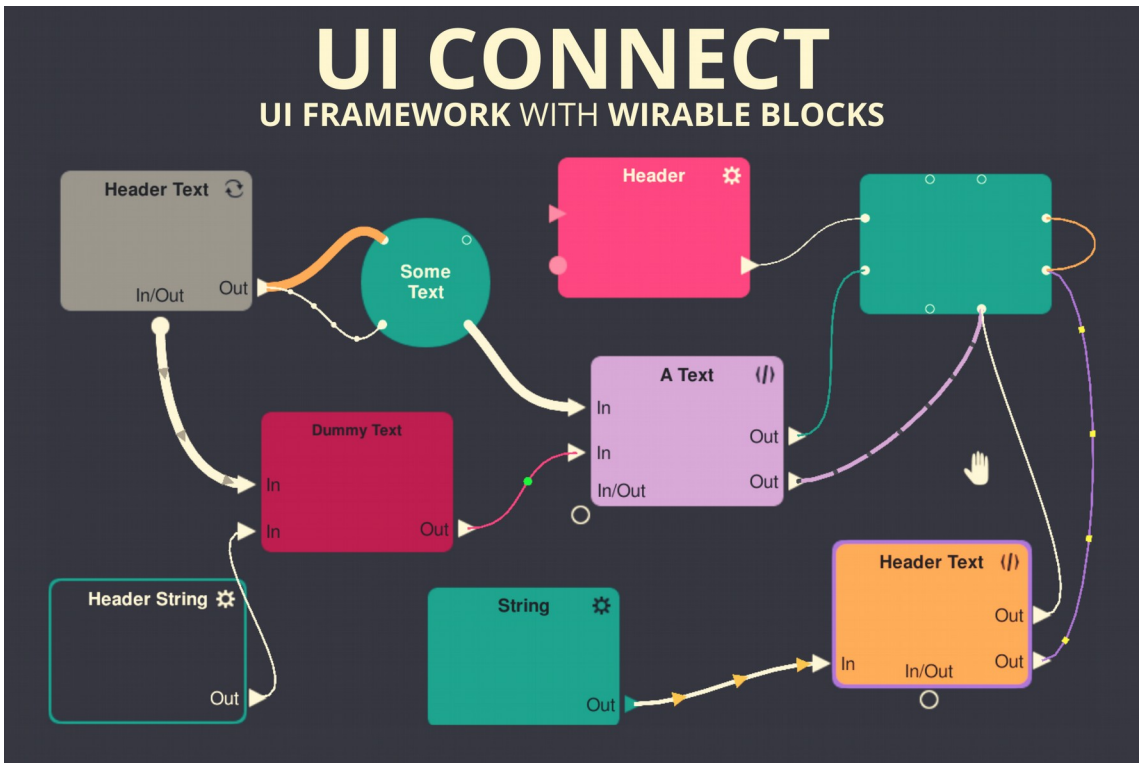
UI Connect



A **ready to play and easily customizable UI framework** with wirable drag/drop blocks and a custom UI line renderer to improve your game project with interactive interfaces as flowcharts, circuit boards, connection puzzles, among other creative gameplays.

This documentation contains information on the implemented features and code, examples and how to customize.

Contact

assetstore.unity.com/publishers/19535

github.com/danielcmcg

d.cavalcante.m@gmail.com

Daniel C Menezes
http://danielcmcg.github.io

1 CONTENTS

2 OVERVIEW

The **UI Connect (UIC)** is a UI framework to create connectable UI objects that are visually linked by lines or splines.

The system is composed of 7 main modules, **Manager, Pointer, Line Renderer, Entity, Node, Connection** and **Context Item**. It adds, to your project, the capability of making flowcharts, circuit boards, connection puzzles, among other creative gameplays.

It is possible to customize the asset with your own sprites, texts, colors, buttons and improve the user interaction by adding scripts that use the UIC Interfaces or create scripts to implement new features calling the main components' methods, variables and properties, as:

- List<UIC_Entity> GetConnectedEntities() method from UIC_Entity
- List<UIC_Connection> ConnectionsList property from UIC_Manager
- List<UIC_Connection> GetCrossedConnections() method from UIC_Connection.
- bool ConnectionsIntersect(UIC_Connection conn1, UIC_Connection conn2) method from UIC_Utility.

In the following sections, the system's modules and example scenes are explained.

3 MODULES

The system is composed of UI modules that enable the connectable blocks for a particular canvas.

It is possible to have multiple UIC views with connectable objects in the same scene, each with a combination of the UIC modules (ex.: a game menu section where each tab has its own chart).

For the system to work, each canvas that will be used as UIC must be set hierarchically as:

- UIC_Manager + Canvas
  - UIC_Pointer
  - UIC_LineRenderer
  - UIC_Entity
    - UIC_Node

3.1 MANAGER

The UIC_Manager is a component attached to a Canvas, it contains reference for all the other system's components in the scene as well as the main methods to manage the user actions and the components along with their interactions.

Once enabled, the manager automatically sets, or create if it does not exist, the UIC_LineRenderer, a needed component of the system to enable the connections to be drawn.

Its prefab already contains the UIC_LineRenderer, a UIC_Pointer and some example entities.

3.2 POINTER

The UIC_Pointer component is responsible for manage the click and drag/drop actions.

This component also has the "default" and "hold" icons for the pointer if needed (if it is not necessary to have a icon for the pointer, make the pointer size equal zero).

3.3 Line Renderer

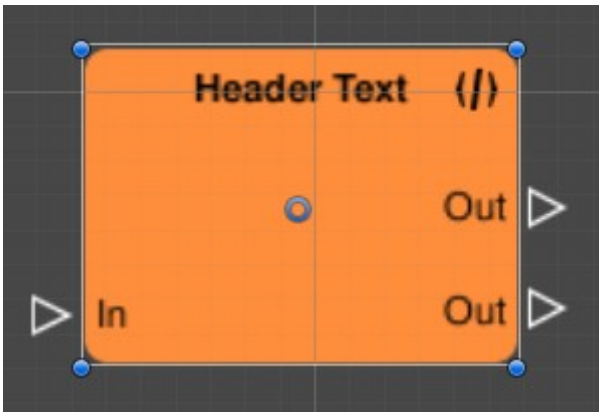
The UIC_LineRenderer is a UI Graphic component responsible for drawing all the lines. It stores all the line references and their points using the UIC_Line class.

It can be used separately from the other components to draw any line by passing a new UIC_Line to the List<UIC_Line> uIlines. The line can be set by its API.

It is also possible to generate points for an Spline and pass them to a line by using the UIC_Spline class.

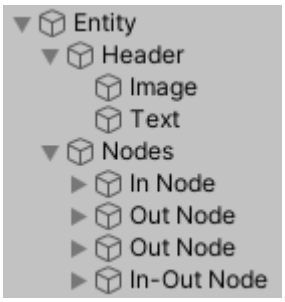
3.4 Entity

The UIC_Entity is an object that can have nodes and be connected to other entities.



An Image component is used to set the sprite of the object and an Outline component is enabled when the entity is selected.

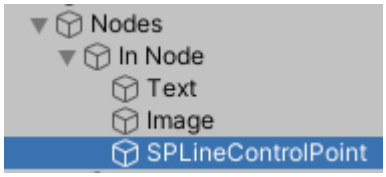
It also can contain child UIC_Node objects as children (the connection slots). On editor mode, nodes added to the hierarchy are automatically added to the entity's node list (List<UIC_Node> nodeList).

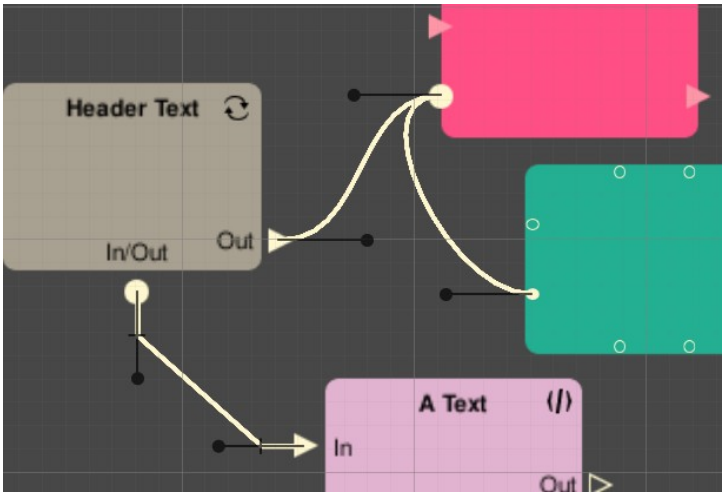


3.5 NODE

The UIC_Node is the slot from/to lines can be dragged to create a connection between entities. This component holds a list of its own connections, making it possible to know the entities connected to this particular node or entity.

Each Node contains a child called SPLineControlPoint. For SPLines, this control point will indicate the line curvature, while for z-shape lines, the middle point between the node and the control point defines where the line turns.





3.6 CONNECTION

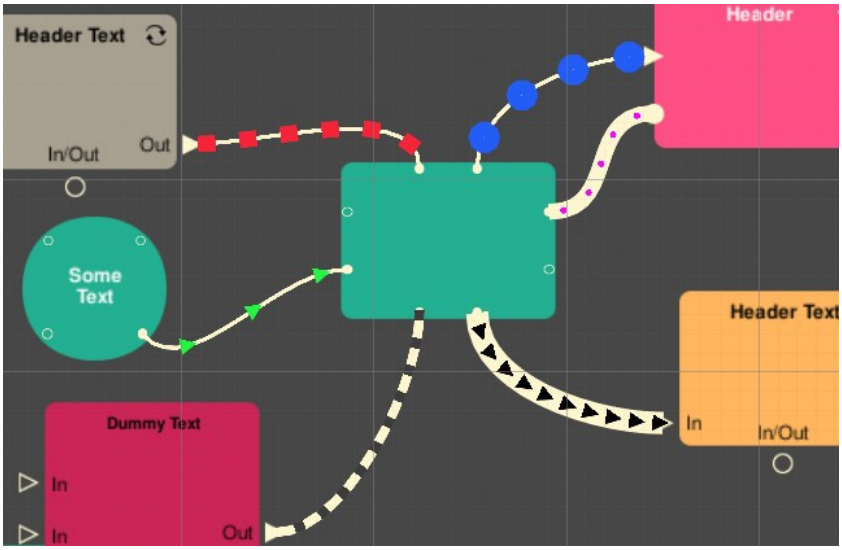
The UIC_Connection is the component that visually connects the nodes with lines. It holds a reference for the UI line that is rendered by the UI line renderer, along with its width, color and shape (that can be a straight line, a spline or a z-like shape).

3.6.1 LINE AND LINE ANIMATIONS

Each connection contains an associated line (UIC_Line). the connection line can be customized in terms of: width, color, cap (type, size, color, angle) and animation. By setting those variables, you can have connection lines with different aspects.

The animation aspect of a UIC_Line is set from the “animation” variable, of type UIC_LineAnimation. It has variables of its own (isActive, type, pointCount,

DrawType, size, color and speed) that can be adjusted to achieve different animation patterns.



3.7 Context Item

The UIC_ContextItem is a component that makes possible to create buttons, sliders, color pickers and more types and update their values on pointer click.

4 CUSTOMIZING

It is possible to customize the system in a variety of ways depending on the project's specificity. In this section, there are starter examples on how to customize parts of the system.

More examples on how to use the system are the actual example scenes.

4.1 STARTING THE ENVIRONMENT WITH INITIAL SETTINGS

The environment of the system is a UI Canvas with a UIC_Manager component attached. It also need a UIC_Pointer and a UIC_LineRenderer game objects as children there are going to be automatically added to the Canvas once the Manager component is added.

To set initial values for new objects, you may call the Manager API, calling the Manager Instance (UIC_Manager.Instance), then finding the variable to set.

The code below, is an example taken from the ConnectionPuzzle_GameManager script:

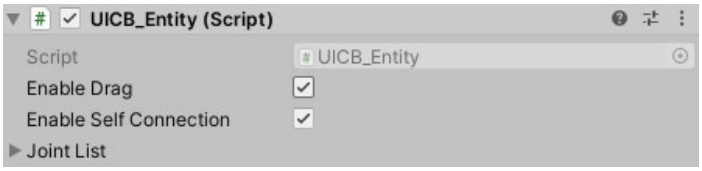
```
void Start()
{
    // set the global line type
    UIC_Manager.Instance.globalLineType = UIC_Manager.LineTypeEnum.Line;
    // disable click on connections
    UIC_Manager.Instance.disableConnectionClick = true;
...
}
```

4.2 CREATING AN ENTITY

The **Entity** Prefab, located at "Prefabs/Resources/Entity.prefab", can be used as a template by adding it to the **Manager Canvas**.

It is also simple to create an **Entity** from scratch using a UI object:

1) Add a UI object that contains an Image (Image or Panel are good choices, but you can also use other UI objects) to the Manager Canvas.
2) Add a UIC_Entity component to the created object (It will automatically add an Outline component as well).
3) Set the "Enable Drag" and "Enable Self Connection" toggles as desired.



4.3 Adding Nodes to Entities

The **Nodes** prefabs, located at "Prefabs/Resources/Node.prefab", can be added as children of **Entities**.

From the UIC_Node component, is possible to set:

- **(enum) Polarity Type {In, Out, All}**: Defines the connection flow, In Nodes can only be connected with Out, while All can be connected with In or Out.
- **(int) Max Connections**: Maximum number of connections permitted (setting it to 0 makes it with no limit).
- **(sprite) Icon Connected**: Sprite set when there is at least 1 connection.
- **(sprite) Icon Unconnected**: Sprite set when there is no connections.
- **(color) Icon Color Default**: Default icon color, also the color when there is no connections.
- **(color) Icon Color Hover**: Icon color when the mouse gets closer to the Node.
- **(color) Icon Color Selected**: Icon color when the Node is clicked and hold.
- **(color) Icon Color Connected**: Icon color when there is at least one connection.
- **(transform) Spline Control Point Transform**: The transform that is used by the connections as the line control point (Used by spline and z-like line types).

4.4 Adding a Context Item

Context Items are components needed in order to use action UI objects (using the Unity's UI event system) as buttons, sliders, color pickers and other types that interact with the system and are updated when the pointer is clicked.

An example is the UIC_ButtonRemoveSelected, that removes the selected (only for selectable) object(s) of the scene, implementing I_UIC_ContextItem and I_UIC_Object.

```
public class UIC_ButtonRemoveSelected : MonoBehaviour, I_UIC_ContextItem, I_UIC_Object
{
    Button _button;
    public string ID = "button";
    public Color objectColor { get; set; }
    public int Priority => -10;

    public void Action()
    {
        for (int i = UIC_Manager.selectedUIObjectsList.Count - 1; i >= 0; i--)
        {
            UIC_Manager.selectedUIObjectsList[i].Remove();
        }
        UIC_ContextMenu.UpdateContextMenu();
    }

    void Awake()
    {
        _button = GetComponent<Button>();
    }

    void OnEnable()
    {
        _button.onClick.AddListener(Action);
    }

    void OnDisable()
    {
        _button.onClick.RemoveAllListeners();
    }

    public void OnChangeSelection()
    {
        gameObject.SetActive(UIC_Manager.selectedUIObjectsList.Count > 0);
    }

    public void Remove()
    {
        Destroy(gameObject);
    }
}
```

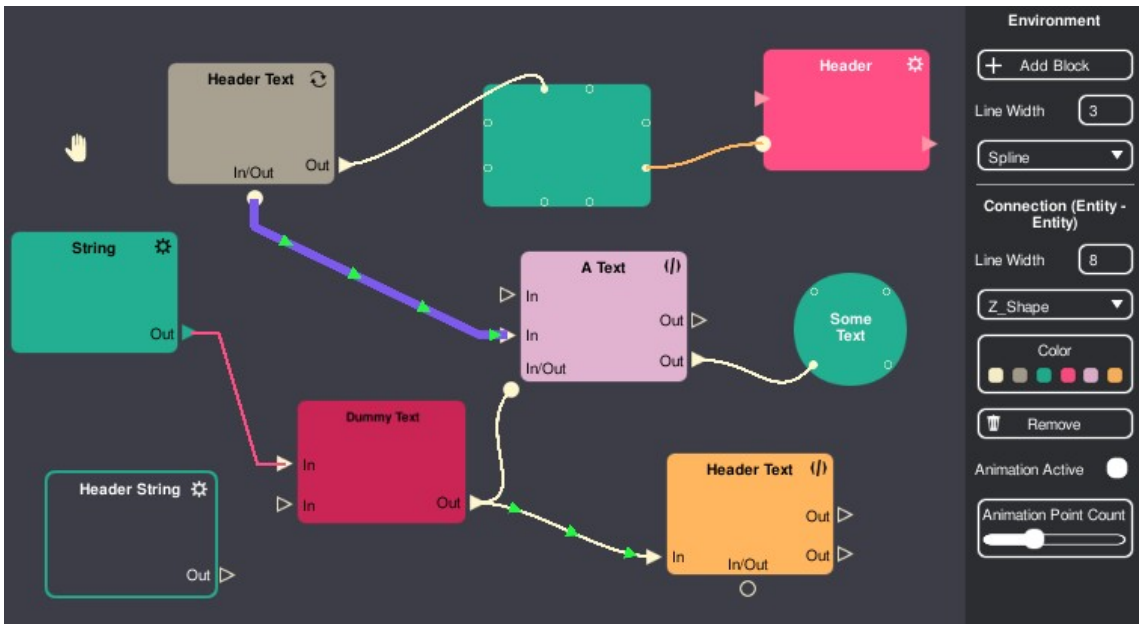
5 SAMPLE SCENES

The sample scenes are examples of how to use and what can be done with the **UI Connect** asset, however, it is not limited to those examples or functionalities, since you can create your own scripts using the components API or adding to the current code.

5.1 FLOWCHART

The Flowchart scene contains a group Entities as examples that can be freely dragged around or connected with each other. It is also possible to see which object is selected and change some of its aspects using the right context menu.

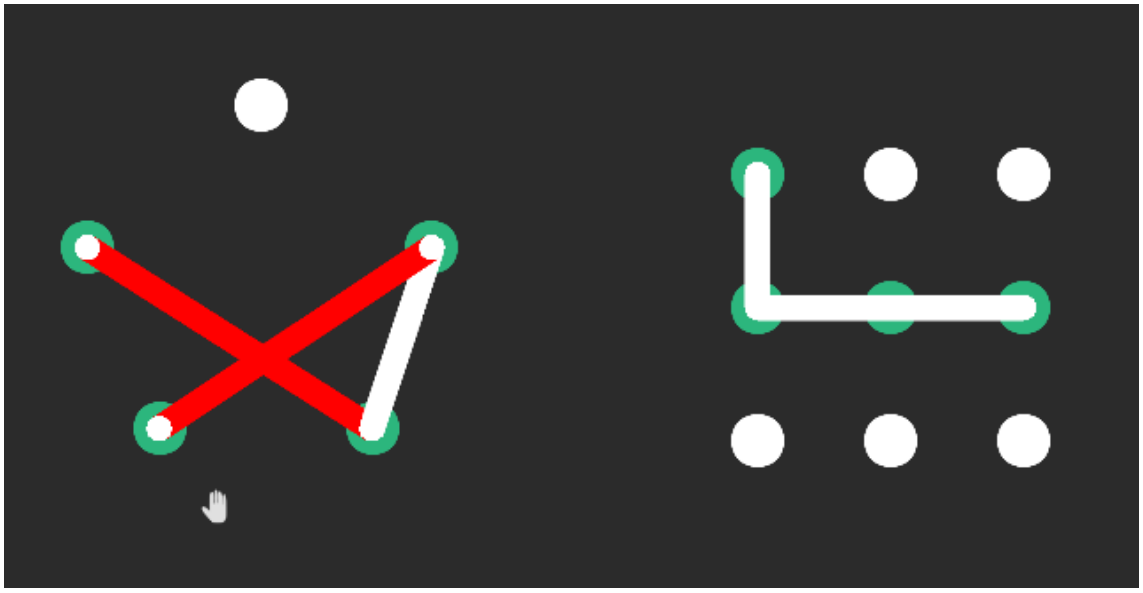
It shows some customization possibilities on Entities and Nodes, also the customization of the connections and examples of possible Context Items as Button, Input Field, Popup and Color Picker.



5.2 CONNECTION PUZZLE

This scene is an example of the usage of the UIC components API, where you drag the pointer around to connect all dots. It detects if a line crosses another line or a dot and highlight the crossing lines in Red.

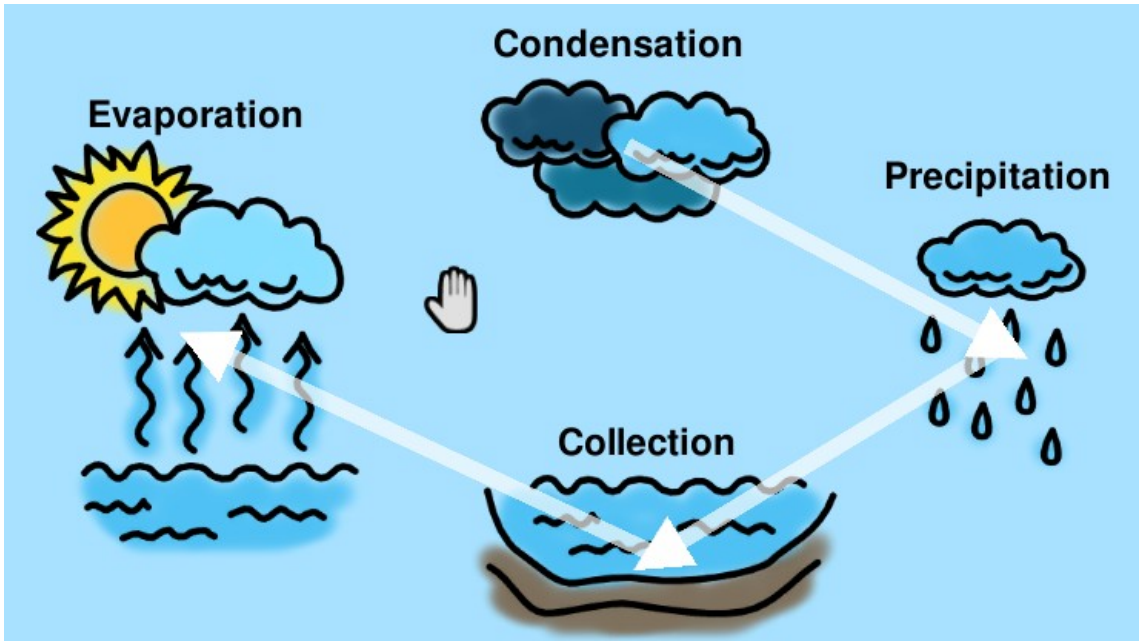
Its manager class calls the UIC_Manager to set up the environment with initial values and, combined with UIC_Pointer events and methods, enable the continuous drag for selecting the dots and check for the crossed lines and nodes.



5.3 Connect the Life Cycles

This example is a type of puzzle directed to kids where the user needs to connect the cycle in the correct order, it con be done for life cycles, seasons or other cycle examples.

Its manager class calls the UIC_Manager to set up the environment with initial values and, combined with UIC_Pointer events and methods, check for the current connections to replace/update them and to check the closed cycles.

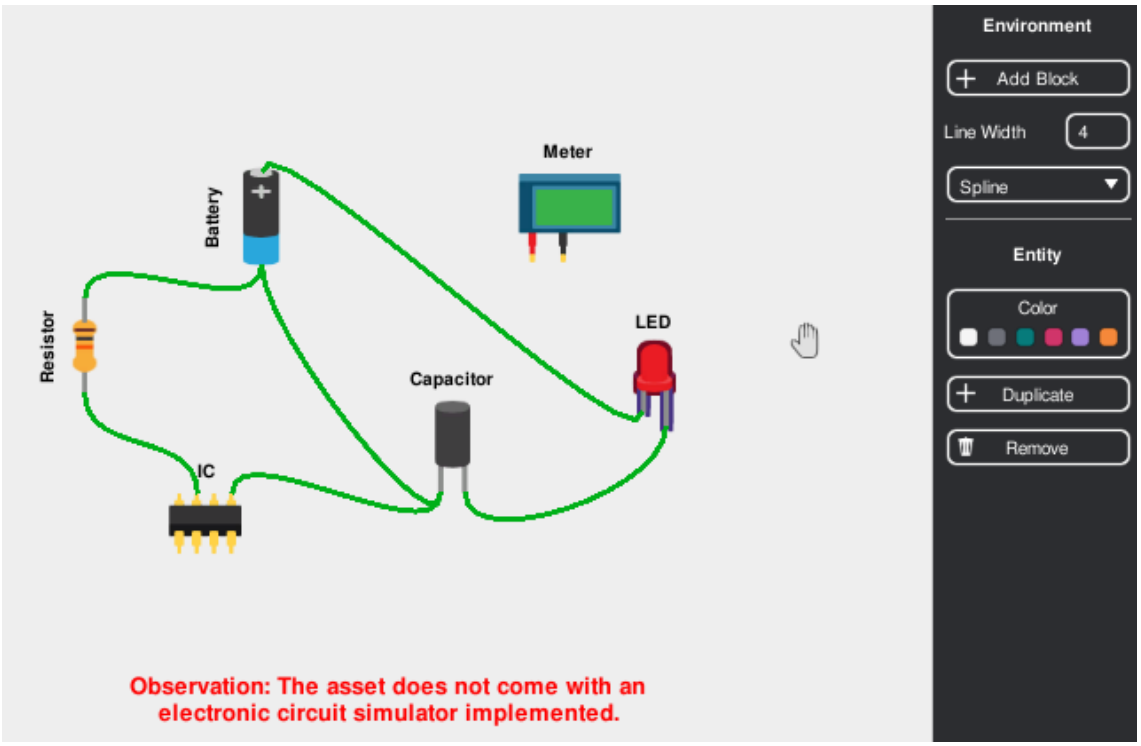


5.4 Circuit Board

This scene is only a framework for a drag and drop circuit board, where you have Entities as circuit components, and Connections as wires.

Observation: The asset does **not** come with an electronic circuit simulator implemented.

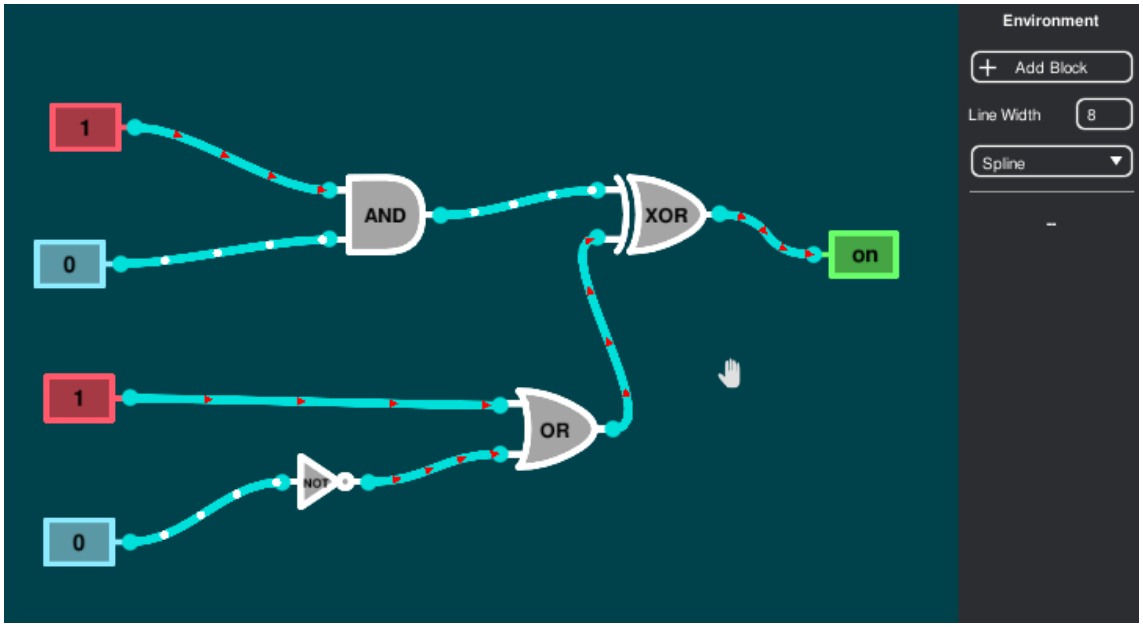
However, as a suggestion, this can be achieved with some work on integrating or coding a simulator library to the project and modeling the needed circuit components. Making the Entities to be circuit components, the Connections, wires and each Node still be circuit nodes, it is possible to model the components accordingly and pass their parameters to construct a circuit model to be analyzed.



5.5 Logic Gates

In this scene, it is possible to connect simple logic gates and verify the logic output from the “End” entity color and the connection line color and animation.

It uses a really simple solve method just as inspiration on using the asset. Feel free to use a more complex solver and expand the project.



5.6 Multiple Charts

This scene show the possibility of using multiple UIC views, where each top button (System 0 to 2) enable the respective canvas and disable tho other ones.

The view maintain the connections if it is disabled and reenabled.