



Xuan Phung Ton

AUTOMATION OF THE TUTORIAL  
WALKTHROUGH FOR RIGHTWARE  
KANZI UI DOCUMENTATION

School of Technology

2021

VAASAN AMMATTIKORKEAKOULU  
UNIVERSITY OF APPLIED SCIENCES  
Information Technology

## ABSTRACT

Author	Xuan Phung Ton
Title	Automation of the Tutorial walkthrough for Rightware Kanzi UI Documentation
Year	2021
Language	English
Pages	55
Name of Supervisors	Pirjo Prosi and Timo Kankaanpää

---

The topic of this thesis is the implementation of automated tutorial walkthrough UI tests for Rightware's Kanzi Studio using Microsoft UI Automation, Test Stack White, and Coded UI Frameworks. The UI tests were created so that the steps indicated in Kanzi Documentations are walkthrough automatically. The idea of the thesis and licenses were provided by Rightware Oy, who also owns the project implementation. Therefore, no source code can be provided in this thesis report.

The professional background needed for this thesis work contains programming skills in C#, familiarity with the Jenkins tool, proficiency in using Visual Studio, and familiarity with the GIT version control system.

The thesis work was implemented to that there will be around ten automated tutorials walkthrough UI tests in the Kanzi UI automated test tool, these tests shall pass during the daily UI test runs on the Jenkins tool. Additionally, this thesis report discusses the theoretical aspects of the thesis implementation and necessary software testing background.

---

Keywords	Software testing, automation test process, test automation framework
----------	--

## CONTENTS

1	INTRODUCTION.....	9
2	WPF APPLICATION .....	11
2.1	.NET Framework in General.....	11
2.2	Microsoft Visual Studio.....	11
2.3	WPF Application.....	12
2.3.1	Definition .....	12
2.3.2	Architecture .....	12
2.4	Kanzi UI Application .....	13
3	SOFTWARE TESTING .....	15
3.1	Software Testing Definition and Types of Software Testing .....	15
3.1.1	Definition of Software Testing.....	15
3.1.2	Different Types of Software Testing .....	16
3.2	Need of Software Testing.....	16
3.3	Software Testing at Rightware .....	17
4	UI AUTOMATION TESTING .....	18
4.1	Definition of UI.....	18
4.2	UI Automation Testing.....	18
4.3	UI Automated Testing at Rightware.....	18
4.4	UI Automation Testing Tools Used at Rightware.....	19
4.4.1	Microsoft UI Automation .....	19
4.4.2	Test Stack White .....	20
4.4.3	Microsoft CodedUI.....	20
5	KANZI UI DOCUMENTATION .....	21
5.1	About Kanzi UI Documentation .....	21
5.2	Kanzi UI Tutorial Documentation.....	21
6	TOOLS FOR UI TEST AUTOMATION.....	24
6.1	Kanzi UI Test Framework.....	24
6.2	Continuous Integration .....	24
6.3	Continuous Delivery .....	25
6.4	Jenkins .....	25

6.5	Artifactory .....	26
7	IMPLEMENTATION OF AUTOMATED TUTORIAL WALKTHROUGH UI TEST .....	28
7.1	Benefits of tutorial walkthrough UI test .....	28
7.2	List of tutorials implemented.....	29
7.3	Implementation of the UI Tests.....	42
7.4	A Scenario when Implementing Tutorial Walkthrough UI Test ....	45
7.5	Testing for Kanzi UI Tests .....	50
8	CONCLUSION .....	53
	REFERENCES .....	54

## LIST OF FIGURES AND TABLES

<b>Figure 1.</b> WPF Architecture (Chauhan, 2018). .....	12
<b>Figure 2.</b> The Software Testing Lifecycle (Peforce, 2018).....	15
<b>Figure 3.</b> Types of Software Testing (Guru99 2021c). .....	16
<b>Figure 4.</b> Screenshot of Kanzi Documentation Tutorials. ....	21
<b>Figure 5.</b> Screenshot of steps in a tutorial .....	22
<b>Figure 6.</b> Screenshot of a part of Kanzi tutorial .....	23
<b>Figure 7.</b> Rightware's Studio UI Test Jenkins build job .....	26
<b>Figure 8.</b> Kanzi UI snapshot from Artifactory.....	27
<b>Figure 9.</b> A snippet of the Materials and textures tutorial .....	29
<b>Figure 10.</b> Bloom tutorial snippet.....	30
<b>Figure 11.</b> Button tutorial snippet .....	31
<b>Figure 12.</b> Dynamic layout tutorial snippet .....	32
<b>Figure 13.</b> Getting started tutorial snippet .....	33
<b>Figure 14.</b> Indicator tutorial snippet .....	34
<b>Figure 15.</b> Keyframe animations tutorial snippet .....	35
<b>Figure 16.</b> Rotation tutorial snippet .....	36
<b>Figure 17.</b> State managers tutorial snippet.....	37
<b>Figure 18.</b> Stencil tutorial snippet.....	38
<b>Figure 19.</b> Toggle button tutorial snippet.....	39
<b>Figure 20.</b> List of UI Tests from Visual Studio .....	40
<b>Figure 21.</b> Tutorial walkthrough tests run on Jenkins .....	40
<b>Figure 22.</b> Internal documentation on tutorial walkthrough UI tests.....	41
<b>Figure 23.</b> Tutorial Walkthrough tasks planned in Hansoft.....	42
<b>Figure 24.</b> Task has acceptance criteria as requirement.....	42
<b>Figure 25.</b> TutorialWalkthroughTests.cs location .....	43
<b>Figure 26.</b> Some steps from the Toggle button tutorial .....	44
<b>Figure 27</b> The first step of the Rotation tutorial .....	47
<b>Figure 28.</b> The second step of the Rotation tutorial.....	48
<b>Figure 29.</b> A snippet of Bloom tutorial walkthrough UI test pull request ..	49
<b>Figure 30.</b> Fail screenshot verification in Rotation tutorial.....	50

**Figure 31.** Highlighted difference screenshot ..... 51  
**Figure 32.** Jenkins UI test failure screenshot ..... 52  
**Figure 33.** A bug ticket for Rotation tutorial walkthrough UI test from Hansoft  
..... 52

**LIST OF ABBREVIATIONS AND ACRONYMS**

2D	Two-dimensional
3D	Three-dimensional
API	Application Programming Interface
C#	C-Sharp programming language
CD	Continuous deployment
CI	Continuous integration
CLI	Command-line interface
CLR	Common Language Runtime
CPU	Central processing unit
CUIT	Coded UI Test
FCL	Framework class library
GDI	Graphics device interface
GPU	Graphics processing unit
GUI	Graphical user interface
HMI	Human Machine Interface
HUD	Head-Up Display
IDE	Integrated development environment
KZB	Kanzi binary
KZPROJ	Kanzi project

MIL	Machine independent language
PR	Pull request
QA	Quality Assurance
SWT	Standard Widget Toolkit
UAT	User Acceptance Testing
UI	User Interface
WPF	Windows Presentation Foundation



## 1 INTRODUCTION

Nowadays, design software for user interfaces is getting more and more popular. These productive applications help designers to create professional 2D and 3D projects efficiently. Therefore, Rightware – a company behind the Kanzi family of tools and services for the design and development of advanced digital user interfaces, always focuses on the quality of their products and user documentation.

Rightware was founded in 2009. The Kanzi product family includes a visual studio that inspires designers, a powerful 3D runtime for superior graphics performance, and an innovative connectivity platform. Rightware is headquartered in Finland and has a presence in the United States, United Kingdom, Germany, Italy, China, South Korea, and Japan. Kanzi is the market-leading UI tool trusted by over 50 automotive brands across the globe. (Rightware, 2020)

At Rightware, it is important to ensure the quality of the product and compliance or the work that produced it. Therefore, continuous integration and development are well-applied among the development process, along with daily automated test runs to ensure each product build is working as expected. For each release, there always are automated and manual tests which ensure the quality of the product before delivering to the customers.

Rightware also provides their users with professionally written user documentation which allow the users to get familiar with Kanzi Studio and feel comfortable in using the product. Kanzi documentations provide users with all the guides and fundamentals in using Kanzi Studio, along with an efficient list of tutorial walkthroughs from beginner to advanced level. The documentation delivers a handful of applied knowledge which helps users to use Kanzi Studio more proficiently. Therefore, it is also important to always

assure the quality of documentation for each release or when new documentation content is added.

Within the scope of this thesis, a UI automated tests list consisting of 10 tutorial walkthroughs is implemented. The tests will run automatically for each build. These UI automated tests will help to ensure that the steps described in Kanzi documentations are still valid and that Kanzi Studio functionalities are verified to be working. Moreover, in the future, the UI automated walkthrough tests can also be used to automatically create video-based tutorials for the Kanzi Documentation team, to provide better instructions and learning experiences for Kanzi users.

## **2 WPF APPLICATION**

### **2.1 .NET Framework in General**

The .NET Framework (articulated as "dot net") is a software framework created by Microsoft that runs principally on Microsoft Windows. It incorporates an enormous class library called Framework Class Library (FCL) and gives language interoperability across a few programming languages. Programs are written for .NET Framework execute in a software environment named the Common Language Runtime (CLR). The CLR is an application virtual machine that offers services such as security, memory management, and exception handling.

FCL provides UI, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. Developers produce programming by consolidating their source code with .NET Framework and different libraries. The framework is intended to be utilized by most new applications made for the Windows platform. Microsoft additionally delivers an integrated development environment (IDE) for .NET programming called Visual Studio.

### **2.2 Microsoft Visual Studio**

Microsoft Visual Studio is an IDE that is used to develop computer programs, as well as websites and mobile applications.

Visual Studio incorporates a code editor supporting IntelliSense as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a code profiler, designer for building GUI applications, website designer, class designer, and database diagram designer. It allows users to use plugins that extend the functionality at pretty much every level—including adding support for source control systems, such as Subversion and Git. It is also easy

to add other toolsets for software development lifecycle for example, the Azure DevOps customer Team Explorer.

Visual Studio supports various programming languages, such as C, C++, C++/CLI, Visual Basic .NET, C#, F#, JavaScript, TypeScript, etc.

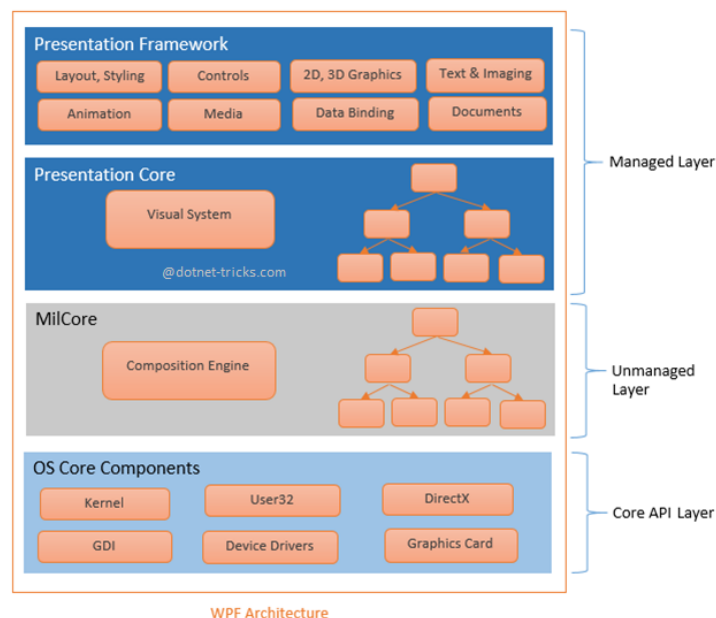
## 2.3 WPF Application

### 2.3.1 Definition

Windows Presentation Foundation (WPF) is a free and open-source graphical subsystem created by Microsoft for delivering UIs in Windows-based applications. WPF, recently known as "Avalon", was at first delivered as a component of .NET Framework 3.0 in 2006.

### 2.3.2 Architecture

The major components of WPF architecture are shown in Figure 1.



**Figure 1.** WPF Architecture (Chauhan, 2018).

As seen in Figure 1, the managed layer consists of Presentation Framework and Presentation Core.

Presentation Framework provides the required functionalities that are needed to build the WPF applications, such as controls, data bindings, styling, shapes, media, documents, and annotations.

Presentation Core is the home for WPF Visual System and provides classes for creating application visual trees. The Visual System creates a visual tree that contains applications Visual Elements and rendering instructions.

The unmanaged layer layer is also called Milcore or Media Integration Library Core. MilCore is written in an unmanaged code to enable tight integration with DirectX (Chauhan, 2018).

DirectX engine is the underlying technology used in WPF to display all graphics, allowing for efficient hardware and software rendering. MIL has a Composition System that receives rendering instructions from Visual System and translates them into data that can be understood by DirectX to render the user interface.

This Core API layer has OS core components, such as Kernel, User32, GDI, Device Drivers, and Graphic cards. These components are used by the application to access low-level APIs. User32 manages memory and process separation. (Chauhan, 2018)

## **2.4 Kanzi UI Application**

Kanzi UI is a cross-platform HMI tool combining high-end 3D graphics with the features and long-term support expected of an automotive-grade UI framework for digital instrument clusters, infotainment systems, HUDs, mobile devices, and more. With Kanzi UI, automakers have complete control

over the user experience and visual brand identity, from prototype to series production. Kanzi UI consists of Kanzi Studio, Kanzi Runtime, and Kanzi Platform packages.

Kanzi Studio is a Windows visual modeling tool used to create UI definitions processed by Kanzi Runtime. Kanzi Studio is written in C# programming language applying the WPF framework.

Kanzi Runtime is a graphics UI engine intended to execute the whole UI on the target device and provide user interactions. Kanzi Runtime is written in C++ programming language.

Kanzi Runtime for embedded targets is delivered to customers as a Platform Package, which is a zip archive that contains all the required headers and pre-built libraries that are needed to build a Kanzi Runtime application for a specific platform. They also include one example application project (KZB player) and build scripts.

## 3 SOFTWARE TESTING

### 3.1 Software Testing Definition and Types of Software Testing

#### 3.1.1 Definition of Software Testing

Software Testing is a strategy to check whether the actual software matches the requirements and to guarantee that the software product is released without critical defects. It includes the execution of system parts using manual or automated testing tools to assess at least one or more of the packed features. The motivation behind software testing is to identify errors, gaps, or missing requirements in contrast to actual requirements. (Guru99, 2021c)

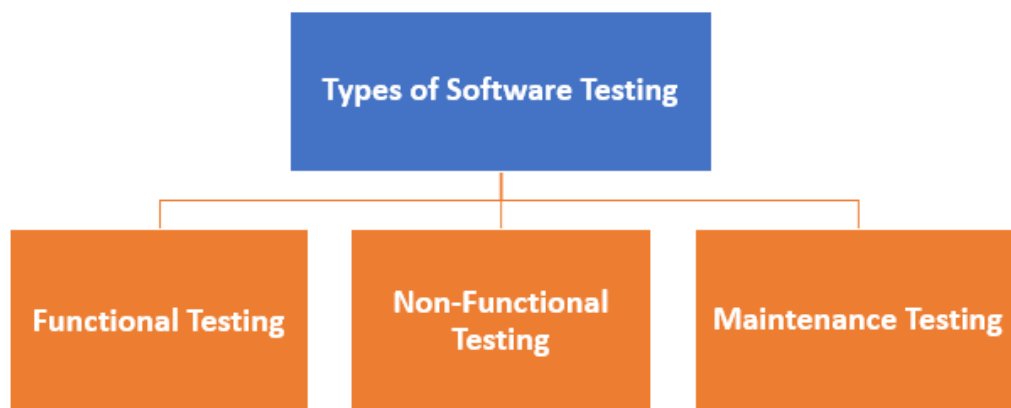
Software testing can be planned, implemented, and executed using a Software Testing lifecycle. Figure 2 provides an example of the activities that are necessary for testing a software product.



**Figure 2.** The Software Testing Lifecycle (Peforce, 2018)

### 3.1.2 Different Types of Software Testing

There are three types of Software Testing which can be found in Figure 3.



**Figure 3.** Types of Software Testing (Guru99 2021c).

Functional Testing includes Unit Testing, Integration Testing, Smoke Test, Localization, Globalization Testing, and UAT testing.

Non-Functional Testing can be listed as Performance, Endurance, or Scalability Testing.

Maintenance Testing includes Regression tests and tests that should be run when some maintenance was made on the software.

### 3.2 Need of Software Testing

Software Testing brings many benefits to the software development process, which are listed below.

- Cost-effectiveness is one of the significant benefits of software testing. Testing any IT project on time helps saving costs for the long



term. It is because if the bugs are identified in the prior phase of software testing, it costs less to fix.

- Security is the most vulnerable and sensitive advantage of software testing. Customers are searching for a reliable software product so testing helps in eliminating security risks and issues prior.
- Product quality is a fundamental necessity of any software product. Testing guarantees a quality product is delivered to clients.
- Customer satisfaction. The primary goal of any software product is to offer fulfillment to its clients. Testing guarantees the best user experience.

### **3.3 Software Testing at Rightware**

The execution of Software Testing will follow the Rightware feature project guidelines in line with project deliveries which are in sync with the release cycle. The scrums teams perform the test execution in the following release phases with QA support:

- Development sprint level: for example unit testing, integration testing.
- Feature branch level: for example integration, regression testing.
- Master /Target branch level: system, security, exploratory, and UAT testing,

At the same time, based on the risk assessment of the project, the criteria for each test phase will be measured. These criteria may vary project-wise and the release cycle will follow the quality gates criteria on every level before deploying in production.

## **4 UI AUTOMATION TESTING**

### **4.1 Definition of UI**

The term "UI" stands for "user interface". The user interface is the graphical design of an application. It contains the buttons users click on, the content they read, the pictures, sliders, text fields, and all of the elements the users interact with. This includes screen layout, transitions, interface animations, and each micro-interaction.

### **4.2 UI Automation Testing**

When performing UI tests, testers make sure that all the logic, UI feature, or flow of activities work as expected. They focus on validating the click of a button, data entry, navigation, calculation of values, and different functionalities utilized for user interaction. (Perforce, 2021)

UI automation testing is a method where these testing processes are performed using an automation tool. Rather than having testers navigate the application to check features and activity flows manually, test scripts are written to continuously verify each version of the software.

### **4.3 UI Automated Testing at Rightware**

UI Automated tests at Rightware are used to verify that the newly added features meet specified requirements and do not break existing functionalities.

UI Tests are executed on a built version of Kanzi Studio and not on a user-shippable installer. UI SDK tests are running on installed Kanzi Studio which is the version that users have on their machines.

Features that will be verified in the scope of UI Testing are:

- Kanzi Studio.
- Kanzi Runtime as part of Kanzi Studio preview.
- Kanzi Platform Packages for Windows and Android.
- Completed Tutorials validity on Windows and Android.
- Examples validity on Windows and Android.
- Project backward compatibility against previous Kanzi UI releases.

#### **4.4 UI Automation Testing Tools Used at Rightware**

UI Testing at Rightware is done applying UI Automated testing framework specific for Windows WPF applications.

##### **4.4.1 Microsoft UI Automation**

Microsoft UI Automation is an accessibility framework for Windows. It addresses the necessities of assistive technology products and automated test frameworks by giving programmatic access to data about the user interface. Similarly, UI Automation empowers application developers to make their products accessible.

Microsoft UI Automation allows one to access, identify and manipulate the user interface of an application. It enables assistive technology products, such as screen readers, to provide information about the UI to end-users and to manipulate the UI by means other than standard input. UI Automation also allows automated test scripts to interact with the UI. (Microsoft Docs 2018)

#### **4.4.2 Test Stack White**

White is a framework for automating rich client applications based on Win32, WinForms, WPF, Silverlight, and SWT (Java) platforms. It is .NET based and does not require the use of any proprietary scripting languages. Tests/automation programs using White can be written with whatever .NET language, IDE, and tools already used. White provides a consistent object-oriented API, hiding the complexity of Microsoft's UIAutomation library (on which White is based) and windows messages. (TestStack.White, 2021)

#### **4.4.3 Microsoft CodedUI**

Coded UI Test (CUIT) is an automated test that drives applications through its user interface (UI). Hence, the name Coded UI Test (CUIT). This test involves functional testing of the UI controls. It checks the functionality of the whole application, including the user interface. It is also used to automate an existing manual test. (Guru99, 2021b)

## 5 KANZI UI DOCUMENTATION

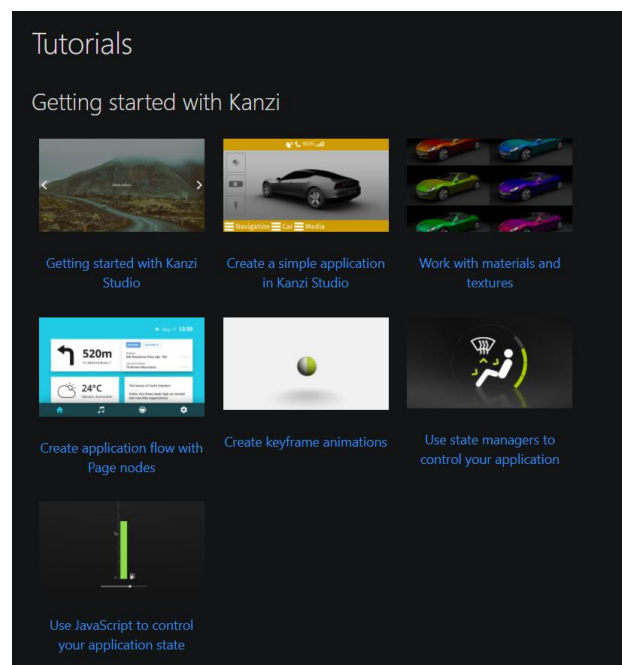
### 5.1 About Kanzi UI Documentation

Kanzi UI Documentation consists of instructions on how to get started with Kanzi UI, Tutorials ranging from beginner to advanced level, Fundamentals used in Kanzi UI, and everything that is needed to get started or get advanced in using Kanzi UI.

### 5.2 Kanzi UI Tutorial Documentation

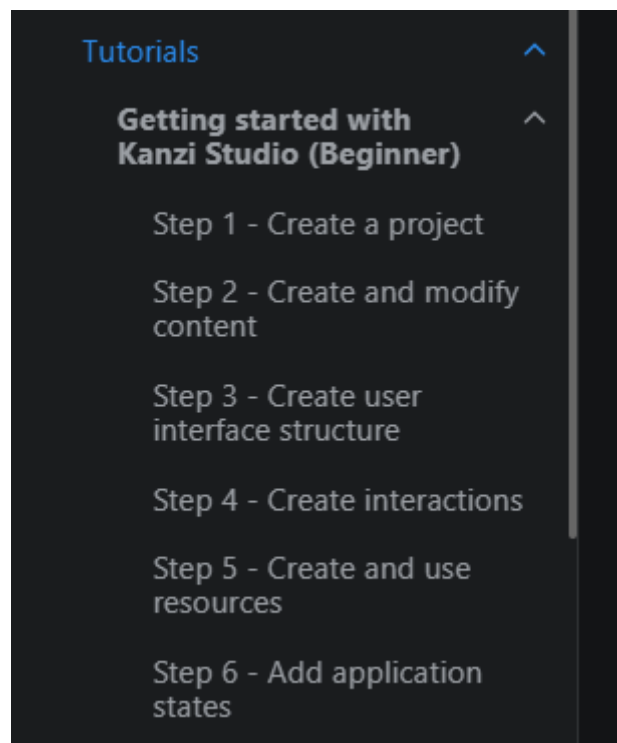
Kanzi UI Documentation contains various tutorials that are documented to help beginners and intermediate users get more familiar with user Kanzi UI.

A screenshot of the Kanzi UI Documentation tutorial page can be seen in Figure 4.



**Figure 4.** Screenshot of Kanzi Documentation Tutorials.

Each Kanzi UI Tutorial contains a number of steps that vary for different types of tutorials. Figure 5 shows the steps which are included in the “Getting started with Kanzi Studio” tutorial.



**Figure 5.** Screenshot of steps in a tutorial

Tutorials are documented along with screenshots to provide a better understanding of each step. See Figure 6.

## Step 2 - Create and modify content

In this step you learn how to use the **Node Tree** window to create content and the **Properties** window to modify the content in your Kanzi Studio project.

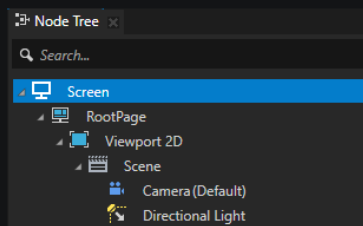
### Create content in the **Node Tree** window

You create the structure of your Kanzi application in the **Node Tree** window. You can find the **Node Tree** window on the top-left side of the Kanzi Studio interface.

In the **Node Tree** window the **Screen** contains the node tree of your application. When you start your Kanzi application on a device Kanzi loads into the device memory all nodes in the node tree.

The node tree is constructed from nodes that display content (for example, **Image** node) and implement logic (for example, **Button 2D** node). The same node tree supports 2D and 3D nodes and provides the means to connect them.

When you work on your project in Kanzi Studio you can see the content of the node tree in the **Preview** window.



To create content in the **Node Tree** window:

1. In the **Node Tree** window press **Alt** and right-click the **RootPage** node and select **Text Block 2D**.

#### Tip

When you hold down the **Alt** key and right-click an item, you access the create menu which shows the types of items that you can create in the selected context.

Use the **Text Block** nodes to show a small amount of text in your application.

**Figure 6.** Screenshot of a part of Kanzi tutorial

A documented tutorial varies from one to five big steps that includes few small steps within it. Therefore, the length of each tutorial might be different for different instruction.

## 6 TOOLS FOR UI TEST AUTOMATION

### 6.1 Kanzi UI Test Framework

Kanzi UI Test Framework is a testing framework that Rightware developed and used to test Kanzi Studio. It contains KanziToolAutomatedUITests solution and UITestFramework NuGet package.

“NuGet is an essential tool for any modern development platform is a mechanism through which developers can create, share, and consume useful code. Often such code is bundled into "packages" that contain compiled code (as DLLs) along with other content needed in the projects that consume these packages.” (Microsoft Docs, 2021)

UITestFramework NuGet package is verified and deployed to the Rightware internal NuGet Server using a Continuous Integration (CI) pipeline in the scope of pull request gate.

### 6.2 Continuous Integration

Continuous integration (CI) is the practice of automating the integration of code changes from multiple contributors into a single software project. It is a primary DevOps best practice, allowing developers to frequently merge code changes into a central repository where builds and tests are then run. (Atlassian, 2021)

At Rightware, CI means building and testing each Github pull request (PR) made against a set of predefined branches, and blocking merging of PRs that contain errors. The build steps of CI builds are selected to be representative of the most common configurations while providing fast results for iterative development.



### **6.3 Continuous Delivery**

Continuous delivery is a software development practice where code changes are automatically prepared for a production release. A pillar of modern application development, continuous delivery expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage. When properly implemented, developers will always have a deployment-ready build artifact that has passed through a standardized test process. (Freeman, 2019)

At Rightware, CD means building on every merge made to specified branches and storing the build artifacts in Artifactory. CD builds produce a complete releasable set of product packages which could be promoted to a release by Release Management and also run an extended battery of tests on the products.

### **6.4 Jenkins**

Jenkins is an open-source automation server that runs in servlet containers, such as Apache Tomcat. It helps to automate the processes of software development, for example building, testing, and deploying. Jenkins helps to facilitate Continuous Delivery and Continuous Integration.

At Rightware, Jenkins is used widely to automate different kinds of processes. There are multiple pipelines used for Kanzi UI like Kanzi UI – Studio UI Tests, Kanzi UI – CD build, etc. See Figure 7 for Rightware’s Studio UI Test Jenkins build job.

The screenshot shows the Jenkins interface for a build job. The top navigation bar includes 'Jenkins', 'All', 'Kanzi UI - Studio UI Tests', 'master', and '#1174'. The main content area is titled 'Build #1174 (Sep 3, 2021 4:26:00 PM)'. Below the title, there is a list of build artifacts with columns for filename, size, and a 'view' link. The artifacts include files like 'FailedFilter.txt', 'buildagent\_RW0160\_2021-09-03\_21\_30\_01.trx', and 'UITESTS-1174-logs.zip'. Below the artifacts, there is a 'Changes' section with a list of commit messages and links to 'detail' and 'githubweb'.

Build Artifacts	Size	View
FailedFilter.txt	78 B	view
buildagent_RW0160_2021-09-03_21_30_01.trx	130.81 KB	view
buildagent_RW0160_2021-09-03_22_09_51.trx	577.79 KB	view
buildagent_RWWD0005_2021-09-03_20_58_39.trx	621.78 KB	view
buildagent_RWWD0010_2021-09-03_20_47_05.trx	865.57 KB	view
buildagent_RWWD0021_2021-09-04_07_45_45.trx	154.33 KB	view
buildagent_RWWD0023_2021-09-03_21_00_31.trx	943.70 KB	view
buildagent_RWWD0026_2021-09-03_22_06_44.trx	198.16 KB	view
buildagent_WINTEST01_2021-09-03_22_56_59.trx	186.96 KB	view
buildagent_WINTESTVM03_2021-09-03_20_53_37.trx	39.49 KB	view
UITESTS-1174-logs.zip	824.52 KB	view

**Changes**

354107. Fixed issue with removing FontFiles during merge ([detail](#) / [githubweb](#))
354107. FontFiles merge issue. Added unit test ([detail](#) / [githubweb](#))
354107. Minor fix. Comments ([detail](#) / [githubweb](#))
354107. Addressed PR comments ([detail](#) / [githubweb](#))
354107. Addressed PR comments ([detail](#) / [githubweb](#))
- [367079] Traceability shows that ui:node:scroll\_view is not used by any ([detail](#) / [githubweb](#))
- Cherry-pick 3.9 Beta1 Android fixes (#9386) ([detail](#) / [githubweb](#))

**Figure 7.** Rightware's Studio UI Test Jenkins build job

## 6.5 Artifactory

Artifactory is a product by JFrog that serves as a binary repository manager. The binary repository is a natural extension to the source code repository, in that it will store the outcome of the build process, often denoted as artifacts. Most of the time one would not use the binary repository directly but through a package manager that comes with the chosen technology.

At Rightware Oy, Artifactory is used to store for example Kanzi-related builds, UI test builds, and Kanzi Documentations. See Figure 8.

Name	Last modified	Size
...		
<a href="#">components/</a>	02-Sep-2021 22:54	-
<a href="#">plugins/</a>	03-Sep-2021 01:54	-
<a href="#">tests/</a>	02-Sep-2021 23:43	-
<a href="#">build_info.json</a>	03-Sep-2021 01:52	21.61 KB
<a href="#">ec2_config.json</a>	02-Sep-2021 22:54	7.54 KB
<a href="#">Kanzi-3.9.0-dev-integrity_nvidia_aarch64-es3-freetype-static.zip</a>	03-Sep-2021 00:39	330.74 MB
<a href="#">Kanzi-3.9.0-dev-integrity_rcar_rvm_aarch64-es3-freetype-static.zip</a>	03-Sep-2021 00:22	330.98 MB
<a href="#">Kanzi-3.9.0-dev-linux_imx6_armhf-es3-freetype-dynamic.zip</a>	03-Sep-2021 00:53	148.40 MB
<a href="#">Kanzi-3.9.0-dev-linux_imx6_armhf-es3-freetype-static.zip</a>	03-Sep-2021 00:53	269.75 MB
<a href="#">Kanzi-3.9.0-dev-linux_imx6_armhf-es3-itype-dynamic.zip</a>	03-Sep-2021 00:53	147.90 MB
<a href="#">Kanzi-3.9.0-dev-linux_imx6_armhf-es3-itype-static.zip</a>	03-Sep-2021 00:54	270.29 MB
<a href="#">Kanzi-3.9.0-dev-linux_wayland_aarch64-es3-freetype-dynamic.zip</a>	03-Sep-2021 00:54	153.70 MB
<a href="#">Kanzi-3.9.0-dev-linux_wayland_aarch64-es3-itype-dynamic.zip</a>	03-Sep-2021 00:54	150.51 MB
<a href="#">Kanzi-3.9.0-dev-linux_wayland_aarch64-es3-freetype-dynamic.zip</a>	03-Sep-2021 01:00	160.86 MB
<a href="#">Kanzi-3.9.0-dev-linux_wayland_aarch64-es3-freetype-static.zip</a>	03-Sep-2021 01:01	290.82 MB
<a href="#">Kanzi-3.9.0-dev-linux_wayland_aarch64-es3-itype-dynamic.zip</a>	03-Sep-2021 01:00	157.62 MB
<a href="#">Kanzi-3.9.0-dev-linux_wayland_aarch64-es3-itype-static.zip</a>	03-Sep-2021 01:01	291.27 MB
<a href="#">Kanzi-3.9.0-dev-linux_wayland_x86_64_neusoft-es3-freetype-dynamic.zip</a>	03-Sep-2021 01:02	296.58 MB
<a href="#">Kanzi-3.9.0-dev-linux_wayland_x86_64_neusoft-es3-itype-dynamic.zip</a>	03-Sep-2021 01:02	295.97 MB
<a href="#">Kanzi-3.9.0-dev-linux_wsegl_aarch64-es3-freetype-dynamic.zip</a>	03-Sep-2021 01:12	160.30 MB
<a href="#">Kanzi-3.9.0-dev-linux_wsegl_aarch64-es3-freetype-static.zip</a>	03-Sep-2021 01:01	290.11 MB
<a href="#">Kanzi-3.9.0-dev-linux_x11_glx-glx-freetype-dynamic-gcc5.zip</a>	03-Sep-2021 00:59	290.77 MB
<a href="#">Kanzi-3.9.0-dev-linux_x11_glx-glx-freetype-static-gcc5.zip</a>	03-Sep-2021 01:01	385.21 MB
<a href="#">Kanzi-3.9.0-dev-linux_x11_glx-glx-itype-dynamic-gcc5.zip</a>	03-Sep-2021 00:58	290.21 MB
<a href="#">Kanzi-3.9.0-dev-linux_x11_glx-glx-itype-static-gcc5.zip</a>	03-Sep-2021 01:00	385.83 MB
<a href="#">Kanzi-3.9.0-dev-Monotype_Preview.zip</a>	03-Sep-2021 00:11	5.18 MB
<a href="#">Kanzi-3.9.0-dev-qnx700_screen_aarch64-es3-freetype-dynamic.zip</a>	03-Sep-2021 01:06	150.32 MB
<a href="#">Kanzi-3.9.0-dev-qnx700_screen_aarch64-es3-freetype-static.zip</a>	03-Sep-2021 01:05	276.69 MB
<a href="#">Kanzi-3.9.0-dev-qnx700_screen_aarch64-es3-itype-dynamic.zip</a>	03-Sep-2021 01:05	149.75 MB
<a href="#">Kanzi-3.9.0-dev-qnx700_screen_aarch64-es3-itype-static.zip</a>	03-Sep-2021 01:05	277.25 MB
<a href="#">Kanzi-3.9.0-dev-qnx700_screen_arm-es3-freetype-dynamic.zip</a>	03-Sep-2021 01:04	148.19 MB
<a href="#">Kanzi-3.9.0-dev-qnx700_screen_arm-es3-freetype-static.zip</a>	03-Sep-2021 01:08	276.89 MB
<a href="#">Kanzi-3.9.0-dev-qnx700_screen_arm-es3-itype-dynamic.zip</a>	03-Sep-2021 01:07	147.67 MB
<a href="#">Kanzi-3.9.0-dev-qnx700_screen_arm-es3-itype-static.zip</a>	03-Sep-2021 01:06	277.41 MB
<a href="#">Kanzi-3.9.0-dev-qnx700_screen_x86_64-es3-freetype-dynamic.zip</a>	03-Sep-2021 01:09	151.86 MB
<a href="#">Kanzi-3.9.0-dev-qnx700_screen_x86_64-es3-freetype-static.zip</a>	03-Sep-2021 01:06	276.38 MB
<a href="#">Kanzi-engine_binary_distributable-3.9.0-dev.zip</a>	03-Sep-2021 01:04	1.44 GB
<a href="#">Kanzi-engine_source_distributable-3.9.0-dev.zip</a>	03-Sep-2021 01:04	6.30 MB
<a href="#">Kanzi-Obfuscation_Maps-3.9.0-dev.zip</a>	03-Sep-2021 01:05	165.25 KB
<a href="#">KanziStudio-v3.9.0.2231-dev.exe</a>	03-Sep-2021 01:02	1.48 GB
<a href="#">KanziStudio-v3.9.0.2231-dev.exe_filelist.txt</a>	03-Sep-2021 01:03	2.89 MB
<a href="#">KanziUIDocumentation-3.9.0.2231-dev.zip</a>	03-Sep-2021 01:03	338.70 MB
<a href="#">kzb_player-3.9.0-dev-integrity_nvidia_aarch64-static-freetype-es3.zip</a>	03-Sep-2021 00:53	48.13 MB
<a href="#">kzb_player-3.9.0-dev-integrity_rcar_rvm_aarch64-static-freetype-es3.zip</a>	03-Sep-2021 00:31	39.96 MB

**Figure 8.** Kanzi UI snapshot from Artifactory

Artifactory is used to store the UI test result for around a good amount of time before they are being removed to save space for newly builds and test runs.

## **7 IMPLEMENTATION OF AUTOMATED TUTORIAL WALKTHROUGH UI TEST**

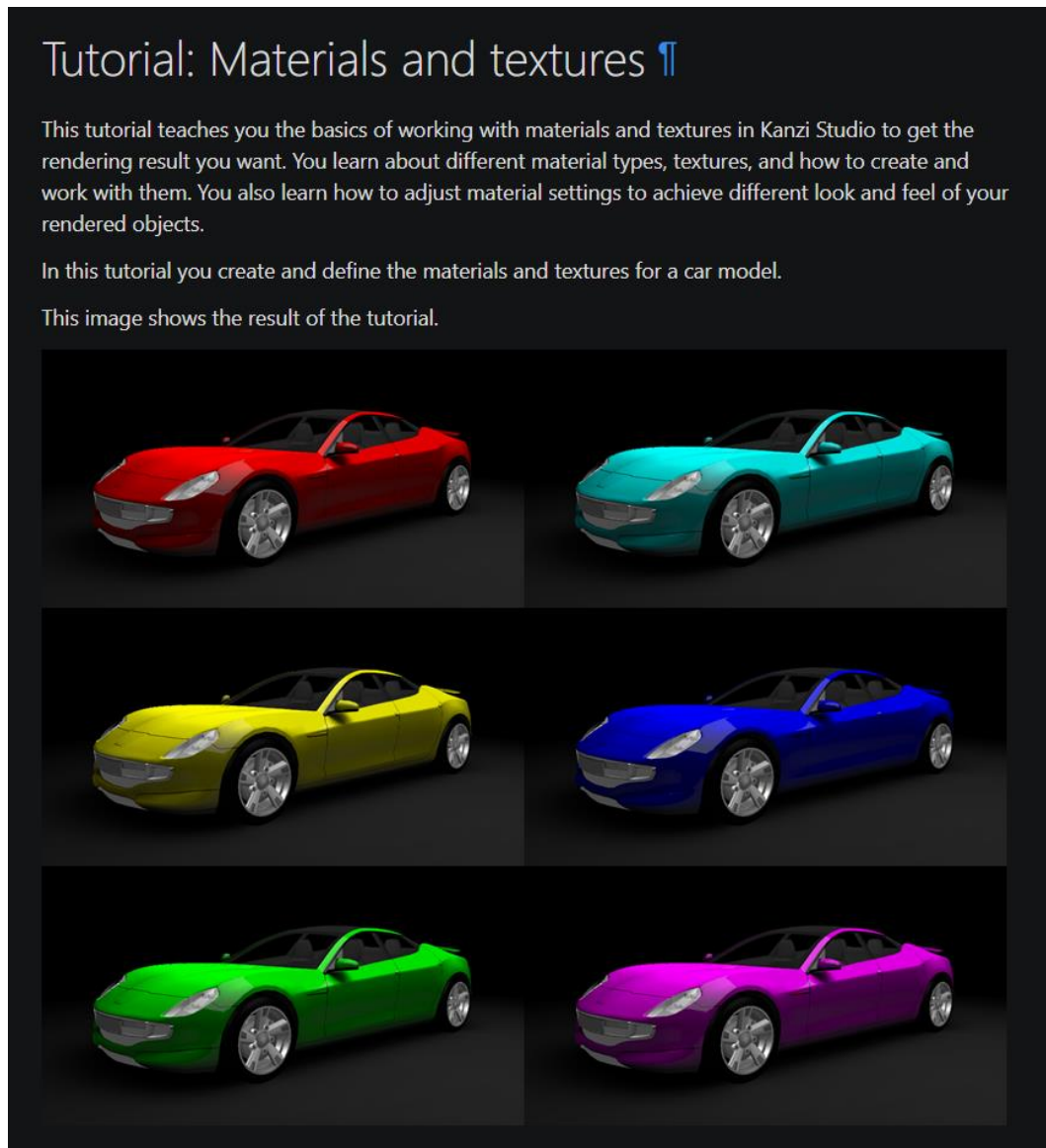
### **7.1 Benefits of tutorial walkthrough UI test**

To provide designers and developers a better look at how Kanzi UI works, Rightware delivers a dedicated Kanzi Documentation for it. Kanzi Documentation provides users with needed information when using the Kanzi UI along with well-documented tutorials that vary from beginner to advanced level. There are in total thirty-three documented tutorials, which is a big number to test manually every tutorial, therefore, automation comes in.

We wanted to create automated tutorial walkthrough UI tests which run for every Kanzi build to make sure that the steps provided by Kanzi Documentation are still valid in the latest build, and that Kanzi UI is still working as expected.

Before the automated walkthrough UI tests were implemented, we usually performed manual testing to walkthrough some selected tutorials for every release. Now with the automated walkthrough UI tests running for each CD build, we reduced the resource in manually testing the tutorial and the cost of bugs by acknowledging them earlier and come with an earlier fix.

On the other hand, we also want to use these automated tutorial walkthrough UI tests to automatically create video-based tutorials in the future for Kanzi Documentation. Currently, we are using mostly text-based tutorials. See Figure 9.

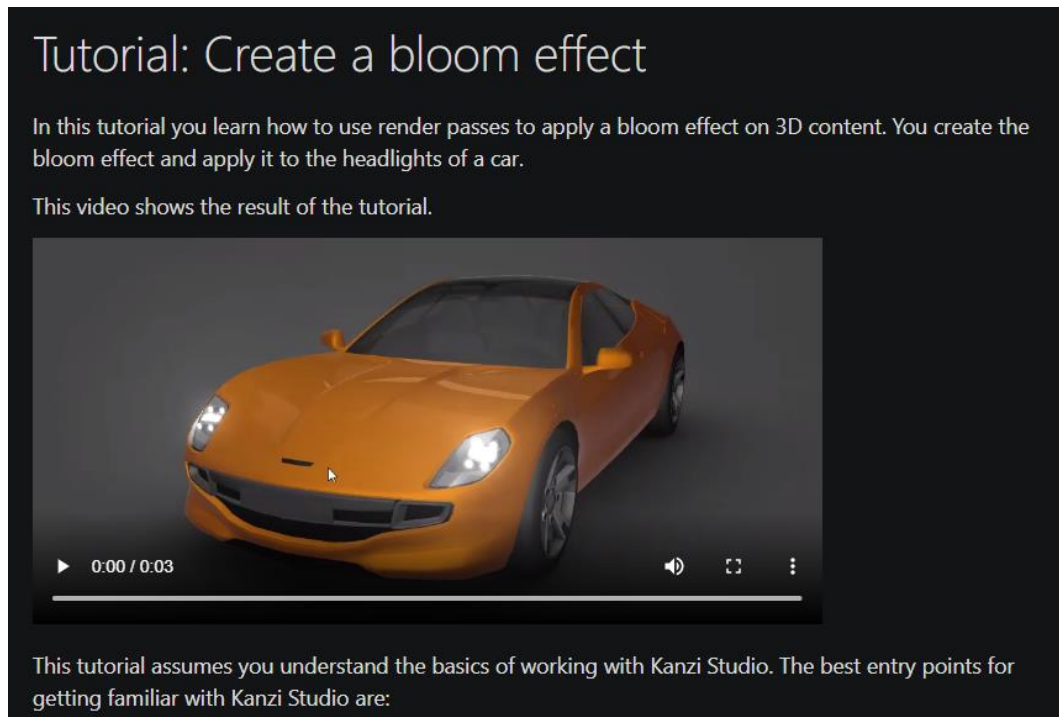


**Figure 9.** A snippet of the Materials and textures tutorial

## 7.2 List of tutorials implemented

In the scope of this thesis, the list of ten automated tutorial walkthrough UI tests was implemented. The tutorials which are now automatically walkthrough by UI test are:

**Bloom Tutorial** instructs users on how to use the Kanzi render pass to apply a bloom effect on 3D content. See Figure 10.



**Figure 10.** Bloom tutorial snippet

The test for this tutorial verifies that users can create a bloom effect on 3D content, specifically a car in this case. The test is required to use render passes as described in the tutorial steps. After following the tutorial steps, the final application is verified by taking screenshot differences of Kanzi Preview.

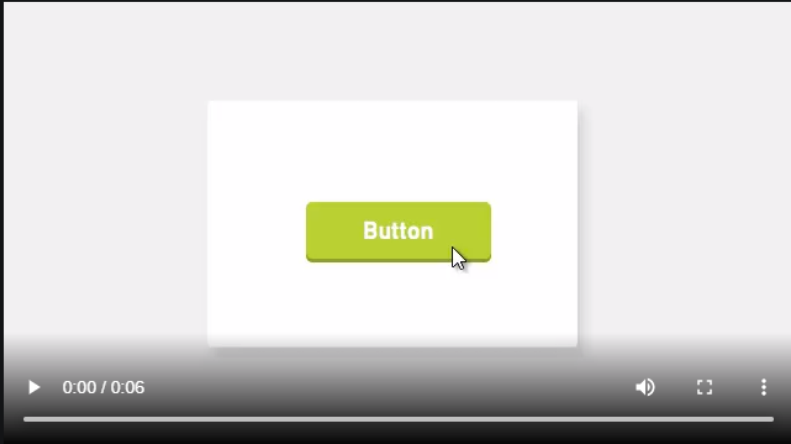
**Button tutorial** instructs users on how to create a Button in Kanzi Studio. See Figure 11.

## Tutorial: Creating a button

In this tutorial you learn how to create a button. Kanzi allows you to create buttons using only triggers and actions. However, to create a robust button use the Kanzi state manager and control the state of the button using the **Is Down** property.

The difference between the two approaches is that when you use the state manager you always know in which state the button is. For example, when a user clicks a button created with triggers and actions, the action sends the message, but your application is not aware of the state of the button. When you use the state manager, you can always check which state the button is in and even revert the state when needed.

This video shows the result of the tutorial.



This tutorial assumes you understand the basics of working with Kanzi Studio. The best entry points for getting familiar with Kanzi Studio are:

**Figure 11.** Button tutorial snippet

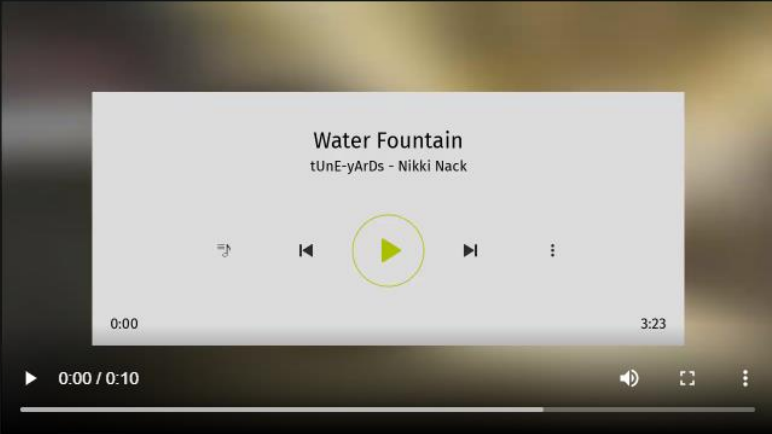
The test for this tutorial verifies that users can create a button using the state manager in Kanzi Studio. After following the tutorial steps, the final application is verified by taking screenshot differences of Kanzi Preview.

**Dynamic layout tutorial** instructs on how to create a user interface that responds to the changes of the device screen resolution. See Figure 12.

## Tutorial: Making applications with dynamic layout

In this tutorial you learn how to create user interfaces that respond to the changes of the device screen resolution. A user interface with a dynamic layout looks good and is easy to use regardless of the device and its screen resolution.

This video shows the result of the tutorial.



This tutorial assumes you understand the basics of working with Kanzi Studio. The best entry points for getting familiar with Kanzi Studio are:

**Figure 12.** Dynamic layout tutorial snippet

The test for this tutorial verifies that users can create an application that is responsive to different screen sizes using Kanzi Studio. After following the tutorial steps, the final application is verified by taking screenshot differences of Kanzi Preview.

**Getting started tutorial** instructs users on how to create a simple Kanzi application. See Figure 13.

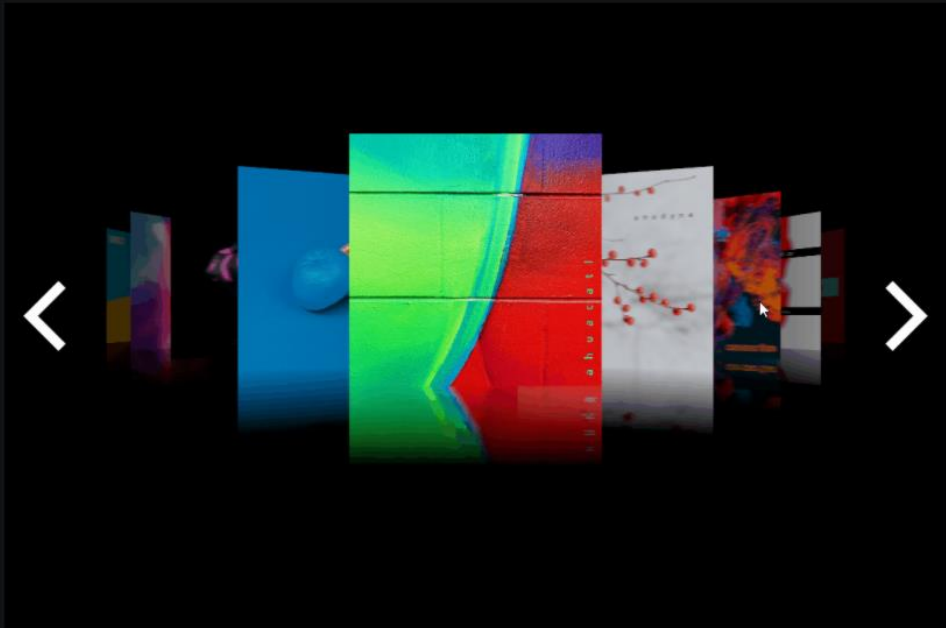


## Tutorial: Getting started with Kanzi Studio

In this tutorial you learn about the basic Kanzi Studio features by creating a simple Kanzi application. If you want to learn how to use Kanzi Studio, this tutorial is the right place to start.

After completing this tutorial you can focus on learning how to use specific Kanzi features by completing other tutorials.

This video shows the result of the tutorial.



Kanzi consists of two main components Kanzi Studio and Kanzi Engine:

- Kanzi Studio is a content creation tool. You can import to a Kanzi Studio project 2D and 3D content, create prototypes, work on composition, interaction, and interface design, and export the production binary files.
- Kanzi Engine is a graphics and user interface execution environment for the binary files that you generate from a Kanzi Studio project. Kanzi Engine supports leading operating systems and hardware platforms out of the box. This allows engineers to focus on application development, rather than on optimization and integration.

**Figure 13.** Getting started tutorial snippet

The test for this tutorial verifies that users can create a Kanzi application consisting of different pages with buttons to switch between the pages. Each important step is verified using screenshot verification. After following the tutorial steps, the final application is verified by taking screenshot differences of Kanzi Preview.

**Indicator tutorial** instructs users on how to create a turn indicator and control with a property using Kanzi state managers. This tutorial has video-based instructions. See Figure 14.

## Tutorial: Creating cluster indicators

In this tutorial you learn how to create a turn indicator and control it with a property using the Kanzi state manager.

Complete the tutorial by watching the video, or by following the written instructions.



This tutorial assumes you understand the basics of working with Kanzi Studio. The best entry points for getting familiar with Kanzi Studio are:

**Figure 14.** Indicator tutorial snippet

The test for this tutorial verifies that users can control the state of an image with a property using Kanzi state manager. After following the tutorial steps, the final application is verified by taking screenshot differences of Kanzi Preview.

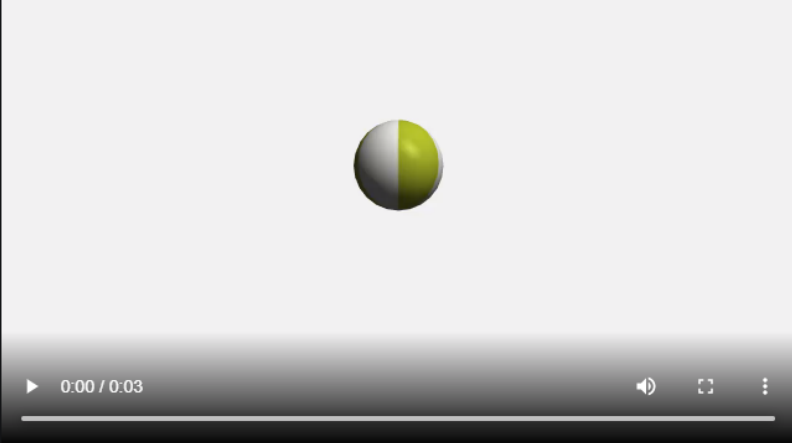
**Keyframe animations tutorial** shows users how to create keyframe animations using Kanzi Studio. See Figure 15.

## Tutorial: Create keyframe animations

This tutorial teaches you how to create keyframe animations in Kanzi Studio. Kanzi Studio provides a flexible system for animating your applications.

In this tutorial you create a one-second keyframe animation of a bouncing beach ball using the **Animation Clip Editor** in Kanzi Studio.

This video shows the result of the tutorial.



This tutorial assumes you understand the basics of working with Kanzi Studio. The best entry points for getting familiar with Kanzi Studio are:

**Figure 15.** Keyframe animations tutorial snippet

The test for this tutorial verifies that users can create keyframe animations in Kanzi Studio. After following the steps from the tutorial, the test verifies that the preview of the final application is animating.


**Rotation tutorial** shows how to create a swiping gesture rotation for a 3D model in Kanzi Studio. See Figure 16.

## Tutorial: Rotate a 3D model

In this tutorial you learn how to rotate with a swiping gesture a 3D model in your Kanzi application.

You create a **Scroll View 2D** node and connect it to the rotation of the camera that shows the model that you want to rotate. Then you use the properties in the **Scroll View 2D** node to fine-tune how the **Scroll View** node rotates the model.

This video shows the result of the tutorial.



This tutorial assumes you understand the basics of working with Kanzi Studio. The best entry points for getting familiar with Kanzi Studio are:

**Figure 16.** Rotation tutorial snippet

The test for this tutorial verifies that users can create a scroll view motion when swiping a 3D model in the Kanzi application. After following the tutorial steps, the final application is verified by taking screenshot differences of Kanzi Preview.

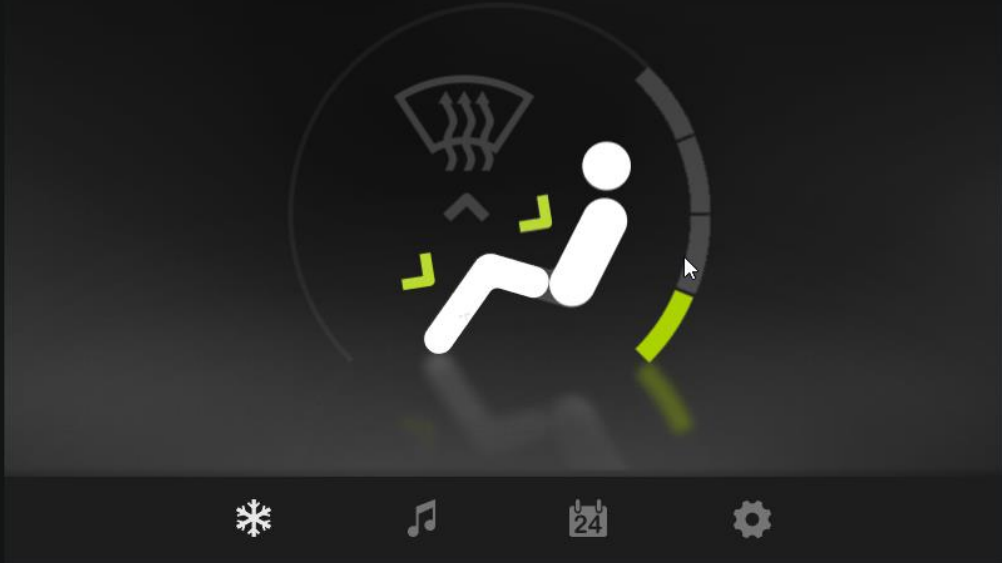
**State managers tutorial** shows users how to control the state of an application using Kanzi state managers. See Figure 17.

## Tutorial: Use state managers to control your application

In this tutorial you learn how to control the state of your application using state managers. You create an air conditioning application and use states to control the air conditioning indicators. You also learn how to use JavaScript scripts to control the state of the application.

**Note**  
This tutorial contains JavaScript. You can use JavaScript only on Windows operating system and only for prototyping.

This video shows the result of the tutorial.

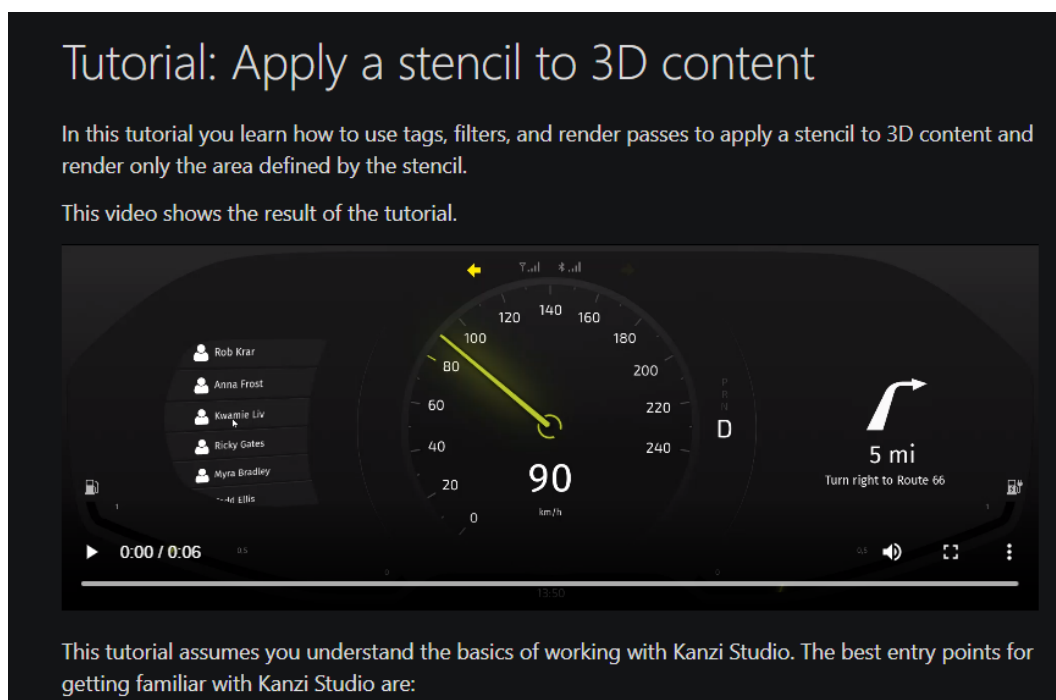


This tutorial assumes you understand the basics of working with Kanzi Studio. The best entry points for getting familiar with Kanzi Studio are:

**Figure 17.** State managers tutorial snippet

The test for this tutorial verifies that users can use Javascript scripts to control the state of a Kanzi application. After following the tutorial steps, the final application is verified by taking screenshot differences of Kanzi Preview.

**Stencil tutorial** shows how to use tags, filters, and render passes to apply a stencil to 3D content and render only the area defined by the stencil. See Figure 18.



**Figure 18.** Stencil tutorial snippet

The test for this tutorial verifies that users can apply a stencil to 3D content and render only the defined area by the stencil. After following the tutorial steps, the final application is verified by taking screenshot differences of Kanzi Preview.

**Toggle button tutorial** shows how to create a toggle button in Kanzi Studio using Kanzi state managers. See Figure 19.

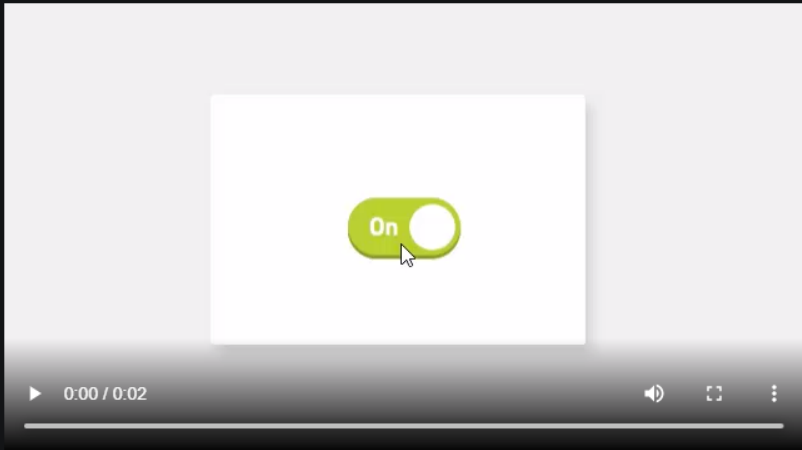
## Tutorial: Creating a toggle button

In this tutorial you learn how to create a toggle button in Kanzi Studio using the Kanzi state manager.

This tutorial assumes you understand the basics of working with Kanzi Studio. The best entry points for getting familiar with Kanzi Studio are:

- [Tutorial: Create a simple in-vehicle infotainment application](#)
- [Tutorial: Getting started with Kanzi Studio](#)

This video shows the result of the tutorial.



**Figure 19.** Toggle button tutorial snippet

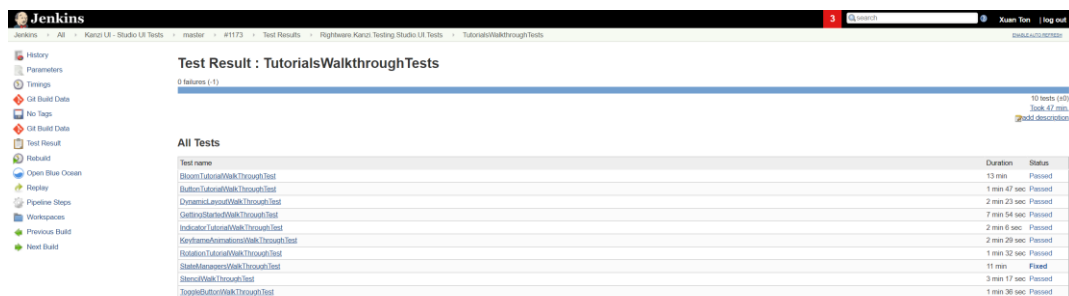
The test for this tutorial verifies that users can create a toggle button using the Kanzi state manager. After following the tutorial steps, the final application is verified by taking screenshot differences of Kanzi Preview. There are two screenshots of the On and Off state.

Moreover, the list of UI Tests can also be seen in Visual Studio when using the KanziStudioAutomatedUITests solution. See Figure 20.

Test	Duration	Traits	Err...
UI Tests (10)			
Rightware.Kanzi.Testing.Stu...			
TutorialsWalkthroughTests			
BloomTutorialWalkThro...		Release te...	
ButtonTutorialWalkThro...		Release te...	
DynamicLayoutWalkThr...		Release te...	
GettingStartedWalkThr...		Release te...	
IndicatorTutorialWalkTh...		Release te...	
KeyframeAnimationsWa...		Release te...	
RotationTutorialWalkTh...		Release te...	
StateManagersWalkThr...		Release te...	
StencilWalkThroughTest		Release te...	
ToggleButtonWalkThro...		Release te...	

**Figure 20.** List of UI Tests from Visual Studio

These tests run daily on our Jenkins Studio UI Test pipeline and every week-end on the Jenkins SDK Test pipeline. See Figure 21.



Test name	Duration	Status
BloomTutorialWalkThroughTest	13 min	Passed
ButtonTutorialWalkThroughTest	1 min 47 sec	Passed
DynamicLayoutWalkThroughTest	2 min 23 sec	Passed
GettingStartedWalkThroughTest	7 min 54 sec	Passed
IndicatorTutorialWalkThroughTest	2 min 4 sec	Passed
KeyframeAnimationsWalkThroughTest	2 min 29 sec	Passed
RotationTutorialWalkThroughTest	1 min 32 sec	Passed
StateManagersWalkThroughTest	11 min	Failed
StencilWalkThroughTest	3 min 17 sec	Passed
ToggleButtonWalkThroughTest	1 min 36 sec	Passed

**Figure 21.** Tutorial walkthrough tests run on Jenkins

This thesis implementation also includes the work on documenting a company internal Confluence wiki page which describes these UI tests and the test list. See Figure 22.



## Kanzi UiTests – Automated Tutorial Walkthrough List



Created by Xuan Ton

Last updated: Jul 13, 2021 • 2 min read • 14 people viewed

- General Information
- List of Automated Tutorial Walkthrough
  - Button Tutorial
  - Indicator Tutorial
  - Keyframe Animations Tutorial
  - Rotation Tutorial
  - Getting Started Tutorial
  - Toggle Button Tutorial
  - Dynamic Layout Tutorial
  - State Managers Tutorial
  - Bloom Tutorial
  - Stencil Tutorial

### General Information

Tutorial Walkthrough Tests are run nightly as a parallel step of the Jenkins job to run UI Tests.

Currently, these tests are run on the **master** branch and the **3.9 LTS** branches:

Latest master results are available from:

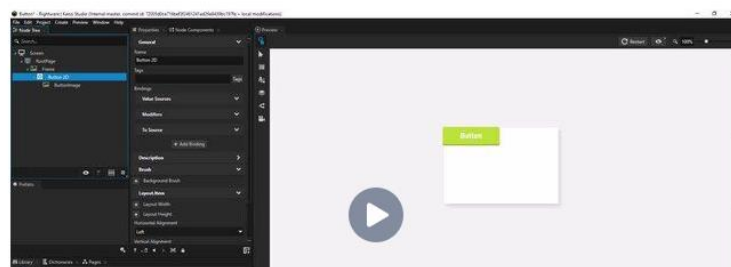
<http://ci/view/all/job/Kanzi%20UI%20StudioUiTests/job/master/lastCompletedBuild/testReport/Rightware.Kanzi.Testing.Studio.UI.Tests/TutorialsWalkthroughTests/>

Tutorials walkthrough tasks in hansoft: [hansoft://hansoftw.rightware.com/Projects;b0ebf8b5/Task/334287?ID=16918](https://hansoftw.rightware.com/Projects;b0ebf8b5/Task/334287?ID=16918)

### List of Automated Tutorial Walkthrough

Videos of test runs are added for future reference in creating tutorial in video form.

#### Button Tutorial



**Figure 22.** Internal documentation on tutorial walkthrough UI tests

### 7.3 Implementation of the UI Tests

Tests are planned using Hansoft project managing tool. The Tutorial Walkthrough backlog item includes all the tasks that are planned for implementing UI tests. See Figure 23.

Task Name	Status	Value	Due Date
Add Tutorial Walkthrough Tests to the SDK testset	Blocked	334287	
Making tutorial walkthroughs a parallel test step in	Completed	357248	2021-06-03
Automate Tutorial: Rotation	Completed	316342	2020-10-20
Automate Tutorial: Indicator	Completed	311578	2020-06-25
Automate Tutorial: KeyFrame Animations	Completed	313332	2020-07-31
Automate Tutorial: Getting started with Kanzi Studio	Completed	314126	2020-08-25
Automate the State managers tutorial	Completed	303430	2021-04-09
Automate tutorial: Dynamic layout	Finished	354210	2021-06-22
Automate Tutorial: Bloom	Finished	354217	2021-06-17
Automate Tutorial: Stencil	Finished	354219	2021-07-13
Automate tutorial: List box	Finished	361787	2021-07-13
Automate Tutorial: Blur	On Hold / Block	354212	
Automate Tutorial: Reflections	In Queue for Imj	354221	
	In Queue for Imj	361789	

**Figure 23.** Tutorial Walkthrough tasks planned in Hansoft

Test requirements can be found within Hansoft by looking at the description. Rightware uses Acceptance criteria to specify the required criterias for the UI tests. See Figure 24.

Field	Value
Name	Automate Tutorial: Rotation
Status	Completed
Assign tag	xuan.ton
Roadmap Size	
Completed date	2020-06-25
Affected releases	
Type	
Tech Area	
Lead time	8.72 days
Ripples in to doc...	

Description
Automate the Rotation tutorial ( <a href="http://buildserver/kanzi_documentation/master/en-us/Default.html#Tutorials/Rotation/Rotation.htm%3FDocPath%3D%2FTutorials%37C...11">http://buildserver/kanzi_documentation/master/en-us/Default.html#Tutorials/Rotation/Rotation.htm%3FDocPath%3D%2FTutorials%37C...11</a> ) using uitest framework
Acceptance criteria:
<ul style="list-style-type: none"> <li>new test method created in TutorialWalkthroughTests class</li> <li>test included to the TestCategories.TUTORIALS test category</li> <li>new test runs in scope of <a href="http://ci/job/Kanzi%20UI%20StudioUITests/job/master/">http://ci/job/Kanzi%20UI%20StudioUITests/job/master/</a> and passes</li> </ul>

**Figure 24.** Task has acceptance criteria as requirement

Next step would be the implementation of UI tests. We first will need to clone Kanzi UI Git repository or fetch new changes if it is already existed, and create a local branch. To create these automated walkthrough UI tests, a TutorialWalkthroughTests class was created under the recommended path in our Kanzi UI Git repository: ui\Tool\KanziToolAutomatedUITests\Tests\UITests. See Figure 25.

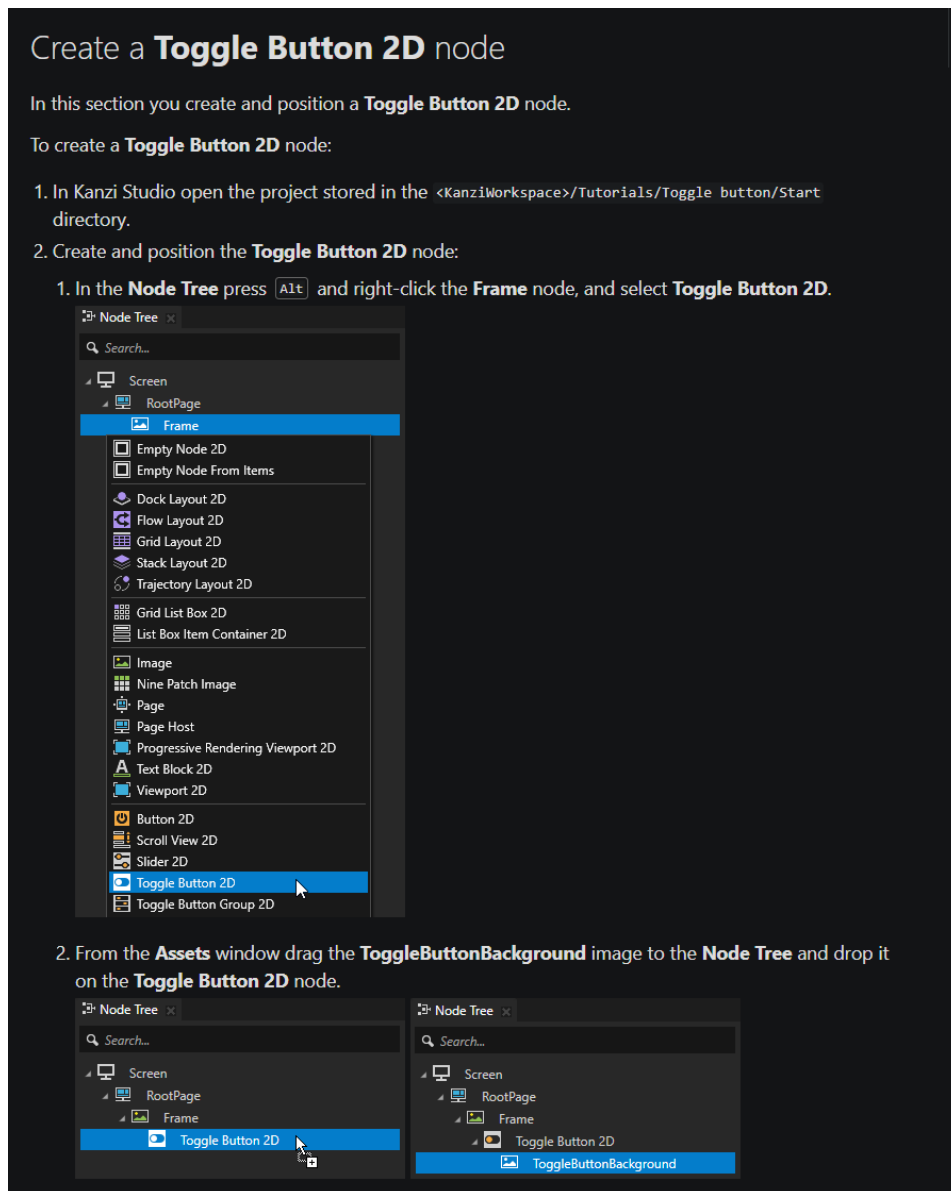
The screenshot shows a Windows File Explorer window with the following path: `Windows (C:) > git > ui > Tool > KanziToolAutomatedUITests > Tests > UITests`. The window displays a list of files and folders with columns for Name, Date modified, Type, and Size. The file `TutorialWalkthroughTests.cs` is highlighted.

Name	Date modified	Type	Size
ExampleTests.cs	9/6/2021 2:29 PM	Visual C# Source f...	16 KB
FactoryContentTests.cs	9/6/2021 1:49 PM	Visual C# Source f...	20 KB
GLTFImportTests.cs	9/6/2021 1:50 PM	Visual C# Source f...	12 KB
ItemHandlingTests.cs	8/2/2021 4:32 PM	Visual C# Source f...	18 KB
LayoutTests.cs	9/2/2021 3:36 PM	Visual C# Source f...	3 KB
LinearColorTests.cs	9/6/2021 1:58 PM	Visual C# Source f...	6 KB
ListBoxTests.cs	9/6/2021 1:58 PM	Visual C# Source f...	17 KB
LocalizationTests.cs	9/6/2021 1:58 PM	Visual C# Source f...	52 KB
MergeTests.cs	8/2/2021 4:32 PM	Visual C# Source f...	15 KB
PagesTests.cs	9/6/2021 1:58 PM	Visual C# Source f...	24 KB
PluginTests.cs	8/2/2021 4:32 PM	Visual C# Source f...	6 KB
PrefabTests.cs	8/2/2021 4:32 PM	Visual C# Source f...	1 KB
PreviewPluginTests.cs	8/2/2021 4:32 PM	Visual C# Source f...	10 KB
PreviewTests.cs	9/6/2021 1:50 PM	Visual C# Source f...	74 KB
ProjectConfigurationTests.cs	8/2/2021 4:32 PM	Visual C# Source f...	10 KB
ProjectDescription.txt	8/2/2021 4:32 PM	Text Document	1 KB
ProjectLoadingTest.cs	8/2/2021 4:32 PM	Visual C# Source f...	4 KB
ProjectTests.cs	9/6/2021 1:53 PM	Visual C# Source f...	132 KB
PropertyTests.cs	9/6/2021 1:50 PM	Visual C# Source f...	56 KB
RenderingTests.cs	9/6/2021 1:51 PM	Visual C# Source f...	3 KB
RenderpassBlurTests.cs	8/2/2021 4:32 PM	Visual C# Source f...	2 KB
ScriptingTests.cs	9/6/2021 1:58 PM	Visual C# Source f...	4 KB
SelectorTests.cs	8/2/2021 4:32 PM	Visual C# Source f...	7 KB
ShaderTests.cs	9/6/2021 1:51 PM	Visual C# Source f...	2 KB
SimpleTests.cs	9/6/2021 2:29 PM	Visual C# Source f...	2 KB
SkinningTests.cs	9/6/2021 1:51 PM	Visual C# Source f...	13 KB
StateManagerTests.cs	8/2/2021 4:32 PM	Visual C# Source f...	6 KB
StyleTests.cs	9/6/2021 1:58 PM	Visual C# Source f...	3 KB
TextboxTests.cs	8/2/2021 4:32 PM	Visual C# Source f...	2 KB
ThemingTests.cs	9/6/2021 1:58 PM	Visual C# Source f...	58 KB
TriggerTests.cs	9/6/2021 1:52 PM	Visual C# Source f...	5 KB
<b>TutorialWalkthroughTests.cs</b>	9/6/2021 4:10 PM	Visual C# Source f...	125 KB
TutorialTests.cs	9/6/2021 2:29 PM	Visual C# Source f...	71 KB
UITests.csproj	8/27/2021 1:37 PM	C# Project file	2 KB
WindowTests.cs	9/6/2021 1:51 PM	Visual C# Source f...	5 KB

**Figure 25.** TutorialWalkthroughTests.cs location

Each tutorial walkthrough UI test will be added to this class as a method using attributes `[TestMethod]` and `[TestCategory(TestCategories.WALKTHROUGH)]`. `[DeploymentItem()]` attribute is also used to copy the tutorial project directory to the test execution directory to not overwrite the original implementation.

For each UI test, steps which were documented in the Kanzi tutorial were followed. For example, looking at the Toggle button tutorial from Figure 26.



**Figure 26.** Some steps from the Toggle button tutorial

The first step is:

1. In Kanzi Studio open the project stored in the <KanziWorkspace>/Tutorials/Toggle button/Start directory.

Most basic execution, such as opening a project or creating a node was already implemented in the Kanzi Automated UI Test tool which can be called by using for example `Studio.OpenProject()`.

For each step of the tutorial, assertion was used to make sure that the step is executed without any issue. With assertion, we make sure that the test will stop and fail if the executed test has an error. On the other hand, the test will proceed until the end and pass if the step satisfies the assertion.

Therefore, for the first step in this Toggle button tutorial, something like `Assert.IsTrue(Studio.OpenProject(ToggleButtonProject))` would be used, “Cannot open project”) – which the first parameter is the condition, here indicates that opening project must return true. The second parameter is the message which will be thrown if the condition is not met.

Using this method, assertion was added for each step of the tutorial into the UI test which will then automatically run and execute the step added.

During the implementation, some operations described from the Kanzi UI Documentation steps might not be already implemented in the existing UI test framework, in that case, a Boolean method to perform the described step should be implemented and added to the UI test framework, then an assertion inside the test method would be called.

We finalize the UI test with a screenshot verification to verify that the test works correctly, along with Kanzi Studio and Kanzi UI Documentation.

#### **7.4 A Scenario when Implementing Tutorial Walkthrough UI Test**

There are a total of ten tutorials so it would be repetitive to describe each scenario when creating each UI test, therefore, the Rotation tutorial walkthrough test was chosen to describe in detail, due to the appropriate

length of this tutorial and because this is the first tutorial implemented while learning about UI automated testing.

Before implementing the UI test, the requirement of the test scenario was specified by having a user story, in this case, “As a Kanzi developer, I want to create an application that has rotating motion when swiping a 3D model”.

The next step was the test planning. By using Kanzi UI Documentation, the steps of the test were planned, as well as the length of the implementation, and the test verification.

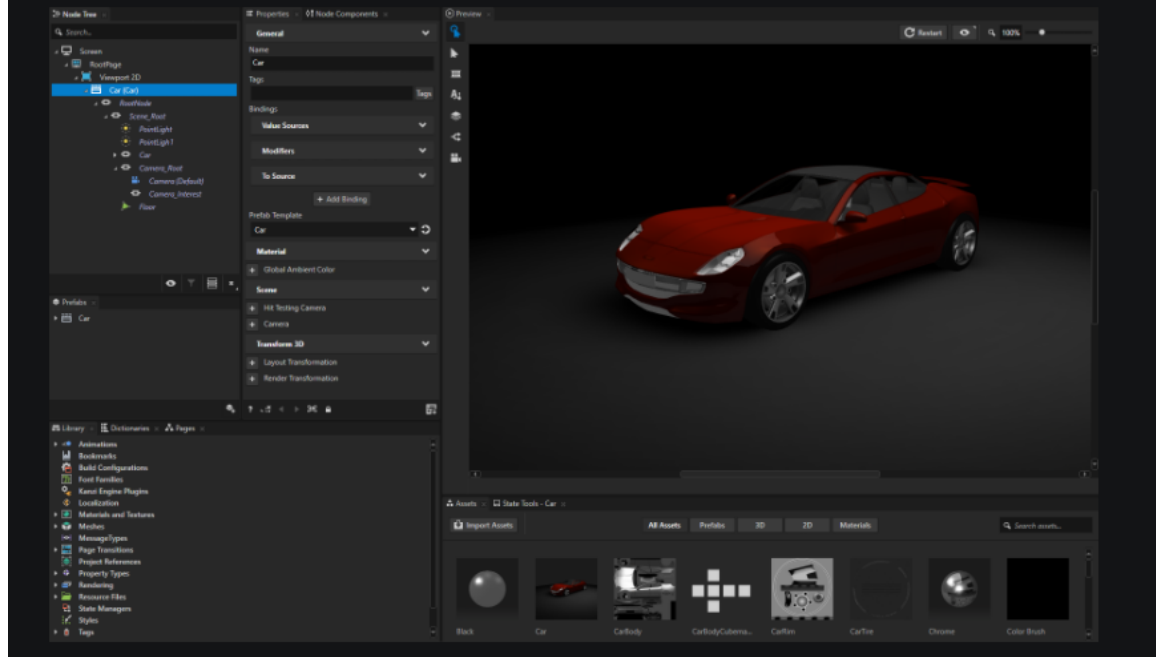
After planning, the implementation of the the UI test case was started for this Rotation tutorial.

Firstly, in the test class, `TutorialsWalkthroughTests`, the test method was specified which contains required attributes such as `[TestMethod]` and `[TestCategory(TestCategories.WALKTHROUGH)]`. The method was defined with `public void RotationTutorialWalkthroughTest()`. The `[DeploymentItem]` attribute was used to specify the files needed for the test. The test system will create a new directory where the tests are run from. With this attribute, the test system copies specific files to that new directory.

The first step of the tutorial, “In Kanzi Studio, open the project stored in `<KanziWorkspace>/Tutorials/Rotation/Start`”, can be seen in Figure 27.

1. In Kanzi Studio open the project stored in <KanziWorkspace>/Tutorials/Rotation/Start.

The project contains the **Car** prefab that contains a model of a car, lights, camera, and the floor below the car.

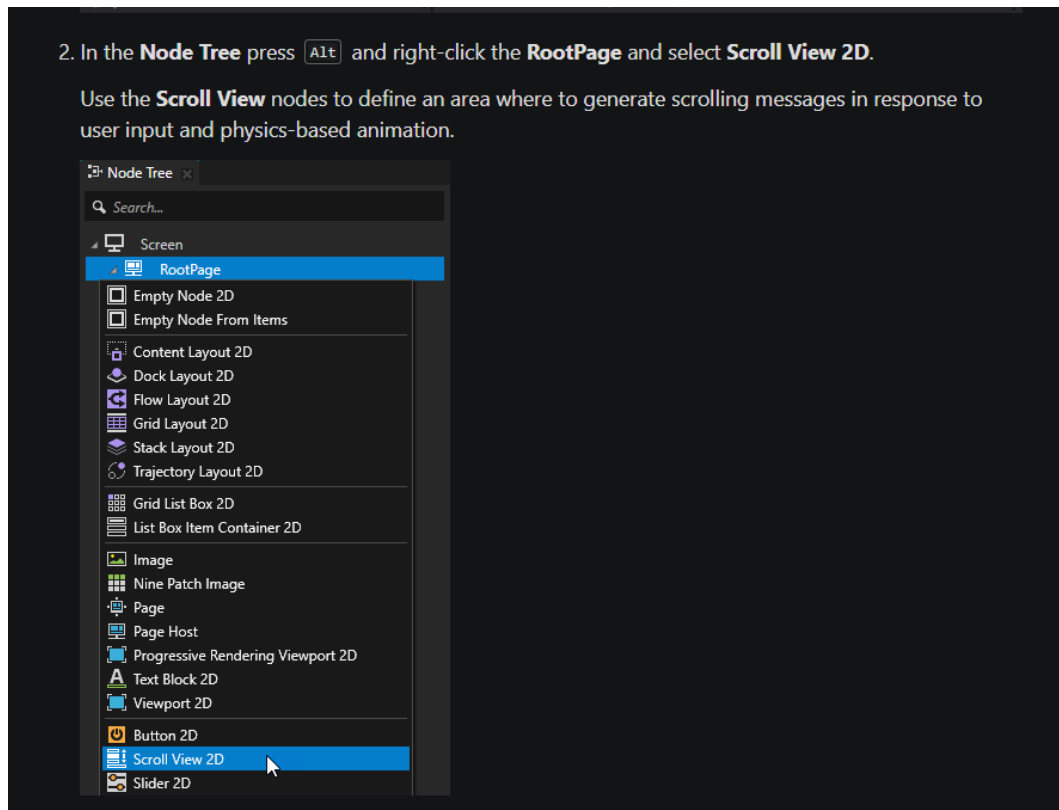


**Figure 27** The first step of the Rotation tutorial

After declaring the method for this test, the implementation of the statements for each step in the Rotation tutorial started.

Due to confidential reasons, no source code can be shared externally, an example for the first statement of this test method can be: `Assert.IsTrue(Studio.OpenProject(RotationProject), "Failed to open Rotation project")`

The second step can be seen in Figure 28.



**Figure 28.** The second step of the Rotation tutorial

For this step, another statement was added which is something like `Assert.IsTrue(Studio.AddNodeByRightClick(Scroll View 2D), "Failed to add Scroll View 2D")`

The work continued by adding each statement for each step from the tutorial. If there is no already implemented method, for example, `AddNodeByRightClick()`, it was implemented within the Kanzi UI test framework.

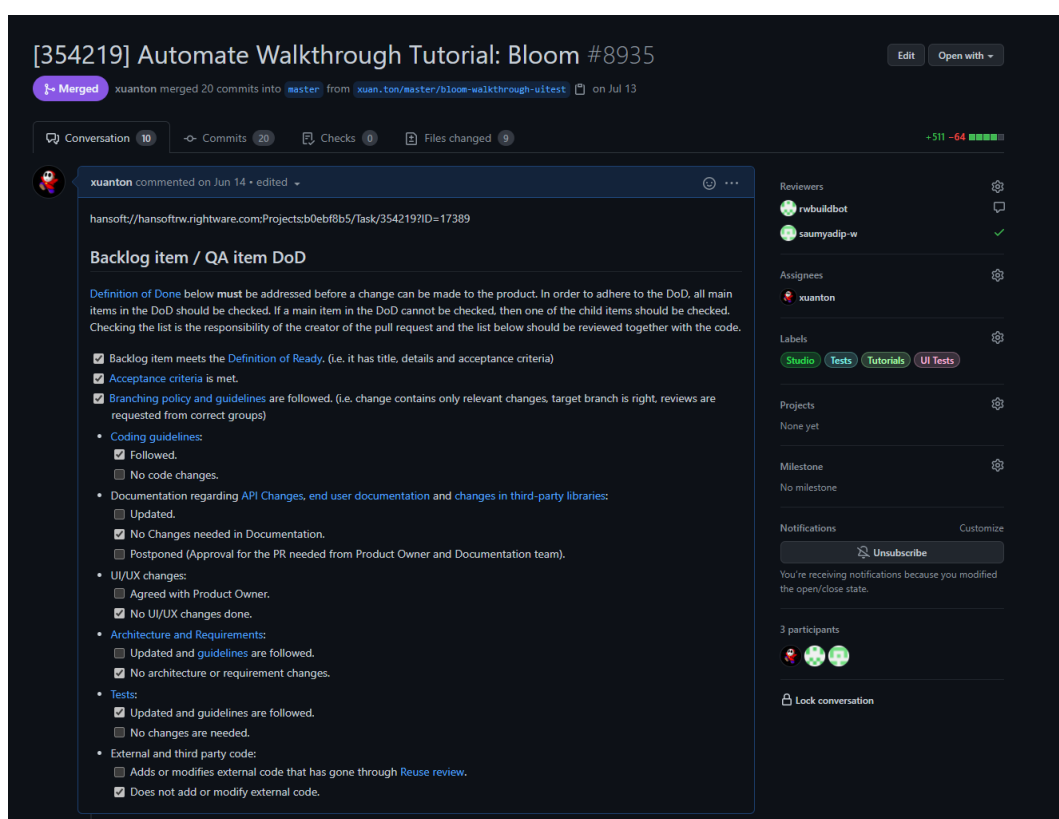
After finishing the tutorial, a screenshot verification statement was added to ensure that the test and Kanzi Studio work as expected. For example, `Assert.IsTrue(Studio.ScreenshotVerification(ScreenshotPath), "Screenshot is not expected")`

The UI test was created within a local branch of the Kanzi UI Git repository. After finishing with the implementation, I push the branch with the latest



changes was pushed and a pull request was created to merge it to the master development branch.

Before being able to merge to the master development branch, at least one approval was required from another software engineer. To make sure that the test works correctly on Jenkins, UI test staging was run only for the created branch. See Figure 29.



**Figure 29.** A snippet of Bloom tutorial walkthrough UI test pull request

After being merged to the master branch, the UI test is ready to be run every day in the evening for the latest master build. The test report can be found on Jenkins for each build.

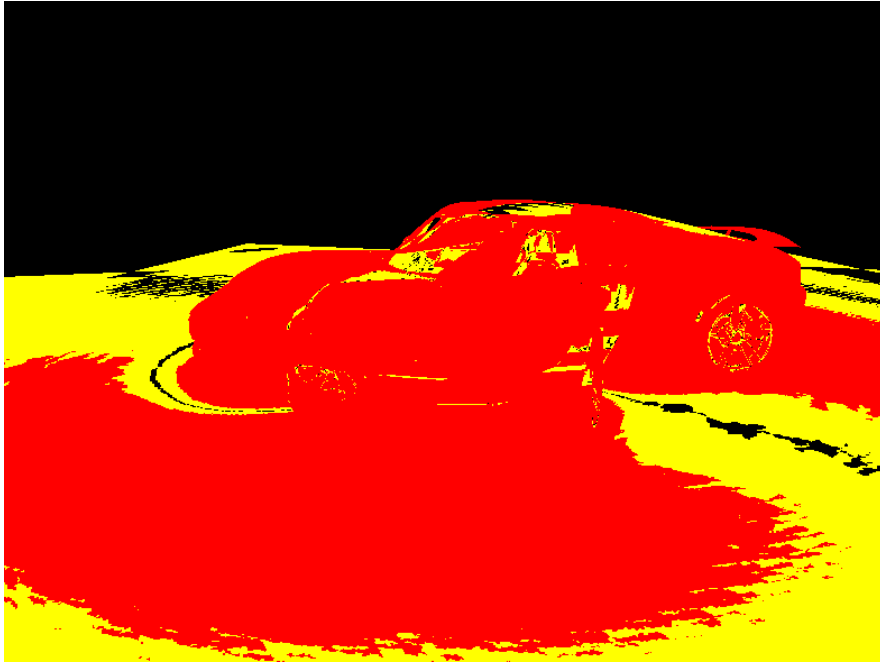
## 7.5 Testing for Kanzi UI Tests

We have unit tests for the UITestFramework solution which can clarify that the methods implementation meets the acceptance criteria. However, for automated UI tests, we use Screenshot verification which was mentioned before, for these tests to verify that tests are working normally along with Kanzi Studio and Kanzi documentation.

For example, this is a scenario where RotationTutorialWalkthroughTest fails because of screenshot verification, we will be able to check the difference in Figures 30 and 31.



**Figure 30.** Fail screenshot verification in Rotation tutorial



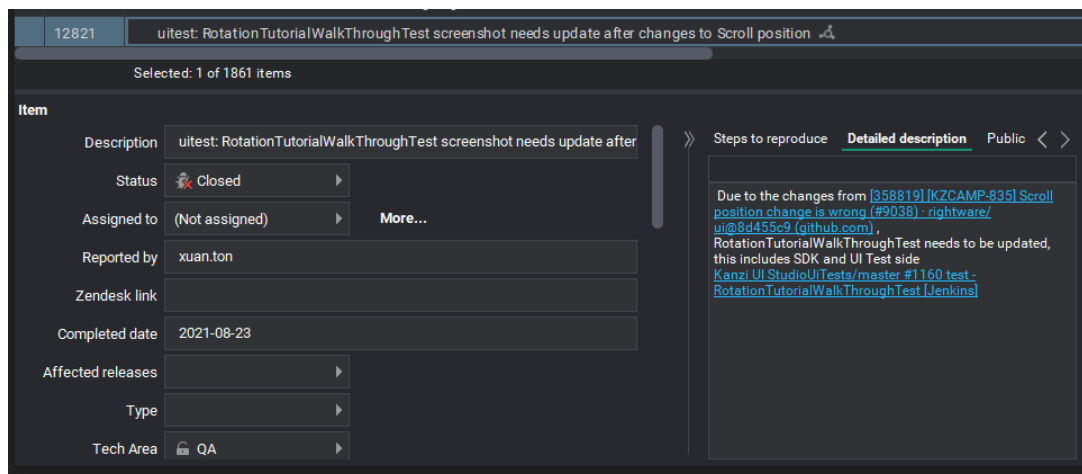
**Figure 31.** Highlighted difference screenshot

UI test is run for each master and release build so whenever there is a failure in test execution, we can check the test report and error message from Jenkins, including the screenshot verification differences if there is a issue in screenshot verification. See Figure 32.



**Figure 32.** Jenkins UI test failure screenshot

When a test fails, the issue is tracked using Hansoft tool, with different status that can be followed if the bug is assigned, in progress or in review. See Figure 33.



**Figure 33.** A bug ticket for Rotation tutorial walkthrough UI test from Hansoft

## 8 CONCLUSION

In software development, testing is always a crucial process to ensure that the software product is acceptable for release. As software gets more and more complicated, automated tests are getting more and more popular. Therefore at Rightware Oy, we also focus on automated testing, such as UI tests and unit tests, along with CI/CD to produce reliable Kanzi UI versions for delivery.

As Kanzi UI Documentation needs to be updated when features are introduced, there must be a solution to make sure the documented steps from tutorials are still valid. Moreover, manual testing is needed for every release which requires resources in walking through some selected tutorials. With these in mind, the QA team at Rightware decided that it would be beneficial to have automated tutorial walkthrough UI tests created in the existing automated UI test framework. Therefore, this thesis implemented ten automated tutorial walkthrough tests which are around one-third of the existing number of tutorials in Kanzi UI Documentation. These UI tests can also be used in the future to create video-based tutorials if there is a need for that.

This thesis implementation only includes the creation of ten automated tutorial walkthrough UI tests but the work which is out-of-scope of this is the maintenance and update of the UI tests when there are required changes. Moreover, there is a plan in creating more automated walkthrough UI tests which can cover more tutorials in the Kanzi UI Documentation.

In the scope of this thesis, I and my supervisors believe the list of ten automated tutorial walkthrough UI tests is sufficient. However, the work on these UI tests continues along with my employment at Rightware Oy after graduation.

## REFERENCES

Atlassian. 2021. What is Continuous Integration. Accessed 22 July 2021. <https://www.atlassian.com/continuous-delivery/continuous-integration>

Chauhan, S. 2018. Understanding WPF Architecture. Accessed 20 July 2021. <https://www.dotnettricks.com/learn/wpf/understanding-wpf-architecture>

Freeman, E. 2019. What is Continuous Delivery?. Accessed 19 July 2021. <https://aws.amazon.com/devops/continuous-delivery/>.

GeeksforGeeks. 2021. What is WPF? . Accessed 22 July 2021. <https://www.geeksforgeeks.org/what-is-wpf>

Guru99. 2021a. What is Jenkins? Why Use Continuous Integration (CI) Tool?. Accessed 6 September 2021. <https://www.guru99.com/jenkin-continuous-integration.html>

Guru99. 2021b. Coded UI Test Automation Framework Tutorial. Accessed 22 July 2021. <https://www.guru99.com/coded-ui-test-cuit.html>

Guru99. 2021c. What is Software Testing? Definition, Basics & Types in Software Engineering. Accessed 22 July 2021. <https://www.guru99.com/software-testing-introduction-importance.html>

Krüger, N., 2021. Why Is Software Testing Important?. Accessed 16 September 2021. <https://www.perforce.com/blog/alm/what-software-testing>

Microsoft Docs. 2020. Overview of .NET Framework. Accessed 20 July 2021. <https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview>

Microsoft Docs. 2019. An Introduction to NuGet. Accessed 18 July 2021. <https://docs.microsoft.com/en-us/nuget/what-is-nuget>

Medium. 2021. What is UI vs. UX design? What's the difference?. Accessed 22 July 2021. <https://uxplanet.org/what-is-ui-vs-ux-design-and-the-difference-d9113f6612de>

Perfecto by Perforce. 2021. Switching to UI Automation: Everything You Need to Know. Accessed 18 July 2021. <https://www.perfecto.io/blog/switching-ui-automation-everything-you-need-know>

Rightware, 2020. About us. Accessed 18 July 2021. <https://www.rightware.com/company>

Singh, S. 2015. Overview Of WPF Architecture. Accessed 18 July 2021. <https://www.c-sharpcorner.com/UploadFile/819f33/overview-of-windows-presentation-foundation-wpf-architectu/>

Teststackwhite Docs. 2021. TestStack.White. Accessed 22 July 2021. <https://teststackwhite.readthedocs.io/en/latest>