


Figure 2.1: Default Blender User Interface
Highlighted areas with editors. Print friendly theme applied. 3D Viewport (green), Outliner (yellow), Timeline (purple) and Properties (blue). Tabs and Panels are visible in blue area.

Blender uses workspaces to store a layout of the window. Workspace defines the location of areas and types of editors used inside a window. The blender comes with already predefined workspaces such as modeling, sculpting, and scripting. Users are also able to create custom workspaces for a specific task.

One of the least used but quite important for this project is the scripting workspace. This workspace provides editors preset to get the most out of scripting right in Blender.

- Python Console is a console that provides access to internal Blender data through its Python API. It has code autocomplete and history executed commands.
- Text Editor for a longer sequence of commands, or just any text in that case. It has a built-in line number, searching features, and can directly run the whole script. The text editor has a template fold with examples of some features.
- Info logs information history about executed commands not just from the user but also from Blender.

.blend file

Data-block is a base unit for any Blender project. Some types of data-blocks include cameras, meshes, materials, and the world. Blender uses many more and those can be found in Outliner set to Blend file. These data-blocks represent an abstraction of what the user works with.

Every `.blend` file would eventually get cluttered and raise in size. Therefore every data-block keeps a number reference count, also known as users. Listing 2.3 show example of accessing user count via `bpy`. This number can also be seen in Blender's UI.

```
>>> bpy.data.materials['Material'].users
3
>>> bpy.data.materials['Material.001'].users
0
```

Listing 2.3: Example of users count on material data-block.

This number is checked when the user saves a file. During this process Blender will check every data-block if his number is higher than 0, if so, the data-block is saved, otherwise will not be saved and eventually removed. In special cases, there is a removal protection in form of a shield icon that will save the data-block no matter the user count.

Scenes

Scenes are the top most organizational unit in Blender. Each `.blend` file can contain multiple scenes. Blender allows creating an empty, linked copy or full copy from the currently selected scene.

Every `.blend` file has a list of scenes. All scenes in a `.blend` file are stored in a list. This list is sorted by scene name. Changing a scene name may result in a change of scene order.

Collections

Blender uses two types of collections, every scene has its scene collection that holds all the objects linked to a particular scene. The second type are collections created by the user, these are typically used to organize objects that relate in some way.

Each scene has its `scene collection`, this collection stores all the objects and even user-defined collections inside that scene.

User-defined collection can be initialed or simply removed. These collections can be nested inside other collections creating a hierarchy of collections. Each collection carries a unique name and to distinguish them even more a color tag is available.

Objects

Scenes contain multiple objects. Blender has many objects meshes like cubes, spheres, and toruses, lights, cameras, and even empty objects. Each of these objects has two parts:

- Object - holds information about the position, rotation, and size of a particular element.
- Object data - holds everything else, for example in meshes it is geometry, materials lists, etc.

All objects are stored inside a `.blend` file and from there they can be linked to as many scenes as one wishes. New objects are created inside the selected scene, within the selected collection, and at the location of the 3D cursor.

Mesh Object

A common object type used in a 3D scene is a mesh. The Blender comes with a number of these mesh objects that can be used right away. Some of them are plane, cube, UV and icosphere, torus, and most importantly a monkey. New, more complex mesh objects are created by editing and/or joining existing ones. Blender has modes with specific tools, that will help with mesh modeling:

- Object Mode - works with whole objects, creation, changing scale, location, and rotation.
- Edit Mode - modifies the mesh by editing vertices, edges, and faces.
- Sculpt Mode - modifies the mesh by using a brush.

Each of these modes can be found inside 3D Viewport. Users can switch between these modes when needed. Each of these modes comes with a set of tools specific to the mode.

Text Object

Another type of object is a `Text` object. This objects just show text value in 3D. Blender maps letter codes to geometry representing them in the 3D Viewport. When newly created, text objects are flat 2D objects. But text objects share the same type as curvers and therefore can be modified in such way as curves do. By just applying some modifiers, they can turn into a 3D text. In this case object mode takes care of object properties and text properties, that is text spacing, size, font used, various alignments and text boxes. Object mode can also convert text object into a mesh or curve. After entering an edit mode white cursor appears, indicating current location, this allows user to change text value, just by typing it in. Blender is not limited only to its own fonts, and it support various external fonts formats like `OpenType` and `TrueType`.

Materials

All objects created in Blender by default appear in gray. To modify how objects appear in the final output Blender uses materials and textures. The material describes the appearance of the objects. It defines what color it is, how light interacts with it, or how shiny or dull it is, thus resulting in an object that looks like plastic, glass, metal, etc. Changing viewport shading into `Material Preview` or `Rendered` shows how the object would look in the final output. The simplest method to change object color is to set its diffuse color. This is the color object that will diffuse when light hits it. Setting this color is done via the color wheel built into Blender. The more complex approach to changing object appearance is to use shaders. Shaders describe how light interacts with the surface, rather than the color of the surface.

Renderer

Renderer or render engine, is a tool, that converts the 3D data from the scene into a 2D image or series of images for animation. The final output is defined by the camera, lights, materials, but also render size, aspect ratio, and file type.

Blender comes with three different render engines:

- Eevee - physically based real time renderer.
- Cycles - physically based path tracer.
- Workbench - fast rendering during modelling and animation preview.

There are more render engines available in form of add-ons.

While Cycles will always result in better renders, it is the interactivity in 3D Viewport that makes Eevee so appealing. Because Eevee uses a process called rasterization, the speed is blazing fast compared to other renderers. The compromise of quality and speed is just perfect for real-time presentations.

The renderer uses a camera object to decide what is visible in the rendered image. Cameras are objects like any other, with properties for adjusting various properties such as lens type, depth of field, and aperture. Blender has a tool that shows a camera view, which can be used as a preview.

In some cases, the rendered image may look fine without any light in the scene altogether. For more complex renders adding some sort of light source can change everything. Lights are objects that shine light and therefore make scenes brighter. Blender provides few options to choose from such as sun, spot, or point. Light objects have various properties mainly the strength and color of the light. These light objects are like normal objects and can be placed wherever needed.

2.4 Blender Python API

Some tasks may require a sequence of commands, which may be time-consuming. Some things are just straight missing in Blender. Blender Python API enables users to create scripts that automatize these tasks or extend the functionality of Blender. Everything that can be done through Blender's user interface, can also be done via API. But there is more, things like running tools with custom properties, creating new tools, or defining new settings in existing Blenders data.

The bpy module

This `bpy` module, when imported it gives access to Blender data, classes, and functions. The bpy module contains multiple application modules, such as:

- Context Access (bpy.context) - access to the current data, e.g., selected objects, current area, and window.
- Data Access (bpy.data) - Blender's internal data, e.g., scenes, objects, and brushes.
- Operators (bpy.ops) - functions for manipulating Blender's data.
- Types (bpy.types) - Python class objects representing Blender's own types.
- Property Definitions (bpy.props) - properties assigned to classes.

These are some of the most used application modules provided by API. Blender also comes with some standalone modules. Most notably the `Math Types & Utilities` (mathutils), which provides access to various math operations, such as `Euler`, `Matrix` and `Vector`. `OpenGL Wrapper` (bgl), which contains OpenGL constants and functions that can be used from within Python in Blender. More of the various modules available in Blender can be found in documentation[4].

Add-on

Add-ons provide additional features and abilities to Blender. These add-ons are then listed inside Blender's add-on list. This list contains all add-ons in Blender, short description, and toggle to enable or disable the add-on. They are very similar to basic scripts, but there are minor differences To be able to show this information, the `__init__.py` file must contain a `bl_info` dictionary. This dictionary contains entries for information such as author name, Blender version, and category. Blender uses this data to populate various metadata related to the add-on. Apart from this dictionary, two more functions are required, `register` and `unregister`, called whenever this add-on is enable or disabled respectively. Listing 2.4 shows minimal structure containing only the necessity to make add-on pass Blender's requirements.

```
# add-on info
bl_info = {
    "name": "Blender Add-on Example",
    "blender": (2, 83, 0),
    "category": "Object",
}
def register():
    # executed when enabling the add-on

def unregister():
    # executed when disabling the add-on
```

Listing 2.4: Minimal structure for Blender add-on.

Add-ons typically register operators, panels and menus, properties, etc., but any script can do that. They are written more formally so that it is easier to share, transfer and use. These add-ons usually come in `.zip` format, this can be installed via Blender Add-on Interface in Properties. They have a preference menu for further customization, and lastly, they can be enabled or disabled when needed. Whereas script has to be imported inside text editor in Blender, and users have to change python code for customization.

Operators

Operators allow calling Blender functionality from within the user interface. These operators include code written in Python, C, or various macros. Every operator must be registered as an operator of class `bpy.types.Operator`, before any use. Any registered operator can be executed either by clicking the appropriate button in the user interface or executing it inside code. Blender uses `bl_idname` identifier to distinguish operators. Operator registration maps the operator execute function to the `bpy.ops` module through its identifier.

Every operator execute function must return a value. The execute function returns a set of string values, mostly `FINISHED` or `CANCELLED`.

Every execute function gets a reference to `bpy.context` as a parameter. There are instances where the operator fails during execution caused by the wrong context. To solve this Blender provides a `poll` method, which is used to avoid this problem. This method checks if all prerequisites are met and returns `True` or `False` accordingly. Sometimes the only solution is to override the context with appropriate settings.

Simple example in listing 2.5, shows basic structure of an operator. This operator is then registered under `bpy.ops.wm.example`, and after execution a text is printed in console.

```
class ExampleOperator(bpy.types.Operator):
    bl_idname = "wm.example"
    bl_label = "Example operator"

    def execute(self, context):
        print("This is just an example.")
        return {"FINISHED"}
```

Listing 2.5: Example of an operator

Operator also provides an `invoke` function. This function is used to initialize the operator before the execute function is called. It is typically used for assigning properties which are later used during execution.

Each operator may define options in `bl_options` class variable. These options include setting such as `INTERNAL`, which removes operator from search or `UNDO`, which will add support for redo panel.

If an operator has custom properties registered and supports a redo panel, every time after execution a panel will pop up. This panel contains all properties from the operator. Changing values of these properties will undo applied changes and apply new ones from this panel. The looks and order of properties in this panel is done automatically by Blender, in order of properties are assigned in an operator. Blender offers a `draw` method, which allows changing the looks of the redo panel. This method behaves the same as the `draw` function inside any user interface class.

Properties

There are two types of properties. The first type is properties associated with operators. Every operator can register its properties. These are different from standard class properties, as these can be passed as arguments to the operator. There are multiple such properties available:

- BoolProperty - True or False value.

- CollectionProperty - Collection of either `PropertyGroup` or subclass of `bpy.types`
- EnumProperty - Enumerator, contains items and their description.
- FloatProperty and IntProperty - Either Float or Int value, respectively.
- Vector - Vector of Int or Float values.
- PointerPropety - Pointer to either `PropertyGroup` or `bpy.types` object
- StringProperty - String value.

Each of these properties then defines various options. Example 2.6 defines FloatVectorProperty, with various options defined.

```
color: FloatVectorProperty(
    name="Color", # name of property
    description="Color used for text object", # short description
    subtype="COLOR", # further specifies property
    size=4, # vector size
    default=(1, 1, 1, 1), # default value of vector
    max=1.0, # minimal and maximal value of each value
    min=0.0,
)
```

Listing 2.6: Example of FloatVectorProperty registered with an operator

The second type of property is registered as a `PropertyGroup`. These properties registered inside a single class are used to avoid many individual settings registered to a single object. Typically an class inherits from `bpy.types.PropertyGroup` and registered various properties. This class is then registered to the object as a group of properties using `bpy.props.PointerProperty`.

Every property can have a callback assigned, which will be called whenever the value of the property changes. Apart from the update function, there are also a `get` and `set` functions available. These two functions are called whenever a value is read or written respectively.

User Interface

Blender allows users to extend its user interface by providing `Panel` and `Menu` both from `bpy.types` module. Every class inherits from either of these and specifies multiples options. Each such class must define `bl_space_type`, `bl_region_type`, these two specify location of panel or menu. Furthermore defines `bl_label` for name, and other options in `bl_options`. These options include various options such as `DEFAULT_CLOSED`, which will make panel closed after creating or `HIDE_HEADER`, which will hide label.

Panels like operator also define poll method, that returns True or False based on circumstances.

Each panel must define a `draw` function, that provides how will the panel look. Users are able to define properties, label and operators inside this functions. Listing 2.7 shows example of panel in Blender user interface, that will be shown in 3D View in side panel. This panel then prints example text with label.

```
class ExamplePanel(bpy.types.Panel):
    bl_space_type = "VIEW_3D"
    bl_region_type = "UI"
    bl_category = "Example"
    bl_label = "Example"

    def draw(self, context):
        context.layout.label(text="Example Label")
```
Listing 2.7: Example of basic panel in Blender 3D View's side panel

Blender offers creation of subpanels by creating a base class and subclasses of this class. All subclasses then define `bl_parent_id`.

Handlers

Blender provides various events that will trigger a callback function. Some of these events include frame change, new file loading, render complete, file save, etc. Each of these events represents a list of all callback functions, that will be executed when this event occurs.

Registering such a handler can be done just by appending the callback function to the list of that event. To remove a handler, just remove the callback functions from the list. Listing 2.8 shows register and unregister process for a handler.

Handlers are freed when loading new files. But if needed, a persistent decorator for a callback is available. This is perfect for add-ons as this will make the handler stay running across multiple files.

```
# add new handler
bpy.app.handlers.frame_change_pre.append( handler_function )
# remove handler
bpy.app.handlers.frame_change_pre.remove( handler_function )
```
Listing 2.8: Example of adding and removing a handler

Register() and Unregister()

Every add-on, must contain `register` and `unregister`. The `register` function is responsible for initializing the add-on. Mainly for registering all classes and properties. The `unregister` does the exact opposite, that is unregistering classes and deleting properties.

Any object that inherits from `bpy.types`, needs to be registered before it can be used inside Blender. Blender provides a module `bpy.utils`, which contain two functions `register_class` and `unregister_class`. These two functions register or unregisted inherited object, respectively.

Chapter 3

Existing solutions

This chapter provides an overview of what are some of the possible options when it comes to presentation software. There are multiple solutions available based on preferences. While traditional desktop applications still exist, quite a few web-based applications are rising in popularity. The last section explores some examples of existing Blender add-ons.

3.1 Desktop applications

Desktop applications are not leaving any time soon. Many people prefer the ability to work offline, use the full performance potential of their devices, and mostly the feature-rich software available.

Godot Slides, Unity and Unreal Engine

Three desktop applications, that deal with 3D graphics to a certain degree. Apart from Godot Engine and its recently built Slide module, none has any form of presentation capabilities.

Godot Engine introduced Godot Slides[1] that help users create a game showcase. It is rather drag-and-drop oriented while many elements are created outside and just inserted into slides.

Unity has few existing extensions for creating presentations. First is the presentation project Presentation by UnityTechnologies[2]. It implements all the basic features for creating a presentation. It provides controls for managing slide sequences during presentation and slide visibility. There are also some missing features like proper fullscreen while presenting and possibly exporting slides in more versatile formats. There is also very similar extension UnityPresentationFramework[3].

Unreal Engine has plugin 3DPresentationCreator[4] that adds various features. Users are able to create slides, custom templates, slide numbers, and presenting.

[1] Godot Slides: https://godotengine.org/article/godot-slides-gamified-slideshows-with-godot
[2] UnityTechnologies Presentation: https://github.com/UnityTechnologies/Presentation
[3] UnityPresentationFramework: https://github.com/lunarmoon26/UnityPresentationFramework
[4] 3DPresentationCreator: https://github.com/byteparrot/3DPresentationCreator

Microsoft Office

PowerPoint is part of a Microsoft Office suite and it is one of the most famous presentation software out there. It is a well rounded software that provides a lot of useful tools:

- Create presentations from scratch or a template.
- Add text, images, art, and videos.
- Add transitions, animations, and motion.
- Share and collaborate with others.
- Give presentations, add notes and translations.

Every area of PowerPoint is highly customizable. Each slide can have either a custom or predefined layout style. Slides can have slide numbers and dates and times. Many fonts are available and various options to enhance text style including font-weight, spacing, and color. Interestingly PowerPoint allows users to add videos, audio, and equations. Creating and inserting3D graphics in PowerPoint can be challenging, even though Microsoft added some sort of support for it. PowerPoint supports multiple formats including `.3mf` `.obj` and `.stl` for 3D graphics. In case the user wants to create their custom, they are out of luck, because of lacking tools to help create 3D graphics.

Microsoft Office primarily available on Windows, but it has bacame available on macOS also. This is its first draw back as there is no Linux version. But it does not end here. Microsoft Office comes in two versions:

- Office Home & Student 2019 - One-time purchase.
- Microsoft 365 - Subscription based.

LibreOffice

LibreOffice is a free and powerful office suite. LibreOffice suite includes an application for presentations called (`Impress`) and several more dealing with word processing, vector graphics, and sheets. LibreOffice natively supports Open Document Format (ODF) while it is compatible with the majority of file formats for both import and export[5].

Impress contains multiple editing and view modes that help with general editing, make organizing contents easier, help view and edit notes, or locate and order slides. Impress comes with many sets of slide show animations and effects. Special Slide Show mode makes it easy to show particular slides, changing pointer visibility, and much more. Impress also supports multiple monitors that give the ability to view a preview of the next slide and slide notes while displaying current slide[6].

Compared to Microsoft, LibreOffice provides a cross-platform office suite with a portable edition. LibreOffice can be easily extended by using extensions[7].

[5]What is LibreOffice?: `https://www.libreoffice.org/discover/libreoffice/`
[6]LibreOffice Impress: `https://www.libreoffice.org/discover/impress/`
[7]LibreOffice extensions: `https://extensions.libreoffice.org/`

Apache OpenOffice

Both LibreOffice and Apache OpenOffice were ultimately forked from the `OpenOffice.org` project, resulting in lots of similarities. Both office suites offer essentially the same set of applications. This time presentation software is also named `Impress`, though from a different Office suite. They both share the same and most of the features.

The main differences are release frequency and file format support. OpenOffice[8] tends to release fewer major updates compared to LibreOffice[9]. OpenOffice is can read newer Office Open XML formats, but is not able to export in such formats[10]

3.2 Web applications

Web applications made a big leap forward in recent years. Many users prefer the simplicity of these apps, the portability and speed. Users do not have to install anything, just log-in and go straight to work.

Microsoft Office Online

This is a free, web version of Microsoft Office suite consisting only from handful of applications, including Word, Excel and PowerPoint. The pricing is one of the advantages to the desktop version of Office. Web version also means it is running in browser, which makes it run on pretty much anything from Linux to Android devices. Microsoft added better collaboration tools to the web version such as multiple people editing same paragraph in real-time. Even though they share the same user interface, the web version of these applications are more simplified and have fewer features.

Google Docs Editors

Google released its office suite free of charge for anyone with personal Google account. Consists of multiple office apps, most notably Docs (word processing), Sheets (spreadsheets) and Slides (presentations). Google apps are accessible through web application, mobile app of desktop application on Chrome OS. Though free Google apps come with file size limitations but most of new and popular file formats are supported out of the box.

The biggest advantage of Google apps is collaborative real-time editing. Changes are automatically saved to Google's own cloud storage while a revision history is kept[11]. Google also introduced support for add-ons, that will expand features in editors[12]. Google also created an add-on, that allows users to edit these documents even offline[13]

reveal.js with three.js

Let's explore one more advanced piece of software, that might be useful for some. This JavaScript framework enables users to create fully-featured and beautiful presentations for free using just web technologies[14]. This framework does it all, enables users to change style

[8] OpenOffice releases: `https://wiki.openoffice.org/wiki/Product_Release`

[9] LibreOffice releases: `https://wiki.documentfoundation.org/ReleasePlan`

[10] OpenOffice: `https://wiki.openoffice.org/wiki/Documentation/UserGuide/FileFormats`

[11] Google file history: `https://support.google.com/docs/answer/190843`

[12] Google add-ons: `https://support.google.com/docs/answer/2942256`

[13] Google Docs offline: `https://support.google.com/drive/answer/2375012`

[14] reveal.js: `https://revealjs.com/`

with CSS (Cascading Style Sheets), or add custom behavior using JavaScript. Reveal.js supports a wide range of features like PDF-export, auto-animate, and Markdown.

There is no support for cloud storage or collaborative workflow, but this can be solved using Git and GitHub.

Reveal.js does provide all the features found in much other presenting software. Slides can have simple colors, complex images, or videos as background. A neat feature is a set of features is syntax code highlighting. Like any other presenting software, reveal.js also supports math equations, layouts, and slide visibility. As far as presenting goes, this framework supports built-in full-screen mode.

Unlike reveal.js, three.js[15] is a JavaScript framework used for graphics in web browsers. This framework uses WebGL to create and display 3D objects. In combination with reveal.js, users can create presentations with 3D graphics, a feature hardly possible with any other alternative.

For more information about the capabilities of these frameworks visit their official websites.

3.3 Blender add-ons

The previous chapter introduced Blender, a 3D graphics modeling software with lots of potentials. Creating presentations is not impossible, but just so time-consuming that users do not want to spend that much time. What users want is to take all the tools for 3D graphics and create an environment that helps the user build presentations. While it is a strange combination, there actually exists an experiment with such an idea. This add-on Blender-Interractive-Presentation[16]. It is implemented in older version of Blender and newer version will not open it anymore. Rewriting would require a lot of work while this add-on uses features that are no longer available, especially standalone player. But the idea behind is to implement PowerPoint like features into Blender.

3.4 Features

Most of the mentioned solutions have many things in common. Starting with a live slideshow, they all include a fullscreen player for the presentation, with buttons and shortcuts to control the sequence. As for creating the presentation itself, they share some similarities. They all allow adding images, text, and videos.

There are multiple ways of customizing the text. Including the bullet points, various fonts, and font styles. They support line and word spacing for more advanced customization.

They are all slide-based presentation software solutions. These slides then have the contents of the presentation. Regularly designed using templates and various color schemes.

But they are all missing the ability to process more complicated 3D graphics. None of them supports tools to edit 3D graphics out of the box. Most of them have some sort of animation array to choose from. Some allow users to customized these animations by changing the duration or direction. Blender contains a complex animation system to create complex custom animations.

[15]three.js, `https://threejs.org/`

[16]Blender-Interractive-Presentation: `https://github.com/Asord/Blender-Interractive-Presentation`

Chapter 4

Proposed design

This chapter aims to propose an add-on design that will ease users creating presentations in Blender. Starting with a user interface and later describing tools to provide a better experience. Covers slides, export, dealing with text, and insertion.

4.1 User interface

The user interface is the most important aspect of the add-on. It is necessary to keep the symbiosis between Blender's built-in user interface and the proposed add-on.

It is important to design the user interface in such a way, that the user would be able to find all presentation-related tools in one place. Use of the add-on should be easy to work with, but effective so that it will provide some support. It should not conflict with any other tool in Blender. The user interface should be designed in such a way, that it does not disturb the user while working on a presentation but also while they are not. Add-on should be implemented in a way, where it does lots of things in default settings, but also gives users options to customize them.

There are many editors, where it would be appropriate to place the add-on. But the most sensible option is to use the 3D View as the user can switch between editing and previewing. This way users can interact with the presentation. Users can rotate, move and zoom as it was a standard Blender, and use camera view to preview the current state of the slide.

User interface layout

Considering the possibilities of Blender user interface extensions, it is important to define where each tool will be. Blender provides a wide variety of options to create new user interface elements (buttons, panels, menus, etc.), but they are all uniform, and changing the looks them is not supported. To make things clear, all following user interface prototypes will be a simplified version of the Blenders user interface.

Proposed layout is shown in 4.1. It is a good practice to place certain tools of add-on functionality into the side panel in 3D View. Therefore the user has direct access to all the tools in one place and does not have to look for them in various places.

Simple layout design, with tools to provide help in presentation creating. Everything is one panel in the side panel in 3D View. There are four different panels each for a certain area.

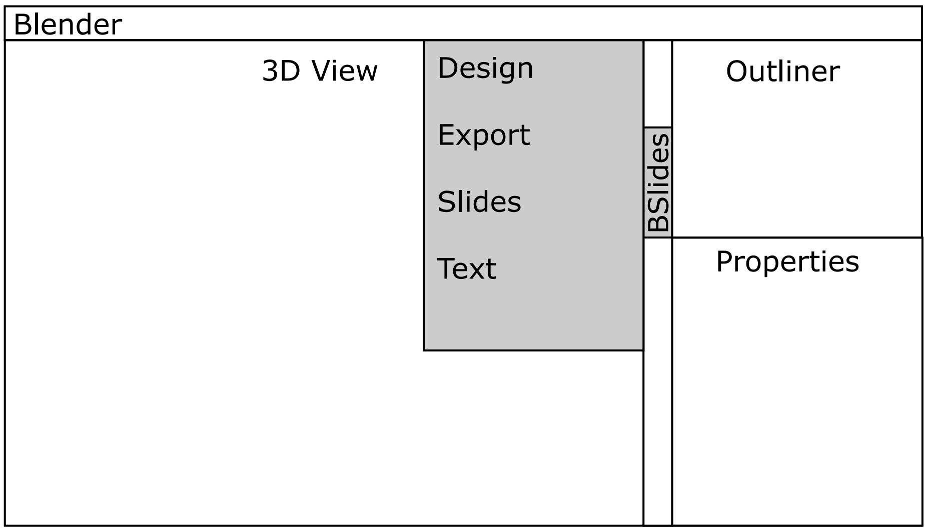


Figure 4.1: Proposed user interface design
Proposed layout in side panel in 3D View, with all panels in one place.

Design panel

Design panel in 4.2 provides options to change slide design. Contains an option to change the slide ratio. This can be then set for all slides. Users can create new templates, or choose and link an existing one. There is also an option to change the slide background. Two more operators from a design standpoint. First, center the objects in front of the camera while the second set the 3D cursor in the center of the camera.

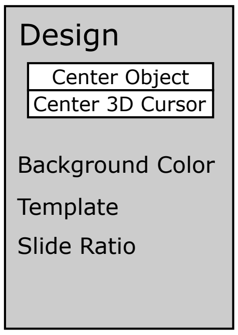


Figure 4.2: Proposed user interface for design panel

Export panel

Export panel in 4.3 provides options to save presentation in more portable file formats. It is possible to render slides as individual images or export slides in `.pdf` format. Users can choose which slides to export.

Slides panel

Slides panel in 4.4 allows user to simply view and manage slides. Two operators, one for creating and the second for removing a slide. Users can choose which slides are visible

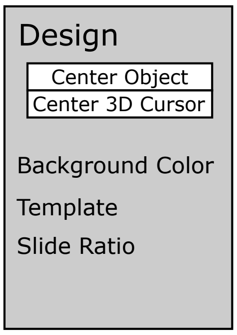


Figure 4.3: Proposed user interface for export panel

during render. If slide numbers are active, the user can change which slides have these slide numbers.

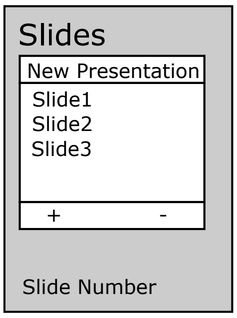


Figure 4.4: Proposed user interface for slides panel

Text panel

Text pane in 4.5 contains many various options for editing text. If the user selects multiple text objects there is a way to merge them or align them underneath in a list-like style. The user can apply font style (bold, italic, underline, and small caps) to the text. More operators include alignment to the left, right, and center. Users can also turn the whole object into upper or lower case. For lists, there is functionality to indent or outdent text. Panels also offer a quick way to access font size, spacing options, and color for text objects. Text panel also contains operators for creating data and time and table of contents.

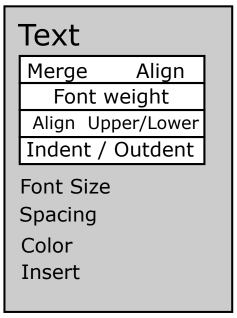


Figure 4.5: Proposed user interface for text panel

Slideshow

The slideshow provides a special mode that will hide everything during the presentation, run it in Eevee render engine. Users can switch slides with ← , → , and exit with Esc key. There is also a built-in control panel with these options.

4.2 Slides

This add-on uses slide-based presentations, meaning users create content on individual slides, and these are then played in sequence. Creating such a slide show inside a single scene would make it unnecessarily harder to animate such slides. Therefore, a different approach was selected. Now, each slide is represented by a single scene in Blender.

New presentation

Starting with a default Blender scene with not usable scene settings. This is simply solved with a `New Presentation` operator. This operator clears the current project, leaving just the bare minimum to start. If it is used for the default scene it will just replace the scene while unlinking all present objects. If however it is used in a more advanced stage of presentation, it will delete all scenes and unlink all objects present. Clicking the operator will prompt the user to confirm the action. Although, nothing is lost after that, as Blender keeps a history of actions, so it can restore them if necessary.

Blender has its system of managing scenes. This system provides a way of creating a new either empty or copy of the current scene. To give the user as little trouble as possible this add-on also provides a simple interface for creating new slides. This will create a new scene, with a camera and background color, ready for building. This approach creates separate scenes every time it is used. Sometimes user wants just delete the whole scene and start a new one. This can be done through Blenders scene with just simple scene delete or use add-ons operator to remove selected scene.

List of slides

This add-on offers a more advanced way of viewing scenes. Although Blender already has a drop-down menu with a list of all scenes present in the project, it is impossible to change anything regardless. Therefore slides panel has a list of all scenes available in `.blend` file. Clicking any of the scenes inside this list will change the current scene to the selected one. This list is also capable of renaming these scenes. After all, users may have many slides in the project, so it would be nice if they could search for one with a particular name. So, this list also uses search functionality for a quicker find. Up to this point, there is not much difference between Blender's built-in and add-ons, apart from visuals. But this is done mainly because the built-in is not extendable. Some slides may not be relevant in certain scenarios, and the user may not want to use them in the presentation. Adding this visibility functionality inside the submenu for every single scene would result in clicking through a lot of scenes. To solve this issue, the list provides visibility for each slide marked by a camera icon. Clicking this would hide that particular scene from rendering in the slide show. To hide multiple scenes at once user can click and drag along those icons.

Slide ratio

Blender creates scenes in 16:9 aspect ratio by default. Sometimes users may want to use a different slide ratio, a less standard one like 4:3 or even 21:9. Changing this to a single scene can be done quite simply, but changing it on multiple scenes will require some time. This add-on offers a way of changing this aspect ratio on all currently created scenes by clicking the `Apply to All` operator. This will apply set resolution to all scenes with just one click.

Slide number

One of the most commonly used presentation features is slide numbers. This number shows the current slide position from the number of all slides available. To make everything reasonably easy slide number is one text object linked to selected scenes. After changing the scene, this text object holding the number will update to show the current scene number out of all visible scenes in the project. To change which scene has a which does not have a slide number use the list with all the scenes. If the slide number object is created this list will show a dot next to the camera. A full dot icon means a slide number is active - an empty dot icon means that the scene does not have a scene number. If the dot icon is missing, it means the slide number object is not created. To create a slide number text object either use `New Slide Number` operator or manually create a text object named `Slide Number`.

Slide visibility affects this slide number. If the slide is hidden, it will not count towards the number of total slides in the presentation.

This slide number text object is linked to all the scenes, meaning all the customization done to one will update all of them. So all it takes is to position the object in a correct location, give it a style and it is ready.

Centering objects

The thing with standard 2D presentations in that presentation software is that everything is created in a limited area. The problem with the 3D space is that everything can be far away from the camera, resulting in not being visible in the presentation. To solve this problem there are two operators. First operator `Center Object` that will place the selected object in front of the camera. This operator supports a redo panel with further customization.

- Copy camera rotation - This will apply camera rotation to the centered object, which will make the object face the camera.
- Center object origin - This will set objects center to the geometry origin, centering it in front of the camera.

The second operator is `Center 3D Cursor`, and it will set the 3D cursor to the center of the camera. This operator will ensure every object is created in this location making it simpler to place objects.

Previewing slides

It is important to see how would look like in the end, especially in 3D space. The problem is that moving any object just a little bit may result in the object not visible in the final presentation at all. Fortunately, Blender offers a functionality called `Camera View`. This will switch to the camera view in the current scene to preview the final view. This way user can adjust objects in 3D space. This approach is valid until users do not pan around the scene. To help with that, a `Preview Slide` is available to preview the slide in the center of the screen.

4.3 Running presentation

Creating the presentation is only half of the problem, the other half is to present it. To run the slideshow in a distraction-free environment, where the presenter can switch between the slides. But there is much more to it than it looks. Blender does not provide any editor which can run the presentation fullscreen. It is missing tools to switch scenes, too.

Window

To create a distraction-free environment, it is necessary to create your own. Using the users' current 3D View is possible. But hiding everything from the viewport presents a problem on its own. Although Blender can hide those menus, panels, and toolbars, most of them only support toggle functionality. And it is near impossible to check if the gizmo is active or not. But Blender can create a new window, that comes with default settings. These settings can then turn off, and with some tweaking fullscreen window with absolutely no distraction can be achieved.

Running a slideshow will result in opening a new distraction-free window. This window is fullscreen and active camera view. After exiting the presentation, this window can be discarded, and none of the settings will transfer to the user's settings.

Slide control menu

Blender does not have any way of switching between slides, so this add-on adds a menu to the full-screen slideshow. This menu contains three buttons for switching back and forth between slides, as well as the exit button. There are currently six positions available to choose from and can be set in the add-on preference menu.

Some sort of slide control menu is also available in the panel in 3D View. These buttons can and will change slides, but the last button is set to start the presentation.

Keymaps

There are also three different keymaps available during the slideshow. These slideshows co-operate with the slide control menu as they do the same job. Instead of using a mouse user can simply press a button to switch to the next slide. These keymaps are currently mapped to:

- ← - Previous slide.
- → - Next slide.
- Esc - Exit slide show.

These keymaps are available only during the presentation and are reverted when the slideshow is over. Keymaps behavior can be unpredictable, and it is best to use the default Blender comes with.

Animation loopback

Lastly, a built-in animation system in Blender needs to be addressed. Blender's default behavior when playing animation is to play the animation in a loop. This is something that cannot be changed and makes a weird effect during the presentation. To change this effect, this add-on provides an option to stop the animation loop, and just play it once. This makes more sense in the presentation context, but if the user insists on using the loop playback, there is an option to toggle in add-on preferences.

Animation auto playing

If the user decides to use animations in their presentation, there is an option in add-on preferences to run these animations on slide change. These animations will always play from the beginning.

4.4 Export

To run the slide show is necessary to have Blender present on the computer, which can make some things rather complicated. Although Blender has a portable edition which is 200MB[1] in size, and it is much more than a traditional presentation from other alternatives. Blender used to have a game engine, which had a standalone player[2] that could run `.blend` files, but it is discontinued. Blenders cycles render engine also has a standalone renderer, but this add-on is based around Eevee. And as of writing, there is no such standalone player/renderer for Eevee. This gets a little tricky.

Export formats

In some cases, users may want to have a version of their presentation that will work without Blender. This add-on supports export to two formats. First more traditional will render all scenes into `.jpg` files. Users then can use the .zip file and transfer it or use other presentation software to process them. The second option is to export these scenes into a single `.pdf` file. This way user does not have to do anything, and a portable version is available right away. To create this `.pdf` file a python package Pillow[3] is used. This package is not installed right away, rather users have to install it. To make things easier this add-on offers a one-click install of this package to Blender. This package is installed inside Blender's python directory at a user level this way it does not conflict with any other versions of Pillow already installed. It is also installed at the user level so it does not require elevated privileges.

4.5 Text

This section explores probably the most important feature of any presentation, it is the text. Using text sensibly can make the presentation more entertaining, while using too much can leave a bad impression.

[1]Blender Portable: `https://www.blender.org/download/`
[2]Blender Player: `https://docs.blender.org/manual/es/2.79/game_engine/blender_player`
[3]Pillow: `https://pillow.readthedocs.io`

Blender supports text objects out of the box. Although it offers many various customization options, working with it can be stressful. Let's explorer what features does this add-on adds and improve.

General

This add-on has two more general-purpose operators. Working with multiple text objects at the same time might be tedious, therefore a merge text object operator is present. This operator, as the name suggests will merge all selected text objects into one. The one, that all remaining ones will be merged into is the first by the name of all selected objects.

Another operator is for aligning text objects on one axis. This way, the user can have multiple text objects aligned at once. This operator offers an option to change the spacing. Although primarily not used for lists, this can make creating them simpler. Creating a collection with multiple text objects and aligning them will result in a list. This add-on provides tools to increase or decrease the indentation. This way, a hierarchy of text objects can be created.

Font style

Before explaining anything about font inside Blender, it is important to understand how fonts work. Blender, unlike any other software, has different font slots for font style (bold, italic, etc.). This is confusing as setting font, for a text object and later making it bold will not change the text style unless bold style has particularly set font.

To make something more or less important it is good practice to use font style. This way, viewers will see the key concepts emphasized. Blender already has tools inside text object properties, but they are somehow counterintuitive. Clicking bold style on objects will not change anything. This is due to Blender's misleading naming. Blender will use the bold style on newly inserted characters instead of applying the style on selected.

To address this, the text panel provides similar functionality to alternative software for presentations. Clicking bold in object mode will change the text object to bold. Clicking bold in text mode will set selected characters' style to bold.

This add-on also offers additional tools to make all characters upper or lower case. This functionality is similar to the previous with object and text object.

Font size, spacing and alignment

These options are placed into this panel for quicker access. Use of font size is recommended for font style management therefore having it above the panel would be more practical.

Changing spacing or alignment is not something used every time but nice to have in one place with other text object-related tools inside the add-on.

Color

No matter the object type, Blender supports materials through nodes, and text object is no exception. Even though users can apply any material style they want, the recommended style is to use RGB as a surface. This mimics mechanics in other presentation software and is a guarantee to provide great visibility of text on the slide.

To make text objects coloring simpler, this add-on provides a shortcut to directly create new material with an RGB surface. Users can quickly adjust the color by the provided panel. This panel shows the first material from material slots.

Font management

Each time a new text object is created, a new data-block is generated. This means that the user has to set the style to all the objects. Linking data-blocks will not work as this will also copy a text. This will result in all objects having the same text value.

To solve this a font management panel is introduced. This panel contains a list will all stored objects that can be used as a template. For every other object, the user can just apply the style from the template. Users may have multiple text object templates and use them to copy styles between objects. This does not copy the object's text value, but all remaining font settings.

To get the most out of the font styles, use font size instead of object scaling. This way, all objects will have the same size.

Date

Used once sometimes more, but it is a great addition. Even though it is simple to create a text object and type in the current date and time, still every presentation software provides this shortcut.

This add-on also implements date and time insertion into a presentation. After inserting a date text object, a redo panel pops up with some customization. For example, there is a date separator symbol. Some people may be used to slash (/), others dot (.) so this way, users can choose whichever they like, even a custom one.

But apart from the date, there is also an option to add time through a redo panel. Again, this can be customized to a person's liking. Placement can be altered between the bottom or in a line. Lastly, a time format can be changed. There is support for both 24 and 12 hours day.

Table of Contents

Table of Contents (ToC) gives users an overview of the document's contents and organization. It is possible to create a simple text object with all titles throughout the presentation. But a little change would mean rewriting the text body again. This add-on offers a quick way of generating a single text object with all titles from the presentation. All it takes is to create a text object and name it with `title` prefix, and it will be added into the ToC. There is a catch if a slide contains multiple objects with such a name only the first one (sorted by name) will be used. To create a title object a `new_title` operator. This will create a new text object in the center of the camera. If any there is a text object selected on option to convert it to title will appear in text panel. This adds the title prefix and allows the option to center the object in front of the camera.

4.6 Reconsidered functions

Many features are missing that can be found in any other presentation software. The reasoning behind this is that is it either impossible from an implementation standpoint or its existence would add more complexity than needed.

Blender has a really strong animation mechanism, that building anything above it would be unnecessary for presentations. The first impression was to implement a set of predefined animations, that users could choose from and further customize them. But why would anyone click through a bunch of various panels, if they can achieve a similar result with much less effort?

Lists are partly implemented. As the way, Blender uses cursor inside text edit mode is done internally with no way of accessing its location. Solving this with a separate operator and forcing users to click a button every time they want to add a new entry to the list is just bad. In comparison to manually adding a bullet point or number.

Chapter 5

Implementation of the add-on.

The last chapter describes the actual implementation. It describes how are certain features implemented and what types of problems had to be solved. Last part deals with sharing of the add-on to the public. For implementation Python programming language was used mainly from books [5], [8], [1] and Python documentation[7]. For the Blender data access official Blender API[3] and Blender Python API book[4] are used.

5.1 Add-on starting point

Every add-on in Blender must be a Python module. This add-on uses folder named `blender_slides` is created and initialized with `__init.py__`. This file then includes `bl_info` dictionary with an add-on description. Blender uses this as a starting point for the add-on.

To make sure Blender will not run into any issues with registered classes a naming convention is used. Every class registered as a part of this add-on starts with `BSLIDES_` and is followed by the type. Types include `OT` for operator, `PT` for panel, `MT` for menu and `PG` for property group.

Class registration

Every class that is required to register is registered inside this file. This is done by calling `register` function. In earlier stage a `autoload.py` from Blender Development Extension, that was later replaced with manual register. Every file in this project contains a `register` and `unregister` functions. These functions can be than called to register all classes from that file. So all it is left is to include these files are call appropriate function.

Keymaps

Keymaps have to be registered before they can be used. Although this add-on only registers three, the problem is that these keys are assigned to different Blender functions. To overcome this issue, all the keymaps are created inactive so that users do not get irritated by that. Listing 5.1 shows sample code of creating a new keymap for the next slide button.

```
km = kc.keymaps.new(name="3D View", space_type="VIEW_3D")
kmi = km.keymap_items.new(
    "bslides.next_slide", type="RIGHT_ARROW", value="PRESS"
)
kmi.active = False
```

Listing 5.1: Creating a keymap for next slide button in inactive state.

After running the presentation this then gets activated by changing the keymap `active` variable to `True`, when closing is it reverted again.

Add-on preferences

Every class is placed inside meaningful location apart from add-on preferece. This class inherits from `bpy.types.AddonPreferences`. This class is used for general settings for the add-on itself. It contains properties for slide control panel location or loop animations and auto play animations. This class defines its own `draw` functions which defines the looks of the panel inside Blenders add-on preference panel.

5.2 User interface

User interface (UI) source files are all under `ui/` folder. This folder then contains four files each for single panel in add-on. The final user interface can be seen in appendix B.

General panels

`Design` panel UI is implemented in `design_ui.py`, `export` panel in `export_ui.py`, `slide` pane in `slide_ui.py` and `text` in `text_ui.py`. Every file implements various panels and subpanels to create the final look of the add-on.

Every class has attributes that tells it, where to place the panel. As the add-on is in one place all of the classes have the same values of there variables. Listing 5.2 shows the values of the variables to have the add-on in 3D View in side panel.

```
bl_space_type = "VIEW_3D"
bl_region_type = "UI"
bl_category = "BSlides"
```

Listing 5.2: To place panels in 3D View in the sidebar, classes set these values

Some panels include subpanels or panels in panels. To achieve this effect a mix-in inheritance is used. A base class defining a location of the main panel is created. All subpanels then inherit from both `bpy.types.Panel` and the base class. To make everything work, the child panel has to set `bl_parent_id` to the name of the parent class name.

Slide control panel

To achieve panel with three buttons to control the sequence of slides a custom gizmo class implemented. This class inherits from `bpy.types.GizmoGroup`. Althought gizmos have been in Blender for quiet a while now, a proper documentation is still missing. The only somehow usable guide was found on Blender devtalk[1]. This class uses two function to work properly:

[1]Gizmo Button: https://devtalk.blender.org/t/how-to-create-overlay-buttons-like-in-the-3d-viewport-with-python

- `draw_prepare` - This function predefines the location of buttons in 3D View.
- `setup` - This function is called every tie this gizmo is rerendered and it sets actual buttons to these predefined locations.

The actual location is fetched from add-on preferences and evaluated to get the exact location of the screen.

This gizmo control panel is only showed during a presentation and only when a top panel is hidden. This is checked in `poll` function. If the top panel is visible, this panel then contains the buttons to control the sequence. This is achieved by appending a custom draw function into the `bpy.types.VIEW3D_HT_header` list.

Custom icons

Most of the icons were used from Blender's built-in icon collection. But two of them were an exception. There was no way of describing an increasing and decreasing indentation. Two custom icons were created and loaded into Blender.

This functionality can be found in `icons.py` file. Instead of using a global variable for icons, this uses `find_icon` function to return the icon. These custom icons are loaded from `icons/` folder via Blenders preview collections. Loading of these icons is done by using the `load_icons` function, which loads all icons with `.png` extension. All loaded icons are then stored as a preview collection value inside a dictionary under `icons` key.

User interface lists

There are two lists used in this add-ons user interface. The first is in slides, and the second is in font styles. These list classes inherit from `bpy.types.UIList` and require `draw_item` function to work properly. This function then defines the looks of each item. The `draw` function in `BSLIDES_UL_font_styles` defines each item as shown in listing 5.3. Item is set to display the name of the item, with no text, icon set to built-in Blender one, with no embossing or translation.

```
layout.prop(
    item, "name", text="", icon="SYNTAX_OFF", emboss=False, translate=False
)
```

Listing 5.3: A snippet of UIList item definition from font styles list.

These lists require a `CollectionProperty` and a index as `IntProperty`. Then they use this index to mark currently selected item from the collection property. In case of `BSLIDES_UL_font_styles` both the collection and index are registered as properties of `window_manager`. The collection item is defined as a `PropertyGroup` with a `StringProperty` for the item name and a `PointerProperty` for the source object.

5.3 Slides

This section describes the implementation of individual slides. Source code are in `slide.py` in `operators/` folder. Slides are represented by scenes in Blender. This terms are interchangeable, but this thesis references scenes from implementation stand point while slides from user perspective.

Slides

Slides can be hidden from rendering during a slideshow. To achieve this a visibility property is registered for each slide. A `BoolProperty` is used and set to `True` by default. This way, every new slide is rendered. Slides with `render_slide` set to `False` will not be rendered during slideshow. Users can use the slides list to turn visibility on or off by clicking the camera icon.

New slides are created with `new_slide` operator. This operator then creates a new empty scene, with a new world and camera. A new world is created by calling `new` on a world data type. This same function is called upon camera data type. These objects are then linked to the empty scene. Lastly, this scene is set to the context as currently selected scene and is now a slide user can use.

To delete the scene there is a `remove_slide` operator. This operator removes a slide by calling `remove` function upon scene data type.

New presentation is implemented as `new_presentation` operator. This operator removes all slides and creates a new single slide. Users have the option to add a title through the redo panel. Before removal a user prompt is shown by `invoke_confirm` from `invoke` method in operator. This method is called before the execute function is.

Each slide has its world. This world is used to create the slide background. If slide background is reset the color is set to `(1.0, 1.0, 1.0, 1)` and strength of `1.5`. This creates a solid white background. Users can change the color from within the design panel. When the user creates a new slide, color from the recently selected slide is used. For changing color on all slides, the user can change the color on a single slide and then use `apply_all_slide_background` operator.

Same apply to all mechanism is supported in aspect ration. Users can select preferred slide resolution and use this setting for all remaining slides.

Switching slides

Slides are easily switched when in Blender familiar user interface. But during the presentation, all these menus ale panels are hidden. To solve this a slide switch panel is used. This panel is just a UI element, that has to execute an action after clicking. This is done by two operators. Both of these operators work the same but in opposite directions. The operator first loads all the slides, that have visible property set to `True`. This will ensure no hidden slides will be used during the slideshow. In the case of the next slide, the current slide index is increased by one and decreased in the case of the previous slide. If the new index is out of range the slide will not change.

User preferences are loaded from add-on setting whether to autoplay animation are played or not and set accordingly.

There are two places where these operators are used. First is the mentioned slide switch panel during the slideshow, and the second is in the 3D View header. To make sure only one of them is visible at any given time a `poll` method in the gizmo panel, checks if the panel is visible. Listing 5.4 shows the poll method to ensure the gizmo panel is visible in case of a hidden header panel.

```
def poll(cls, context):
    return not context.space_data.show_region_header
```

Listing 5.4: The poll method, which tells the gizmo panel if it should be visible.

Templates

Templates were implemented so that users can create a single collection with the slide style usable on multiple slides. If the user decides to change the style all other slides with this template will get updated. This is done through linking and unlinking the template collection to or from slides.

This template system works by searching for collections in `.blend` file. Every collection used as template must have prefix `template` (not case sensitive). This templates are then shown inside a drop down menu in `Template` submenu.

Users can create one by clicking the `+` symbol next to the template drop-down menu. This will ask for the name of the template and automatically link it to the current slide. This is done in `execute` function where a new collection with the given name is created and linked. To unlink any template user can choose the particular template and click the `Unlink` button. In case the selected template is not linked to the current slide, there will be a `Apply` button. This is done by linking or unlinking the selected collection to or from the scene collection of the current slide.

When slides are switched and the user has enabled the animation autoplay it will automatically play it. This is done by checking the user preference and acting accordingly. To play the animation operator `animation_play` from Blender is used. Starting a presentation and without stopping it would end up in a loop. Blender has no feature for stopping playing the animation in a loop, so this add-on implemented one. It is implemented as an handler subscribed to `frame_change_post`. The function checks if the current frame is the last and if so, it stops the animation.

Slide numbers

Slide numbers are implemented as a single text object linked to all slides with slide numbers enabled. If no object named `Slide Number` is created, it will provide an option to do so. This is done by executing an operator. This operator creates new text objects which are then linked to the current scene.

Each slide has a property for a slide number, denoted by a dot in the slide list. Clicking it will disable the slide's slide number. This will not take effect till the slide numbers are updated. This can be done through the reload icon. This will execute the operator to update the slide number by going through all the slides and either linking or unlinking the slide number object accordingly. To notify user the user on this change a `BoolProperty` in `window_manager` is used as a dirty bit. This bit will be set if any of these slides had their slide number property changed. This is an unfortunate solution, as it could be automated. Even though every property has an update callback, these functions do not get a reference to the object whose property has changed. Meaning, these functions do not know the exact slide that had the slide number changed they only know that it changed. This could be solved by letting the callback function go through all the slides, but that would result in unnecessary looping over all the slides.

Instead of using the update call back function, this add-on implements a handler. This handler is called whenever there is a change in frames. To achieve this a handler was registered to `frame_change_post`. This handler is implemented in `handlers.py` and it changes updates the text value of slide number text object to reflect the current slide. Any hidden slide will not get counted towards the slide number.

Fullscreen slideshow

After clicking a play button in the 3D View header, a new window is opened. Blender has a `New Window` operator in `Window` panel, but there is a catch. This window, when opened creates a new entry in `.blend` file in `Screens`. Every window in Blender has a property named `is_temporaty`, and when created by using the operator, this variable is set to `False`. Because this variable is read-only and set in Blender internally, there is no way of changing it. This will result in unnecessary clutter in `.blend` file after just a few runs of the slideshow. Another problem is, that Blender keeps these references stored even after saving the file, there is no such user mechanism as found in other data-blocks.

But Blender can, create new screens with `is_temporary` set to `True`. There are two options to consider, one of them is a `Preferences` menu found in `Edit` menu, the other is a `View Render` in `Render`. Both of them are created as `temp` screens, which will not be saved in `.blend` file. But after closer inspection only the first option is applicable. The problem with the render screens is that Blender implemented a feature that allows users to set the behavior of temporary screens[2]. This results in new screens opened in an existing window rather than a new one.

The most important part is that the screen has to be opened in a new window. Blender supports a fullscreen feature, but there is no way of knowing whether the user does or does not use it currently. Only thing that is available is `window_fullscreen_toggle` operator, and it does what the name suggests. It only toggles the value. To fix this the new window comes into place. Blender opens every new window in the non-fullscreen window - toggling the fullscreen value will always result in a fullscreen window.

After successfully opening a new window in fullscreen, it is a matter of changing the editor type to the 3D View, setting overlays and gizmos visibility to `False` and switching to the camera view. The camera view part had to be tweaked a little bit. While the operator is creating the new window, the current screen in context is still the main window, which will make the operators fail. To solve this, the context of `view_center_camera` operator was overridden by the temporary window.

This complicated approach to creating fullscreen may look complicated, but this window can be closed with no effect on users' user interface settings. It will also not create any redundant window references in `.blend` file.

Align objects

For aligning multiple objects in single-axis a `align_text_objects` is used. This operator takes all selected text objects and aligns them under or above each other. This is done by moving these objects around their local axis. These objects are stacked with spacing equals to the number entered in the redo panel.

Center 3D cursor

New objects are created in the location set by the 3D cursor. To create objects in center of the camera users can use the `center_3d_cursor` operator. This operator sets the location of the 3D cursor to the center of the camera's frame. To get the center of the camera the corners of the frame have to be obtained. This is done by `camera_frame` in `utils.py`, which returns list of four points in 3D space. These points are averaged, and the final result is the camera frame center. This point is then set as the location of the 3D cursor.

[2]Temporary screen behavior: `Preferences > Interface > Editors > Temporary Editors`

Center objects

For centering objects in front of the camera, users can use `center_object`. This operator moves objects to the line of the camera. It allows the user to copy the camera's rotation and center the object's origin. To calculate the new location of the object, `intersect_point_line` from `mathutils` is used. This function takes three arguments. It takes two points that create the line and one point to calculate the intersection. This function then returns the location of the intersection that becomes the new object's location.

Copy object

Templates are good for creating static styles. But in the case of titles users would have to manually adjust the title text object. The `copy_to_all` operator allows users to copy the selected object to all slides in the presentation. This will create a copy of these objects allowing for changes in the text.

5.4 Export

Export operators contain classes for installing required Python packages and the actual exporting operators. `BSLIDES_OT_install_packages` is a operator that installs the Pillow package. This operator uses Blender's built-in Python executable and bundled pip to install the Pillow package. To execute the install command in pip `subprocess.call` from Python was used. Unlike in Windows, Linux had some troulbe with installing packages which was later fixed by first calling `ensurepip` module from Python. There was also another problem with getting the actual Python executable. Blender changed location of its Python executable somewhere along version 2.91[3]. To solve this a function `get_python_path` from `operators/utils`, which checked the version of Blender and appropriately returned the path of executable.

The export is then done through another operator. This operators takes the values from the export panel and acts accordingly. Users can render set which slides (visible, hidden) they want to export. This operators creates a list of these slides and calls render for each of these slides. Final images are stored into user provide directory. In case user selects `.pdf` as output format, these images are then combined via Pillow package and also stored into the same directory. But this time images are removed and the final `.pdf` is named based on user input.

5.5 Text

This section describes steps on text object operators implementation. This operators are implemented inside `text.py` in `operators/`.

Font format

Changing font style (bold, italic, underline, and small caps) is supported in both object mode and edit mode. Clicking any font style in object mode will transform the whole text. This is done by using a `BoolProperty` registered for `TextCurve` object. This property tells is all letters are a certain style. With changing the value, an update function is called

[3]Blender Python: https://github.com/Thicket-Blender/thicket/issues/48

that sets the value for each character accordingly. When in edit mode the selected text is transformed using Blender's built-in operators.

To change the text to either upper or lower case, users can use both object and edit mode. Object mode uses the `upper_case` operator. This operator changes the whole string to either upper or lower case. This distinction is achieved by using a `EnumProperty`. If users the edit mode, this will transform only the selected characters. This transformation is carried out by Blender's internal font operators.

Increasing and decreasing the indentation is done `indentation` operator. This operator takes two parameters. First is the number of spaces to indent, the second is for the direction. In case of increasing indentation, the given number of spaces will be prepended to the text value of the object. The decreasing indentation operation checks if the given number of spaces is available and then strips the given number of spaces from the text. Decreasing will not remove more spaces than available.

Both font size and font spacing are properties taken from Blender's built-in object properties. As this is an internal functionality there is no way of extending this feature.

Working with font as of writing is all over the place. Blender does not provide the location of cursor in edit mode, nor does it provide which part of the text object is selected. Without this any type of meaningful operators upon text objects in edit mode is impossible.

Font style

Font styles are used storing visual look of text objects. This style can be later copied on any other text object. This way users can create styles for certain text objects. List was introduced earlier in user interface section. To create new text style, select a text object and by clicking `+` sign. This will execute the `new_font_style` operator and save selected objects for reference. To remove any font style, select the style to remove and click the `-` sign. Both of these operators work upon a collection of font style objects. This collection is registered into `window_manager`.

To apply selected font style, choose a text object and click `Apply Font Style`. This will execute the `apply_font_style` operator and copy the font style to the selected text object. This copying process first fetches all the writable properties of text object data and then sets the values for new object based on font style. This approach can skip object name and text value from source object.

This approach has its drawback, and that after reloading Blender's instance (restart), all of the styles are gone. This is due to Blender's data-block storage. Blender considers `scene` as a top-level storage. Registering a property to the scene would save them but deleting that particular scene would also delete all the styles. There is also a possibility of storing these styles into add-on preferences, but this would result in font styles shared between all `.blend` files. There is no way of creating a custom object and storing it directly to the `.blend` file. To create a property that is available for all slides more general storage is needed, and that is `window_manager`. This then creates a `.blend` file wide storage for font styles.

Color

To create color for text objects a `add_text_color` operator is used. It will create new material for the object and prepend it to the first material slot. After creating new material, the default nodes created by Blender are removed, and a new RGB node is created. This RGB node has color set from `FloatVectorProperty`. To create a color wheel style for the color selection a `Color` subtype is used on the property. Users can set the value in the redo panel appropriately.

Date

Date and time text objects are generated using `date_text` operator. This operator creates a single text object with the current date. It also supports a redo panel, which provides more customization options. First of all, it supports date delimiter (separator) for the date, month, and year. This is achieved by formatting string with the value from `StringProperty`. The operator allows adding a time to the date. It supports 12 and 24 hours by providing a drop-down menu. This menu is a `EnumProperty` with these two options. Operators then check which option is selected and formats the output accordingly. Time and date are accessed from Python's `datetime` module.

Table of Contents

Generating table of contents is handled by `generate_toc` operator. This operator creates a list of all text objects throughout all visible slides. It will only use the first text object with prefix `title` in the object's name. All other objects with this prefix on the current slide are ignored. After collecting all title objects a new text object is created, and its text value is set to all these titles.

The final text object is then linked to the current slide. Titles are in a list format, each on a new line. The operator allows for customization for the bullet point character.

Merge text objects

In case of users need to join multiple text objects into one they can use the `merge_text` operator. This operator takes the first object sorted by the name and appends all remaining ones while unlinking them.

Title

For creating new or converting existing text objects into a title two operators are available. First `new_title` will create a simple text object with prefix centered in front of the camera. The second `convert_title` operator will take the selected text object and convert it into a title with options to rotate and locate in front of the camera.

5.6 Public availability

The last part was to make the add-on publicly available. To make things simpler a GitHub repository was used throughout the implementation. So the simplest way of making it available to download, it to create a release on GitHub. This way, anybody can download and start creating some presentations. To make sure everybody would know how to use this add-on, two variants of the manual are available. First is a `README` file, describes the basic on how to install and accessible features. The second is more thorough compared to the previous. This manual contains each feature detail and how to use them. This manual also includes a description of the user interface. This manual is available in wiki pages for the repository.

Chapter 6

Conclusion

The aim of this thesis was to create an add-on for open-source software Blender, that would improve development of presentations. As of writing, there is no such tool, that would help during the process of designing and presenting a presentation.

To create a proposal of the solution first Blender basic had to be studied to the point how things work behind the scene. Using Blender's UI and later API via built-in console. More things were learnt through books and forums.

Result of this project is a open-source add-on for presentations in Blender. Add-on in hosted on GitHub page free of charge. This add-on is known to work on both Windows and Linux OS. Various examples can be found in appendix C.

To make the the add-on feature-complete some things about Blender would have to change, mainly in areas of text object (index), export (standalone player) and way to store data on more general level than just scenes. Much sophisticated animation system could be implemented, to create animations with few clicks. Animations that are played in order waiting for user input and support for `.gif` format during presentation. Some of these can be solved by using other, more single purpose add-ons.

Bibliography

[1] BEAZLEY, D. *Python Cookbook.* 3rd ed. Brian K. Jones. Sebastopol, California: O'Reilly Media, 2013. ISBN 978-1-449-34037-7.

[2] *Blender 2.91 Reference Manual* [online]. 2021 [cit. 2021-1-12]. Available at: https://docs.blender.org/manual/.

[3] *Blender 2.91.0 Python API* [online]. 2021 [cit. 2021-1-12]. Available at: https://docs.blender.org/api/current/.

[4] CONLAN, C. *The Blender Python API.* 1st ed. Bethesda, Maryland: Apress, 2017. ISBN 978-1-4842-2802-9.

[5] DOWNEY, A. *Think Python: How to Think Like a Computer Scientist* [online]. 2nd ed. Needham, Massachusetts: Green Tea Press, 2015 [cit. 2021-3-22]. Available at: https://greenteapress.com/wp/think-python-2e/.

[6] GUMSTER, J. van. *Blender For Dummies.* 3rd ed. Hoboken, New Jersey: John Wiley & Sons, Inc., 2015. ISBN 978-1-119-04700-1.

[7] *Python 3.9.4 Documentation* [online]. 2021 [cit. 2021-2-16]. Available at: https://docs.python.org/3/.

[8] RAMALHO, L. *Fluent Python: Clear, Concise, and Effective Programming.* 1st ed. Sebastopol, California: O'Reilly Media, 2015. ISBN 978-1-491-9-46008.

Appendix A

Contents of the CD

- **examples/** - Examples of presentations
- **latex/** - LaTeXsource code for thesis
- **src/** - Implementation source code
- **blender_slides_v1.2.0.zip** - Installation archive for Blender
- **Manual.pdf** - Detail usage manual
- **README.txt** - Basic description
- **xtelma00_bp.pdf** - Thesis text
- **xtelma00_video.mkv** - Showcase video

Appendix B

User Interface

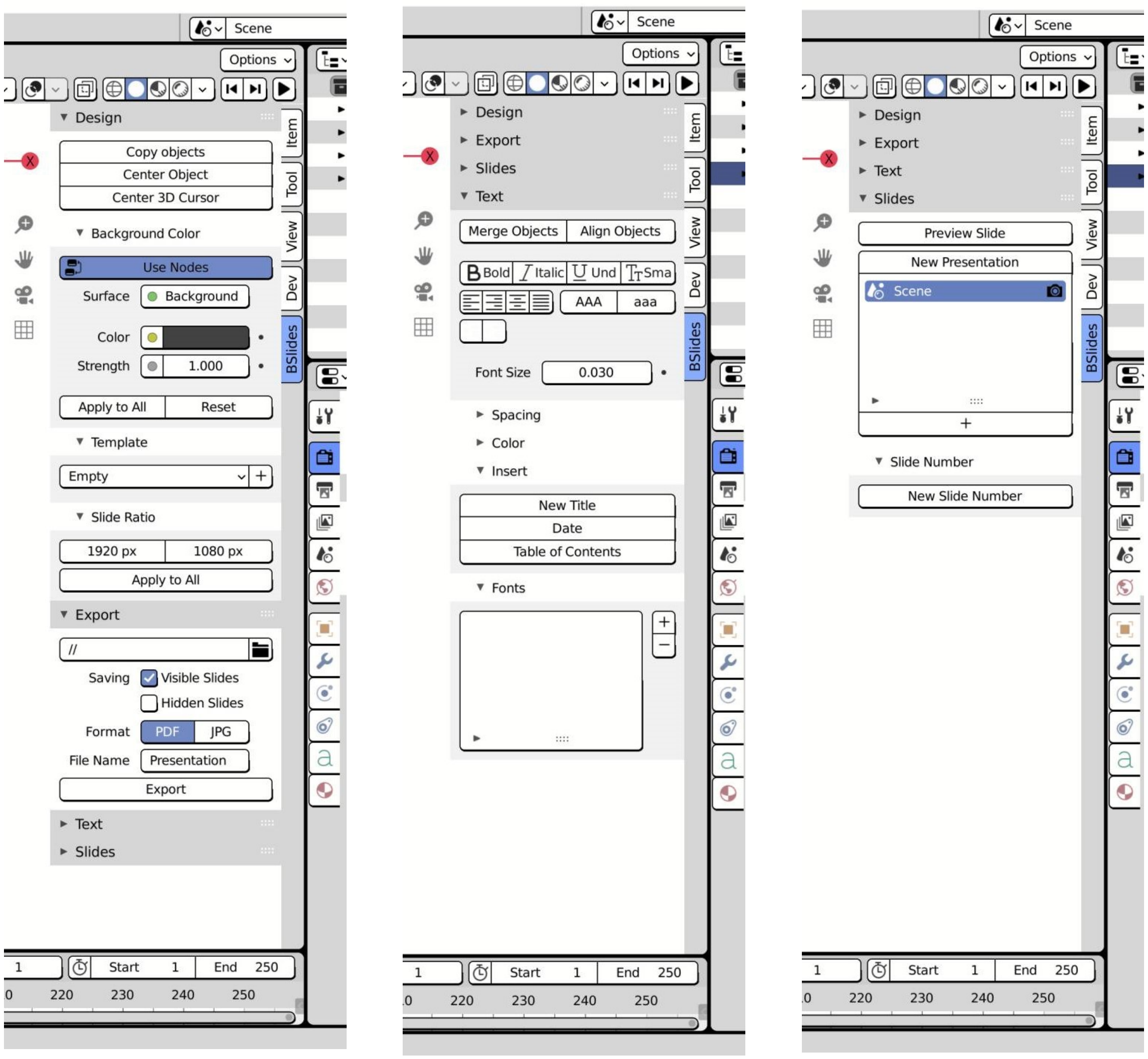
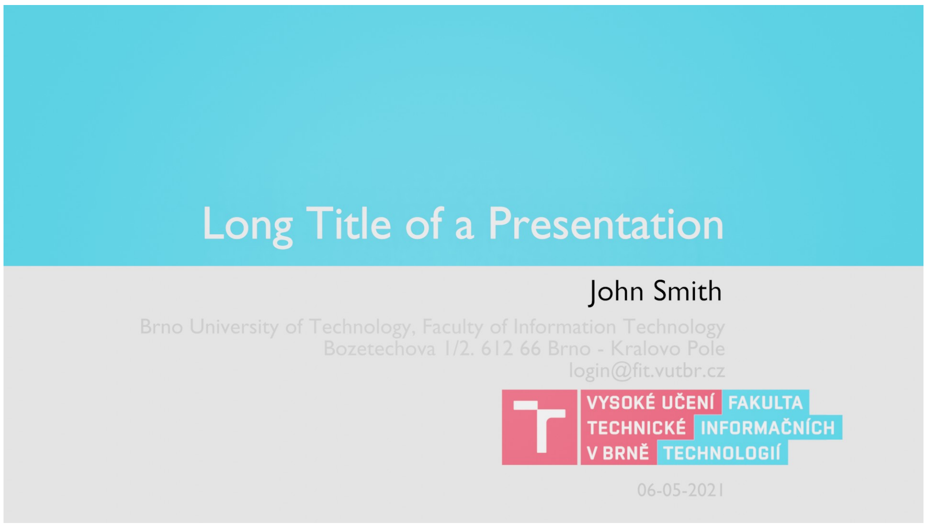

Surface: Background
Strength: 1.000
Slide Ratio: 1920 px, 1080 px
Format: PDF
File Name: Presentation
Font Size: 0.030

Figure B.1: Add-on's user interface
Design and export panel (left), text panel (center) and slides panel (right).

Appendix C

Examples



Long Title of a Presentation

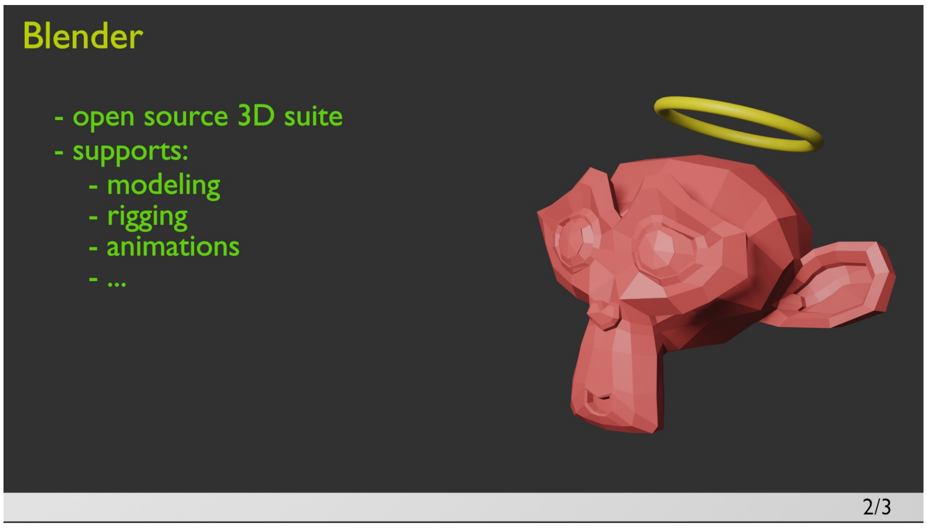
John Smith

Brno University of Technology, Faculty of Information Technology
Bozetechova 1/2. 612 66 Brno - Kralovo Pole
login@fit.vutbr.cz

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

06-05-2021

Figure C.1: VUT FIT presentation template



Blender

- open source 3D suite
- supports:
  - modeling
  - rigging
  - animations
  - ...

2/3

Figure C.2: Basic 3D slide with Suzanne