uViLEd

Visual Logic Editor

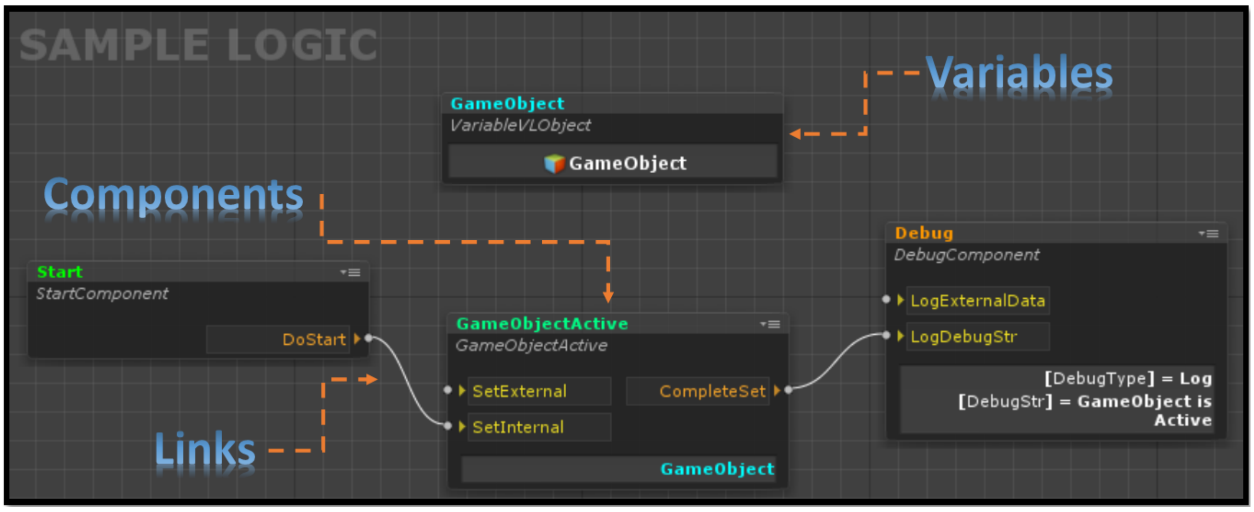
Last edition 28.06.2018

User manual

Introduction

Welcome to the uViLEd user guide. This document will describe the basic principles of working with the logic editor: creating and editing logic, setting up, working with components, the catalog and the property inspector. Also, the specifics of working with sound, localization and global messages will be considered.

Before proceeding with the description of the editor, let's summarize what the visual programming system of uViLEd is in general.



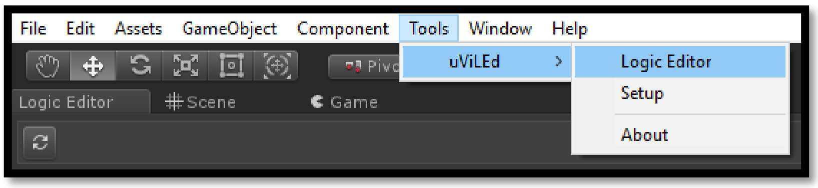
The figure shows a generalized scheme that describes the main parts that determine the system of visual programming. Here:

- **Components** are logic that implements a certain functional. In general, this is the part that receives, processes, and transmits data.
- **Variables** are data that is shared between components in one logic.
- **Links** - this is the part that determines the sequence of execution of the components logic.
- **Logic** is a set of components, variables, and links that implements some functional (s) of the application.

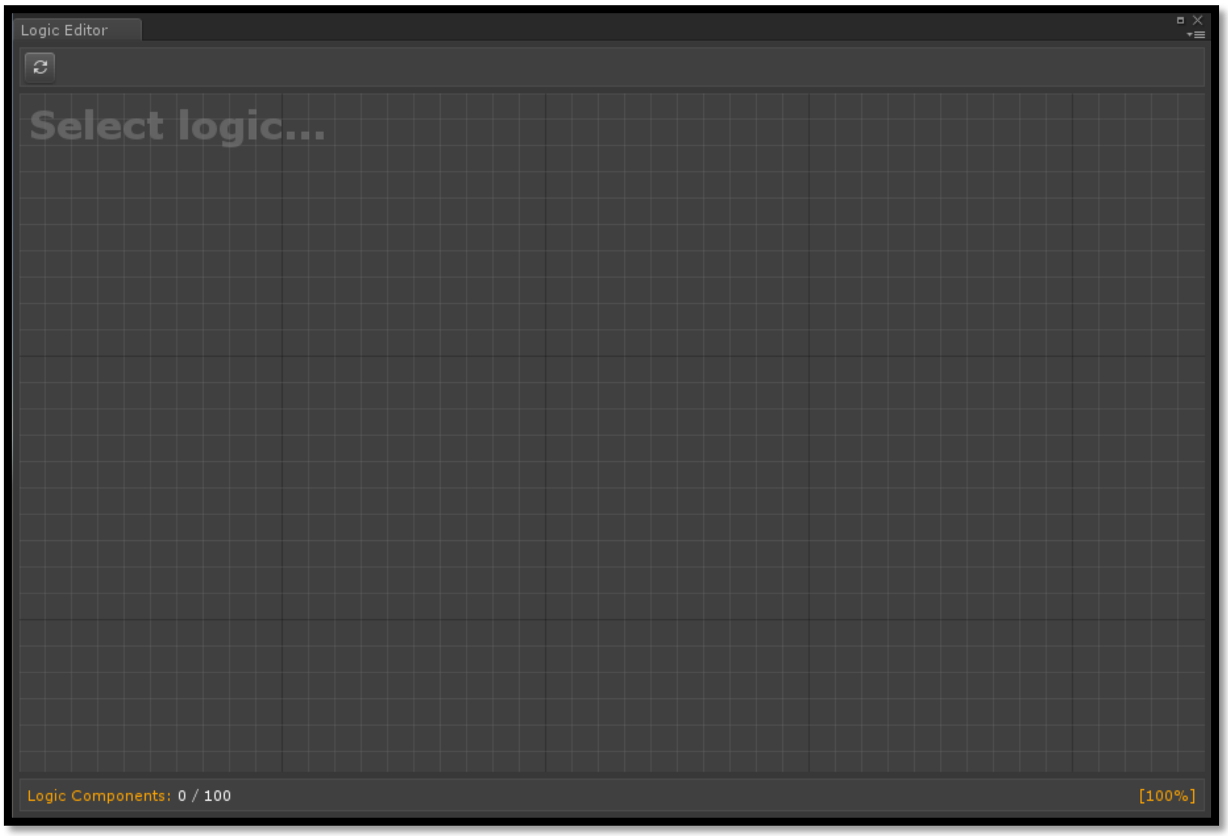
Let's now describe in more detail how to create logics and breathe life into them

Start working with uViLEd

To start working with the visual programming system, open the scene and select the menu item, as shown in the figure below.



After starting the logic editor, to initialize the scene, you should press the button as shown in the figure (the same button recovers the scene logics, if any).

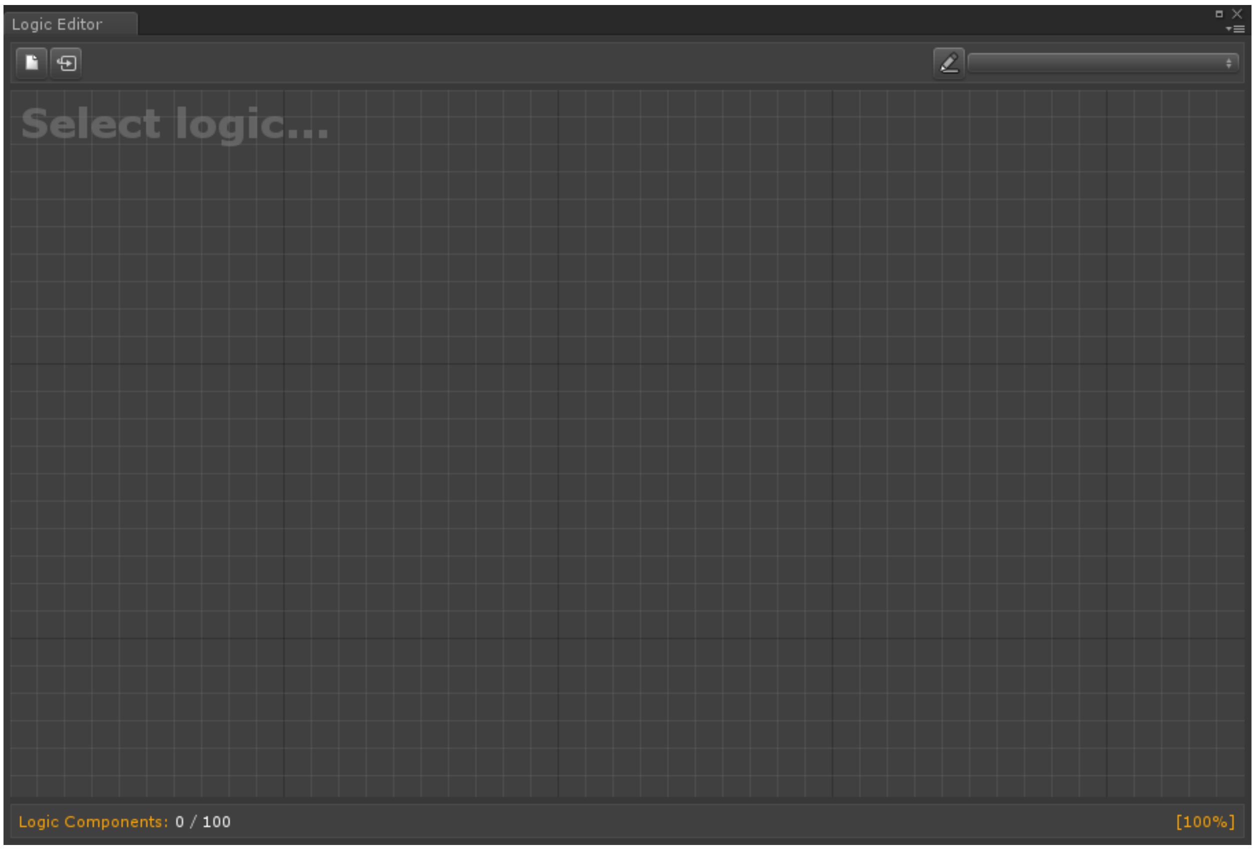


Now the scene is ready for work.

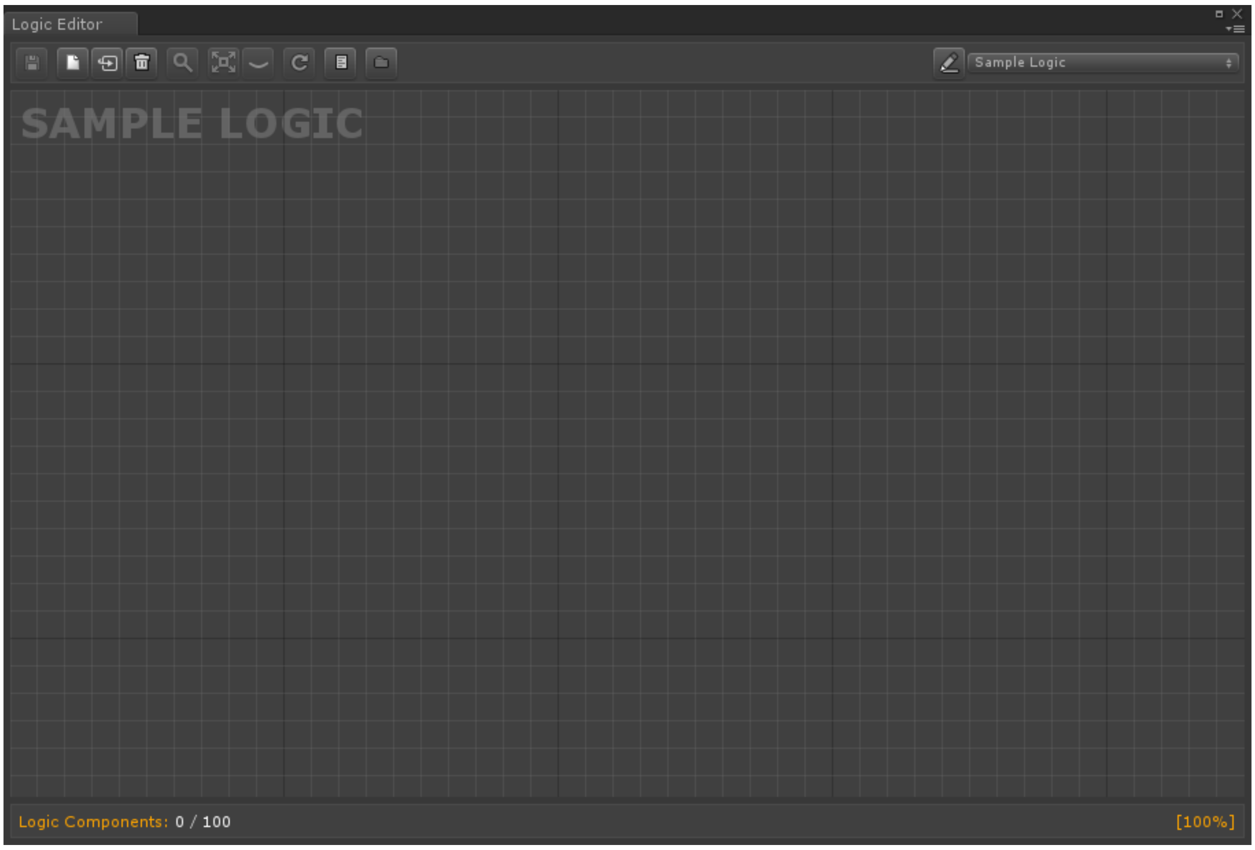
The editor

Working with logics

After the scene is initialized, to start working with the logic, you need to create it, or add an existing one (for example, from templates), use the buttons as shown in the figure.



After creating a new logic (or adding an existing one), the editor is ready to work with it. Here are its components in more details.



The logic editor menu items are:

1. Save the current logic (saving is performed automatically if the application is launched in editor mode). You can also configure periodic automatic saving (see section Settings).
2. Create a new logic
3. Add an existing logic or a ready template to the scene
4. Delete the current logic from the scene (logic file will remain, to totally delete the logic you must delete the logic file)
5. Search component in the current logic
6. Minimize or maximize all components of the logic
7. Hide or show components parameters
8. Undo all unsaved changes in the logic.
9. Open scene logic messages window
10. Change the name you need to click on it.
11. Select a scene logic and change its name.

Zooming and scrolling

You can change the view in the editor window by scrolling to the different parts of a logic, and by changing the view scale.

To scroll the view:

(PC) - drag the mouse with 3-d button pressed or hold the Alt key and drag the mouse with the left button pressed.

(Mac) – Hold the Alt key and drag the mouse with left button pressed (or touchpad corner).

To zoom the view in/out:

(PC) – use the mouse wheel.

(Mac) – use your apple mouse surface or move two fingers over the trackpad.

After getting familiar with logics, now let's learn how to fill them with components.

Working with components

Adding components to a logic

In order to add an component to the scene, there are two ways:

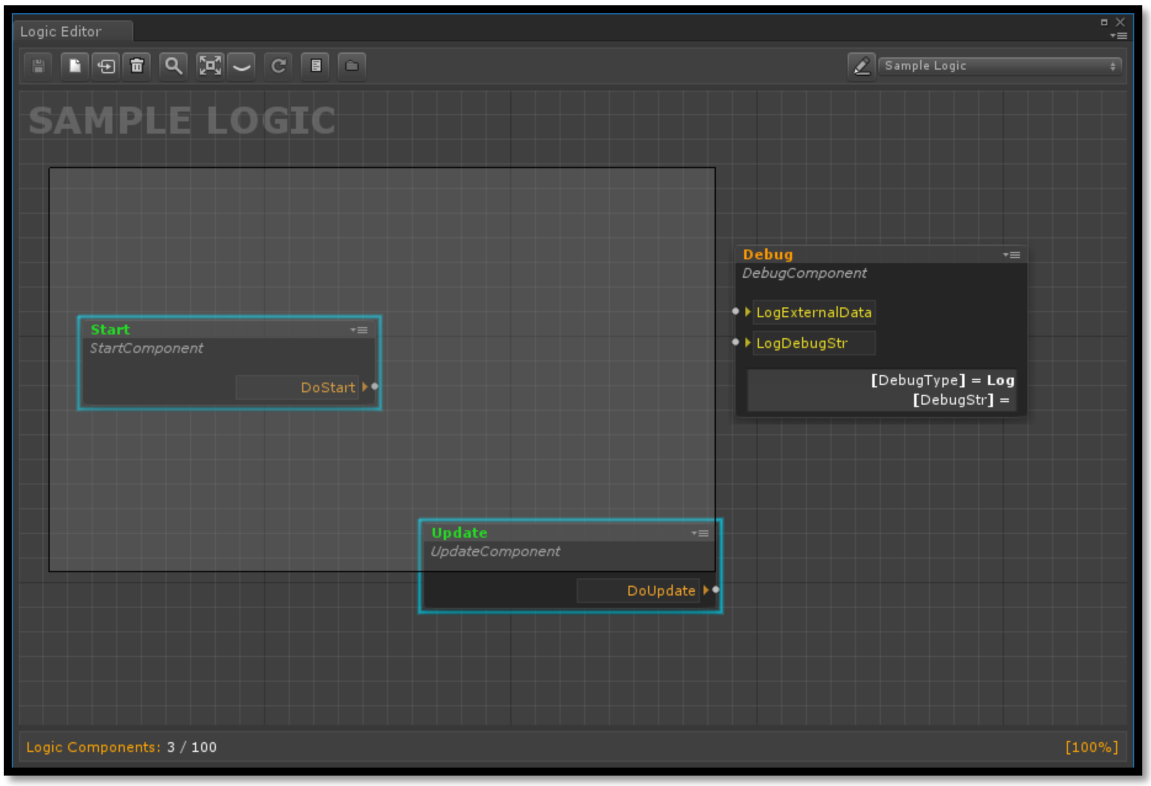
1. Drag a component from the catalog to the drawing area of the components.
2. Add a component through the context menu, called by clicking on the right mouse button.
3. Drag a component script form project to logic area

This way you can fill the logic with components, after which you can perform various manipulations on them.

Moving components

To move a component, you need to select it, just click on it with the left mouse button. You can also select several components. To do this, you need to perform the above manipulations, but with the Ctrl key pressed. Selecting several components in the area is done by moving the mouse with the left mouse button pressed.

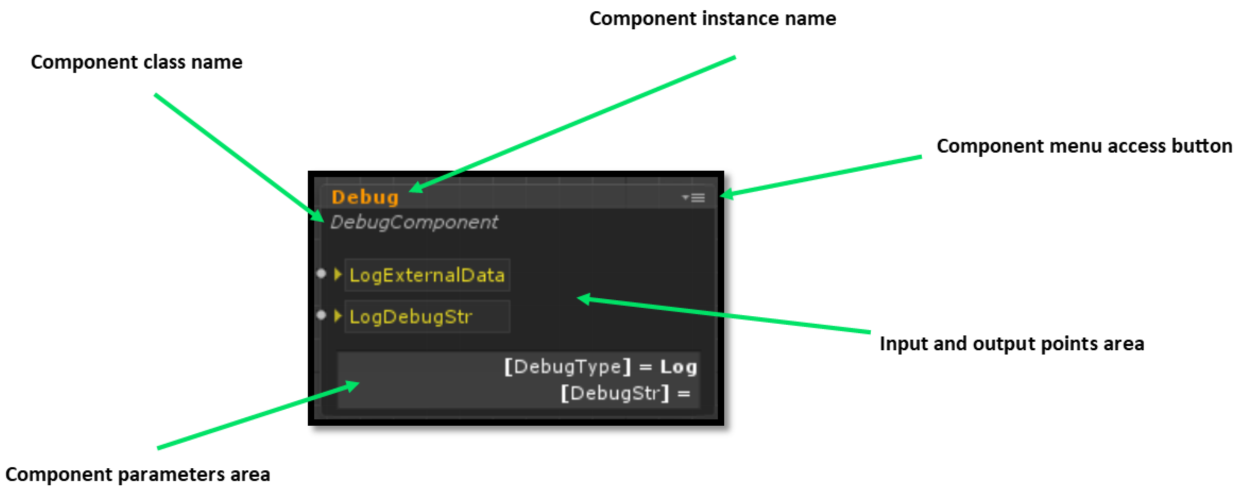
Once the components are selected, you can move them by holding the same mouse button on the component title and moving it to the desired position. Below is a picture for explanation.



Interacting with components

The same actions can be performed on an component through the context menu called by clicking the right mouse button on the component.

In addition to the menu buttons described above, you can directly interact with a specific component. The component consists of several components:



Here:

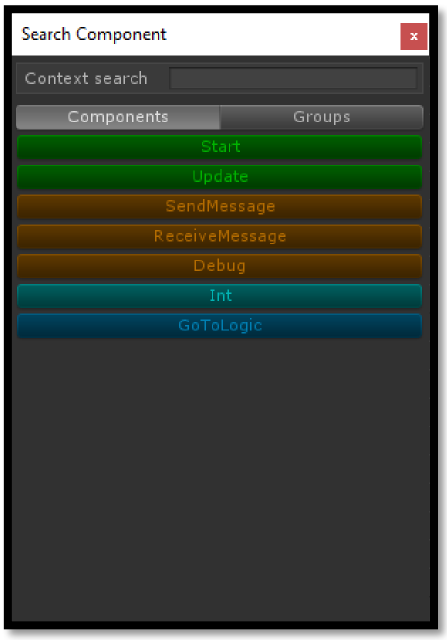
- The component menu button opens a context menu for interacting with an component, which can contain the following items:
    - **Minimized** – button for minimization or maximization of the component (switch the view of the component into a compact form). The button may be absent if the component has no more than one entry point or an exit point.

Also you can switch visual state of the component by double-clicking on its header.
    - **Inversion** – this button inverts the position of the input and output points.
    - **Parameters** – this button hide or show component parameters
    - **Edit** – opens a file with the code that implements the component.
- The component parameters area is the zone where the key parameters of the component and their values are displayed.

Finding components in a logic

Depending on the needs of the developer, the logic can contain a large number of components, which in turn can complicate the navigation on it. To solve this problem in the editor, you can search the logic with the possibility of focusing on the found component. To open the search window, press the menu button.

Below is a picture of this window.

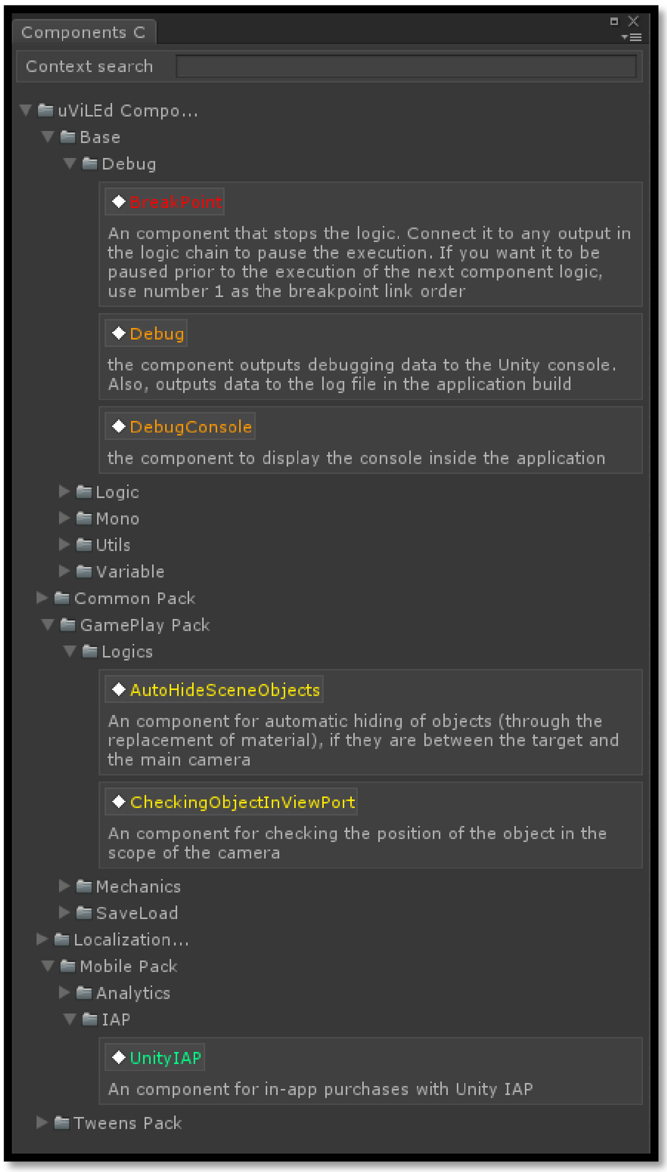


In order to find a specific component by name, it must be entered in the input line. Search among the component s is carried out contextually. To focus on the component, you must click on the corresponding button in the search results area.

The catalog of components

To add an component to the logic it can be dragged from the catalog to the drawing area. To open the catalog window, click the button in the main editor menu.

Below is a picture showing the window of the components catalog:



Components C

Context search

uViLEd Compo...
Base
Debug

BreakPoint
An component that stops the logic. Connect it to any output in the logic chain to pause the execution. If you want it to be paused prior to the execution of the next component logic, use number 1 as the breakpoint link order

Debug
the component outputs debugging data to the Unity console. Also, outputs data to the log file in the application build

DebugConsole
the component to display the console inside the application

Logic
Mono
Utils
Variable
Common Pack
GamePlay Pack
Logics

AutoHideSceneObjects
An component for automatic hiding of objects (through the replacement of material), if they are between the target and the main camera

CheckingObjectInViewPort
An component for checking the position of the object in the scope of the camera

Mechanics
SaveLoad
Localization...
Mobile Pack
Analytics
IAP

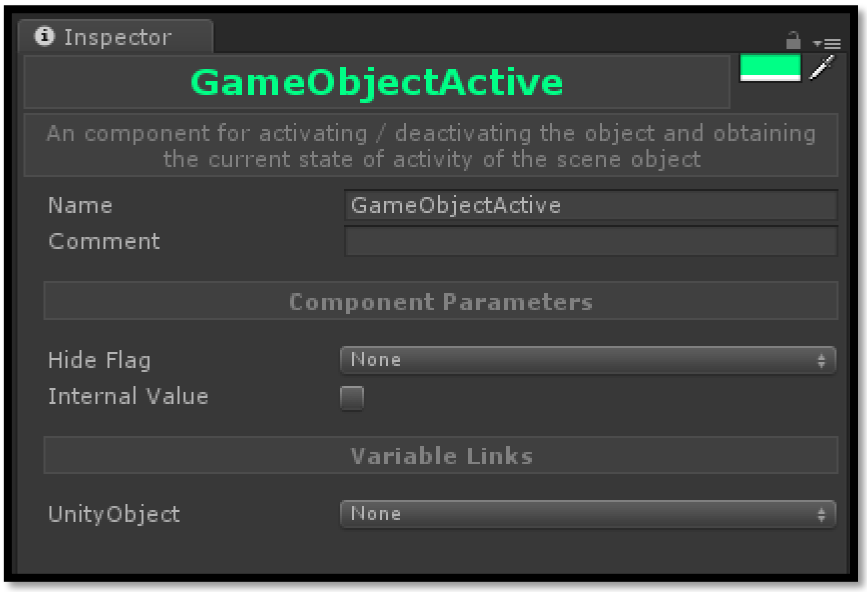
UnityIAP
An component for in-app purchases with Unity IAP

Tweens Pack

The catalog is a structure of folders and components in them. Each component is presented in the form of a name and a brief description of what it does. Since a large number of components can be present in the catalog, for convenience of navigation on them, it is possible to search through the input line at the top of the window. Search among the components is done contextually as you type.

Component inspector

Above we have considered almost all aspects of interaction with the component, the last important part is setting up component parameters For this, the uViLEd system uses the Unity 3d inspector


Inspector

GameObjectActive

An component for activating / deactivating the object and obtaining the current state of activity of the scene object

Name GameObjectActive
Comment

Component Parameters

Hide Flag None
Internal Value

Variable Links

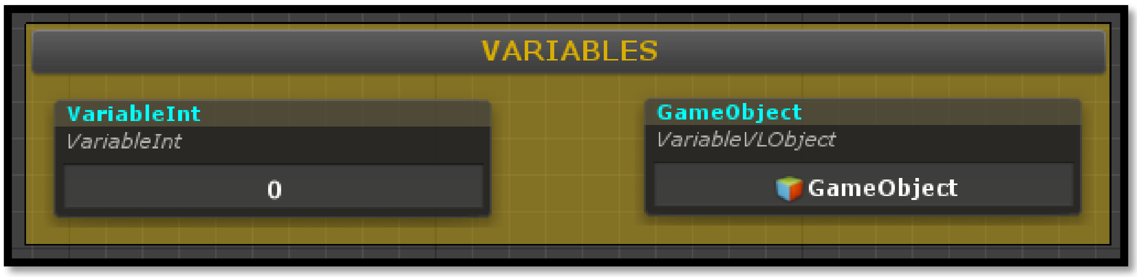
UnityObject None

Here:

- In the upper right corner, there is a button for setting the color of the component header.
- **Name** – field for changing the name of a component
- **Comment** – field for setting a comment for the component (the comment is displayed when you hover the mouse over a component).
- **Component Parameters** – component parameters area.
- **Variable Links** – an area for setting references to logic variables, if any. Read more about the variables below.

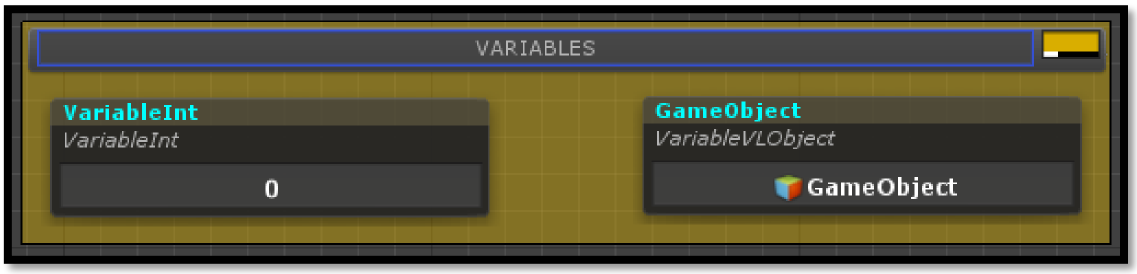
Component groups

While working with the logic, you may need to increase its readability, so that other developers can quickly understand what is happening there. In the basic version, the logic of the game is divided among the logics, so that each of them is responsible for its specific part. However, even in this case, the number of components can be quite large, and it can be difficult to understand logic logic. To improve the perception of the logic in the uViLEd system, the visual component groups can be created. Below is an image showing a group of variables.



To create a group, select the component s and chose **Group** in the right-click menu. After creating a group, you can move it, just like a single item (holding the mouse cursor on the group header).

To configure the name of the group and its color, double-click on the header. The figure below shows the editing mode.



For the newly created component to be immediately placed in a group, you must either drag it from the catalog to the group area, or right-click it in the group.

To exclude components from the group, select them and select **Ungroup** from the right-click menu.

Note: Group boundaries are automatically adjusted depending on the position of the contained components.

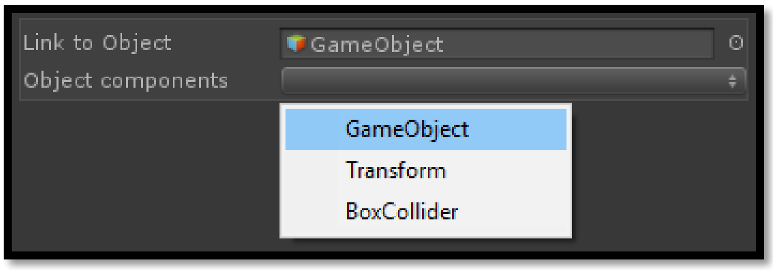
Logic variables

Variables are the same components that do not have input and output points and have only one parameter that is responsible for the value of the variable. The main meaning of this entity is to be able to separate data between the component s of the logic so that you do not have to perform the same manipulations with each new component. Thus, if two component s refer to the same variable, then if you change the value of the variable, you do not need to do anything with the components, they will automatically use the new value.

Note: when setting references to variables, their list in the inspector is automatically generated for the type of data that is required for the operation of the components.

Working with Unity objects

Due to the specifics of the uViLEd system operation, direct links to Unity objects cannot be used in it, since they cannot be restored when the logic is loaded. To solve this problem, a specialized VLObject shell was created, which allows you to create such references, and can save and load them. Among other things, this shell has a special property editor that allows you to get components from any scene object (see the figure below) if you want to refer to them. With VLObject, you can store references not only to scene objects and their components, but also to prefabs and resource files, such as textures, sounds, and so on.



Important!: If you use an existing logic in another scene, object references will be lost, including links to prefabs, because the scene acts as their store. This also needs to be considered, if you plan to use the logic as a template, in this case, the best option would be to transfer the necessary links to it from the outside (for example, from a logic that is tied to a scene).

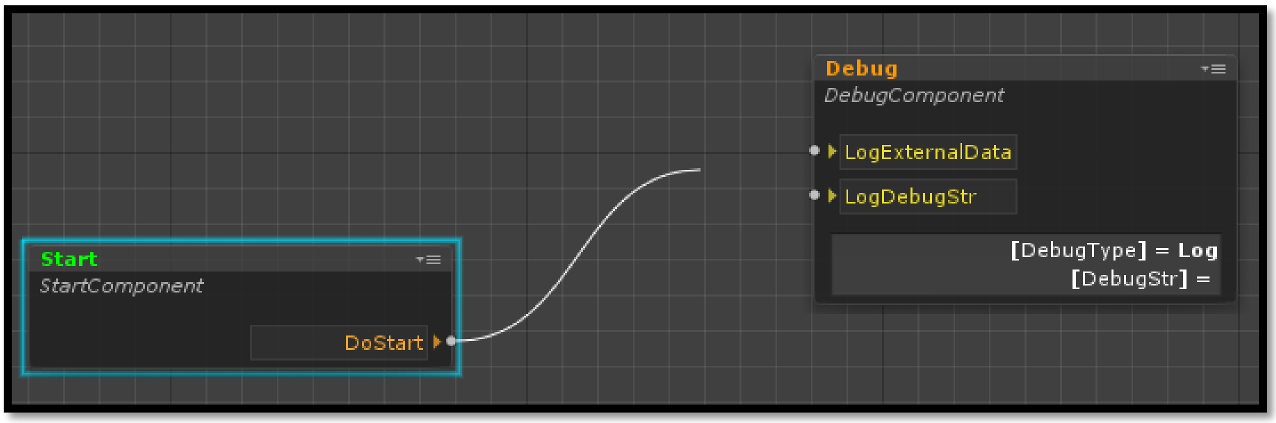
Links between components

Creating a link

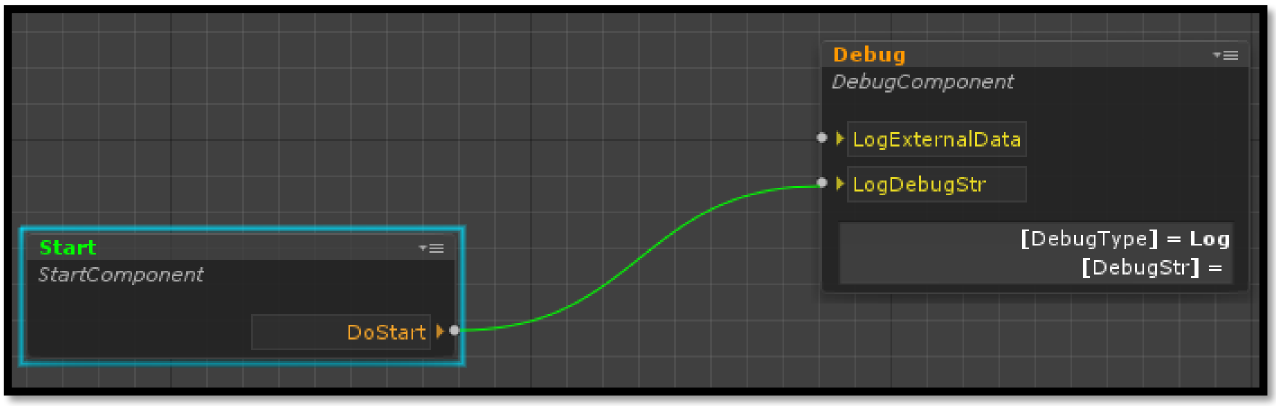
Previously, we determined how to create logic and how to fill it with components and variables. The last step is to create links between the components to determine the sequence of their work. The process of creating links is simple enough, but it has several rules:

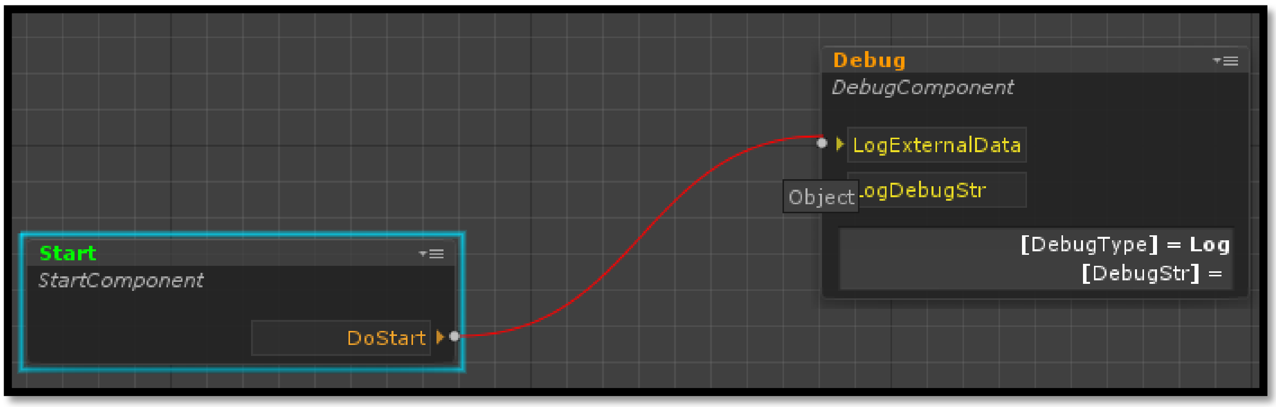
1. The link is always established between the output point of one component and the input point of another component (in a particular case, you can establish a connection between the points of the same component).
2. The link cannot be established between points transmitting and receiving data of different types.
   Note: You can establish a link if the input point is of type object or does not accept data at all.

To create a link, click on the output point with the left mouse button and without releasing it, start moving to the input point of the component (see the figure below).



When you move the cursor to the input point, in case the link can be established, it will be highlighted in green, otherwise it will be red (see the figure below).

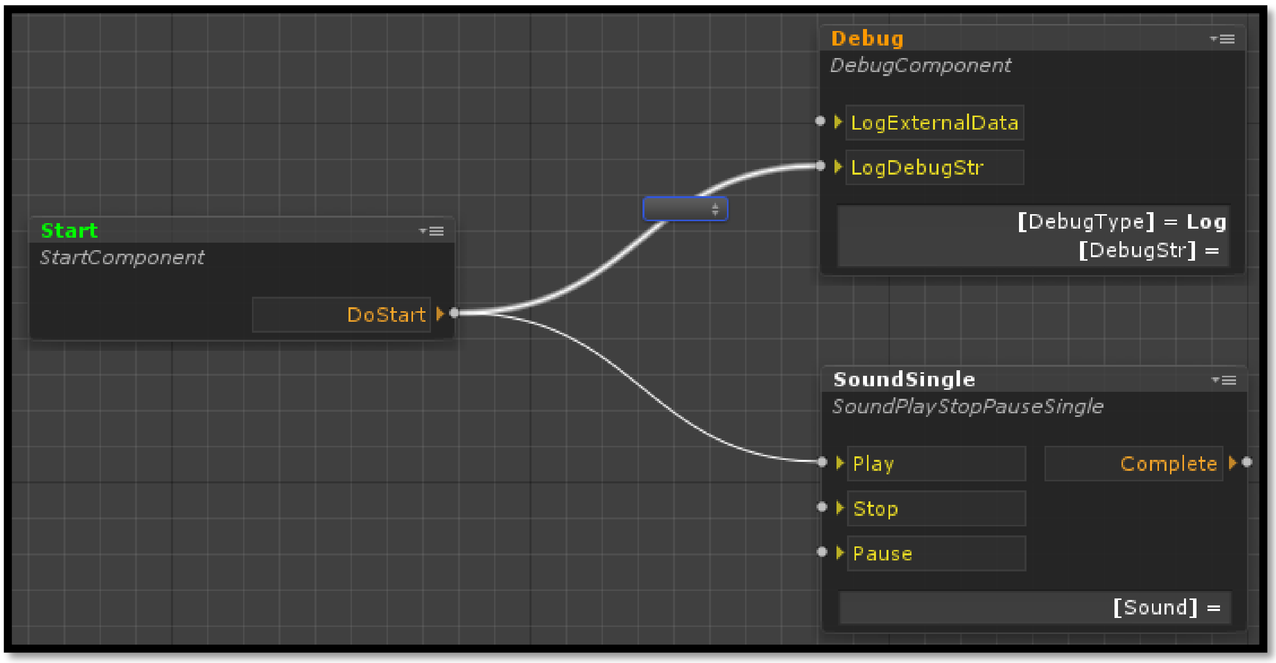




To create a link, you must release the mouse button when the line is highlighted in green.

Working with a link

Once the link has been established, we can conduct some operations with it. If we need to delete a link, we need to select it, just click the left mouse button on it (see the picture below).



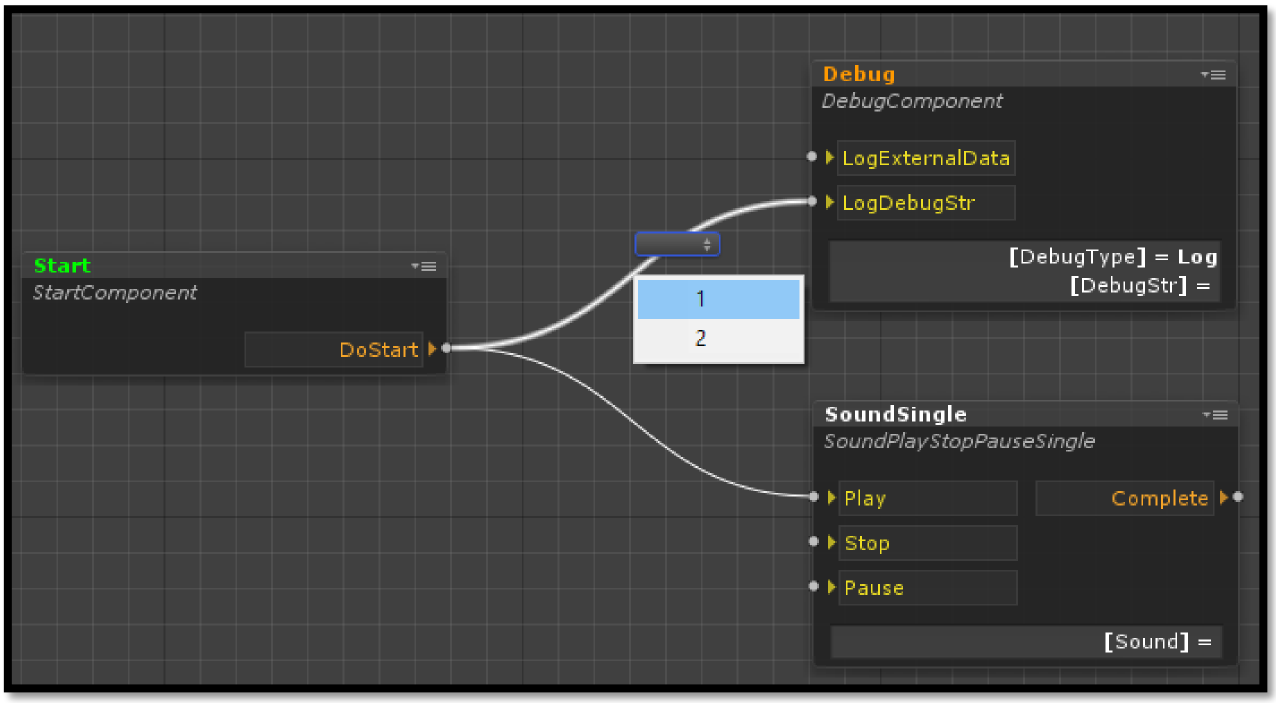
After the link is selected, you can either press the **Del** key to delete it, or use the context menu that is called by the right mouse button.

To select several links, perform the above manipulations, but with the **Ctrl** key pressed.

Note: if you delete an component, all links to it are automatically deleted.

Setting the links order

During the creation of a logic, often, one output point of an component can be connected to several input points of other components. In this case, it may be necessary to establish the order of execution of the links to ensure the correct functioning of the logic. To do this, select the sequence number in the drop-down menu that corresponds to the selected link (see the figure below).



Note: by default, all links have a sequence number of execution equal to one.

Logic recovery

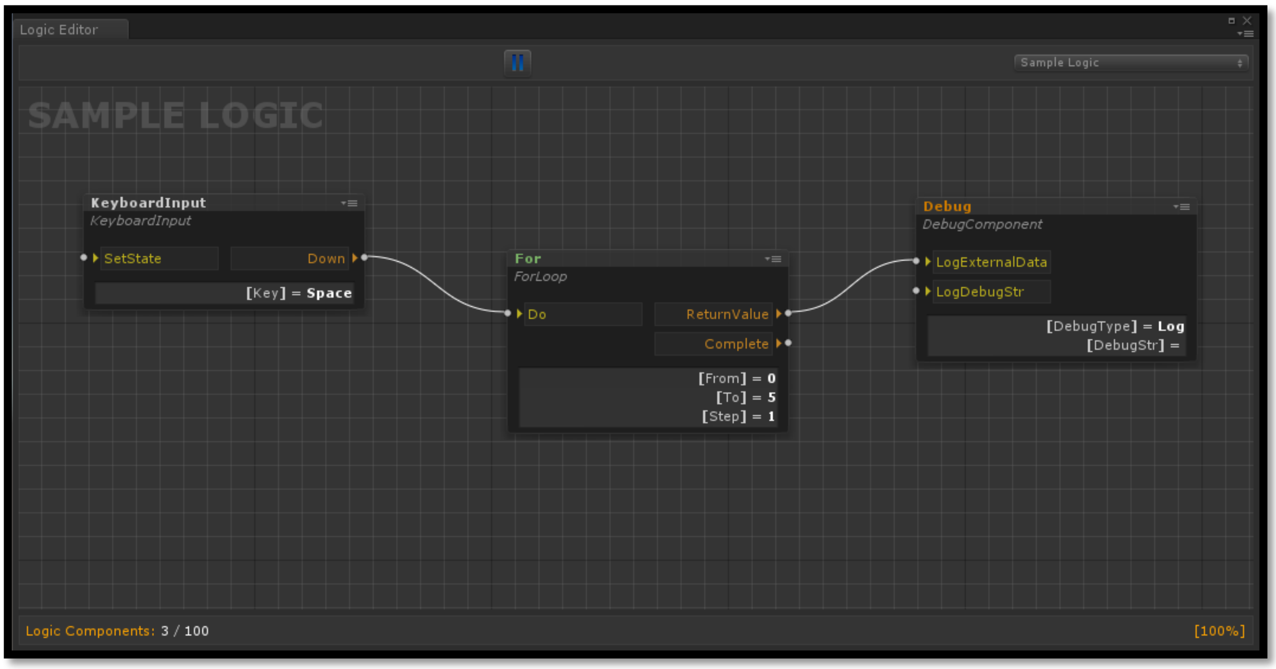
During the work on the application, there are possible force majeure situations of the loss of scene data. Since the logics are part of the scene, they will automatically be lost, but they can be recovered. There are two options to restore them:

1. Since the logics are files, you can add them to the scene again manually.
2. Use automatic logic recovery, with the menu button

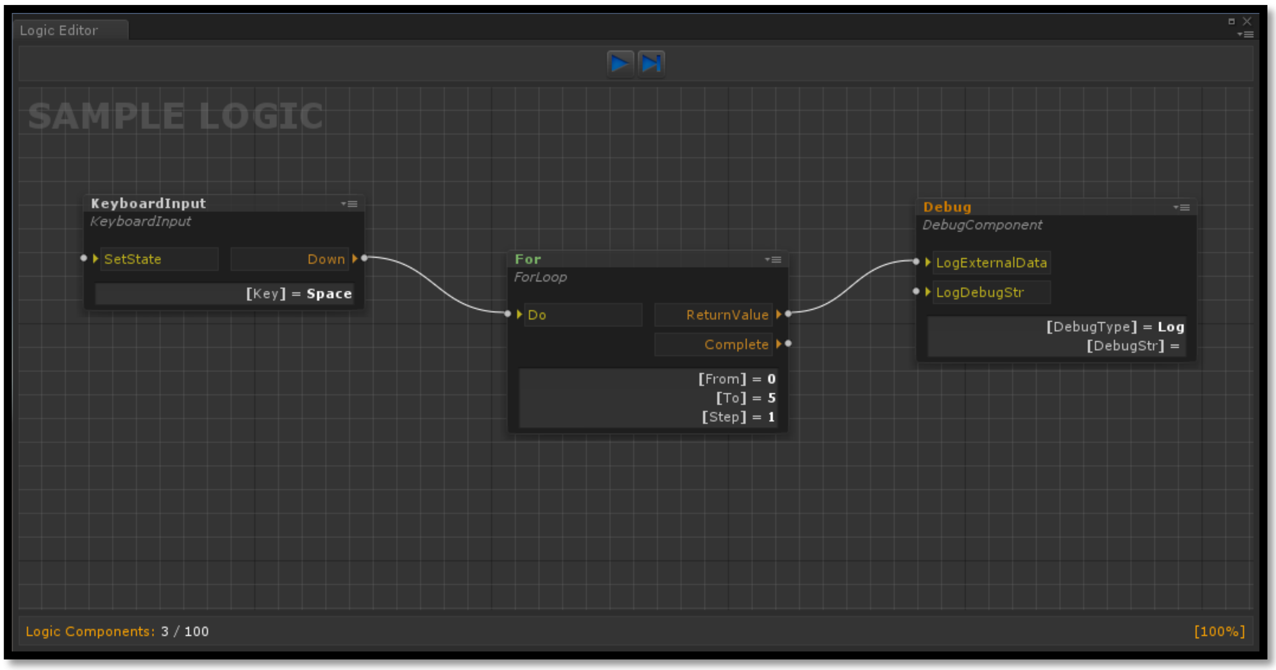
However, while restoring logics, there is one nuance that must be taken into account - if the logic contained variables with links to Unity objects, then all of them will be lost, **so create backups more often!**

Debugging a logic

To debug a logic, you need to start the application in editor mode, after which the editor view will change to this (see the figure below).



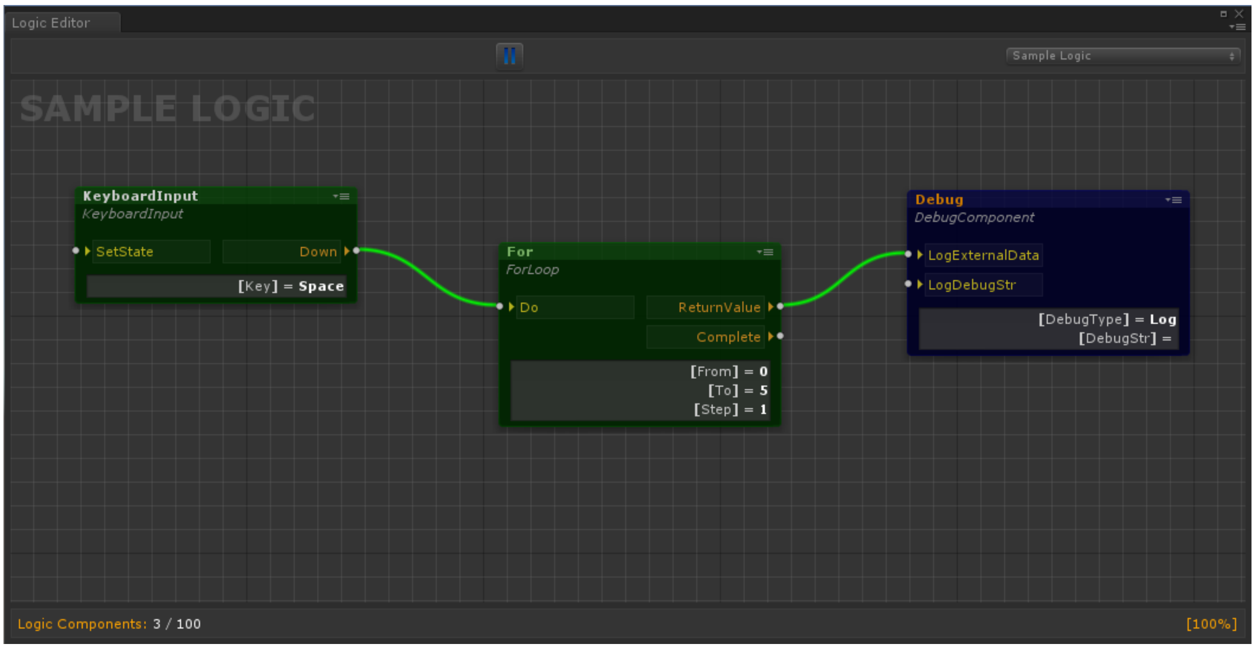
After clicked pause button, the editor goes into a mode of step-by-step calling of links.



Here:

1. Play button switches execution to the normal mode (without step-by-step debugging)
2. Step button executes the next step in the chain of link calls.

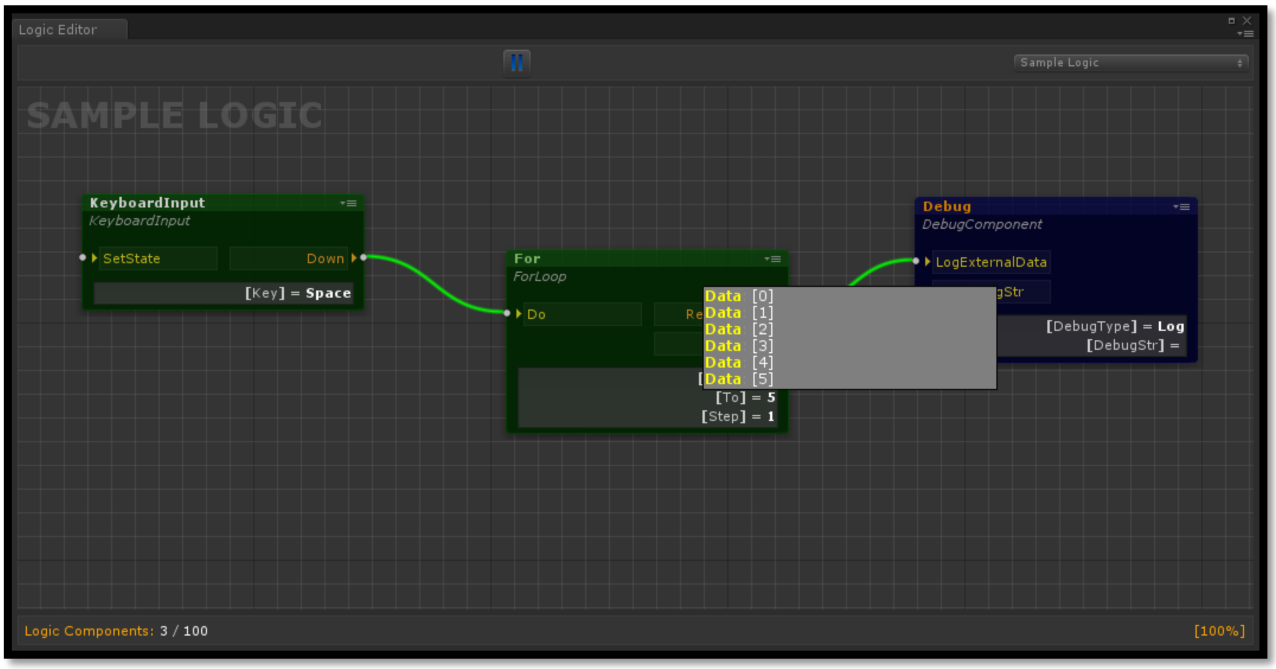
During the execution of the logic, components and links are highlighted, depending on the current state.



Here:

- The components that have completed their work are marked with a **green** color.
- The **blue** color highlights the components that are currently working, or were the last ones in the call chain.
- Blinking links show that their call is taking place at a given moment.

To see the data that the component has transmitted over the link to another component, it is necessary to highlight the link. Also, if the data that is displayed in the component parameters changes during the operation of the component, these changes will be reflected in the logic editor.



In addition to the above, there are some limitations of the editor in debug mode:

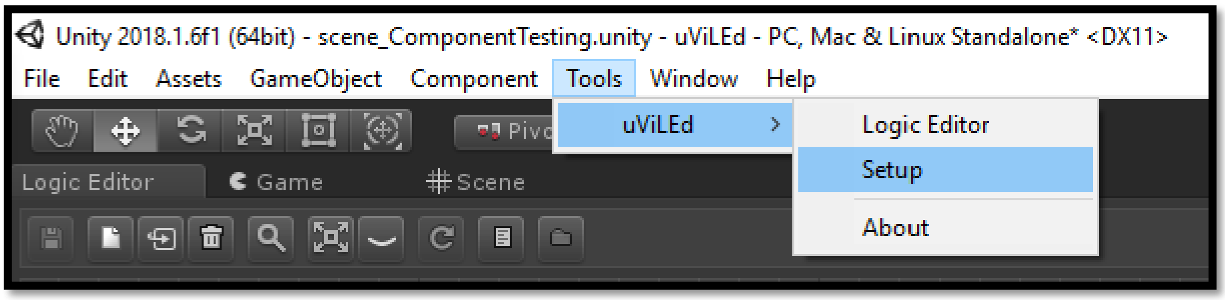
1. It’s not possible to add component s during debugging.
2. It’s not possible to change the parameters of an component.
3. It’s not possible to delete links and components.

4. Changing components positions will not be saved when you exit application launch mode.

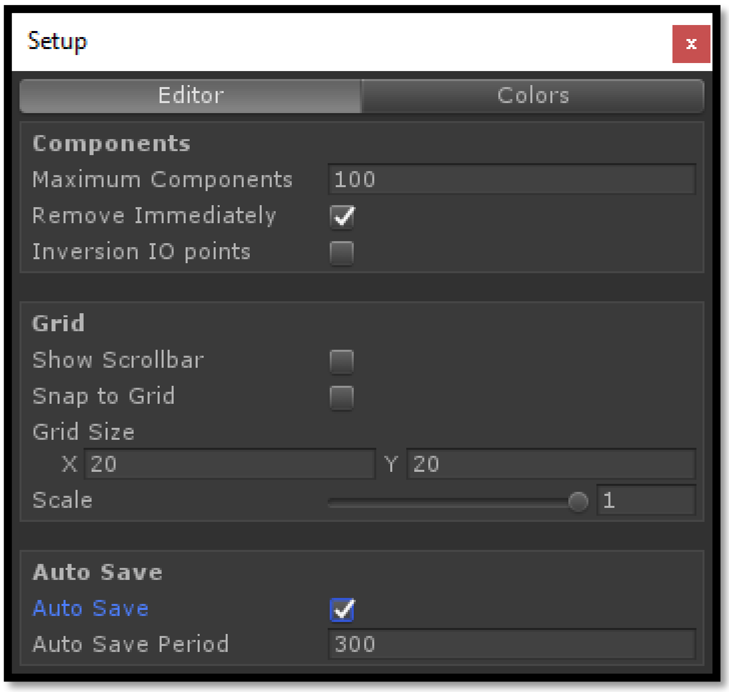
Also, you can use **Breakpoint** component for pausing the execution at specified place. Consult the components reference for details.

Settings

To open the settings window, use the menu item as shown in the figure.


Unity 2018.1.6f1 (64bit) - scene_ComponentTesting.unity - uViLEd - PC, Mac & Linux Standalone* <DX11>
File
Edit
Assets
GameObject
Component
Tools
Window
Help
uViLEd
Logic Editor
Setup
About
Game
Scene

In the settings window, there are two sections. The first is the settings section for the logic editor, the second is for adjusting the colors. The first one in the picture below.



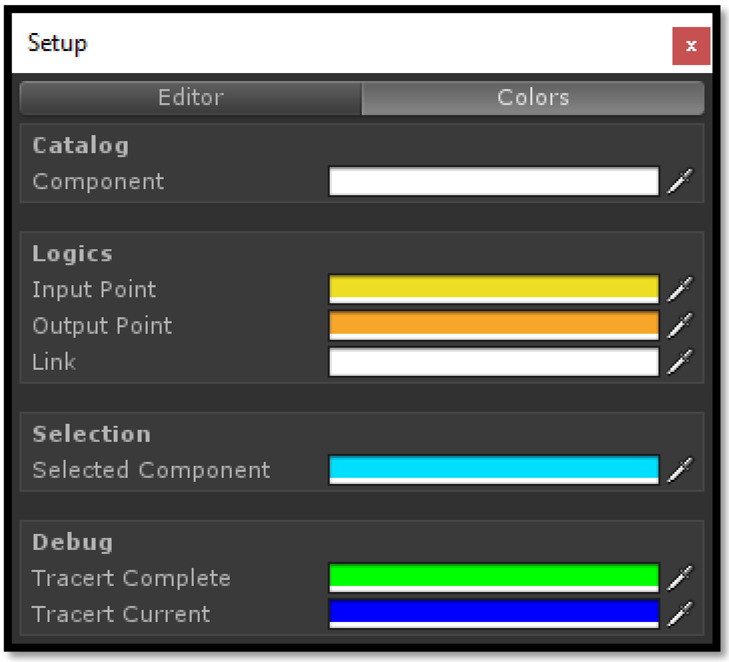
Components
Maximum Components 100
Remove Immediately ✓
Inversion IO points
Grid
Show Scrollbar
Snap to Grid
Grid Size
X 20 Y 20
Scale 1
Auto Save
Auto Save ✓
Auto Save Period 300

Here:

- **Maximum Componentss** – parameter that limits the number of components per logic (to control the size of the logics). If set to 0, there are no restrictions.
- **Remove Immediately** – parameter that is responsible for removing components without confirmation.
- **Inversion IO Points** – parameter responsible for inversion of the input and output points by default (when creating components).
- **Show Scrollbar** – parameter, responsible for displaying of the scrollbars in the logic editor window.
- **Snap to Grid** – parameter that is responsible for binding to the grid of components when they are moved.
- **Grid Size** – grid step.

- **Auto Save** – parameter that is responsible for the automatic saving of the logic.
- **Auto Save Period** – parameter that defines the period in seconds with which the logic is automatically saved.

The second section of the settings is devoted to color settings.



Catalog

Component	⬜

Logics

Input Point	🟨
Output Point	🟧
Link	⬜

Selection

Selected Component	🟦

Debug

Tracert Complete	🟩
Tracert Current	🟦

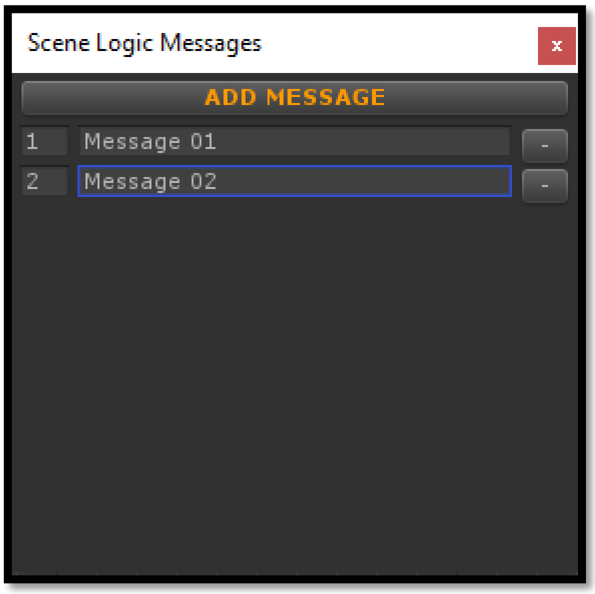
Здесь:

- **Component** – default component header color.
- **Input Point** – input point color.
- **Output Point** – output point color.
- **Link** – color of a link between components.
- **Selected Component** – selected component color.
- **Tracert Complete** – the color of the component that finished its work.
- **Tracert Current** – the color of the component that is currently executing its work or which was the last in the call chain.

Scene logic messages

uViLEd uses a system of global events. The complex part in which you can define these events and set the required data set is available for programmers through the API. In the logic editor, a simplified version is used, which is an add-on for a more general version.

Scene messages are events that are sent to all subscribers to them, if certain conditions are met (for example, pressing a key). These messages may transmit data or may be of a notifying nature. In both cases, to use this system in the logic editor, the set of messages should be created. This can be done in the messages window, that can be opened by pressing button in the main menu. The window is shown on the figure below.



Note: all messages in the logic editor are identified by an integer value, but in the components, that work with them, they are represented in a string form. This is done to be able to delete messages without consequences.

Sound

In the uViLEd system, there are several components responsible for working with sound. These components can play, stop and pause various sound sources. If you look at the components, you will notice that they all have a Channel parameter. A channel is a group of sounds that are played with certain conditions, rules and parameters. In this system, the definition of the channel is done on the base of AudioMixerGroup. If this parameter is not set in the sound source, the sound is automatically placed in the Default channel.