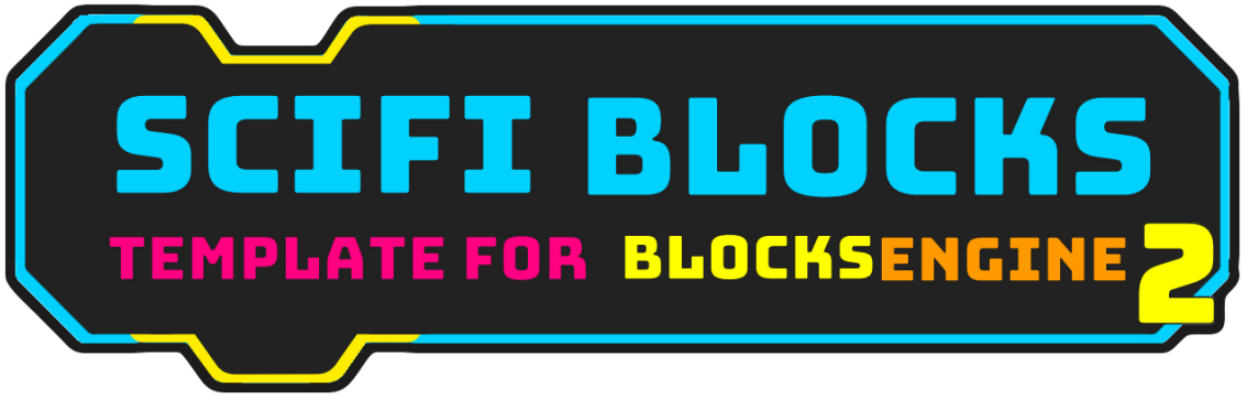


Add the **Scifi Blocks Template** to your Blocks Engine 2 and change the atmosphere of your game. Friendly and familiar, based on the most popular block programming languages. Perfect for Educational, puzzle or any type of Game.

This Scifi Blocks Template for Blocks Engine 2 asset documentation contains the detailed explanations for the implemented features and how to set up the template.

This asset requires Blocks Engine 2

Contact

Asset Store Page

Github

support@meadowgames.com

MeadowGames
MeadowGames.com

CONTENTS

1. OVERVIEW

The Scifi Blocks Template for Blocks Engine 2 is an expansion for the original Blocks Engine 2 that adds a Scifi look to the blocks and the environment. **It is important to note that this template asset needs the Blocks Engine 2 as base for it to work properly**.

The asset expands the possibilities by making the environment suitable for different game ambiances. All the original blocks are already setup with the Scifi template as well as the inspector's Block Builder, which can be used to create new blocks with the same look.

2. Getting Started

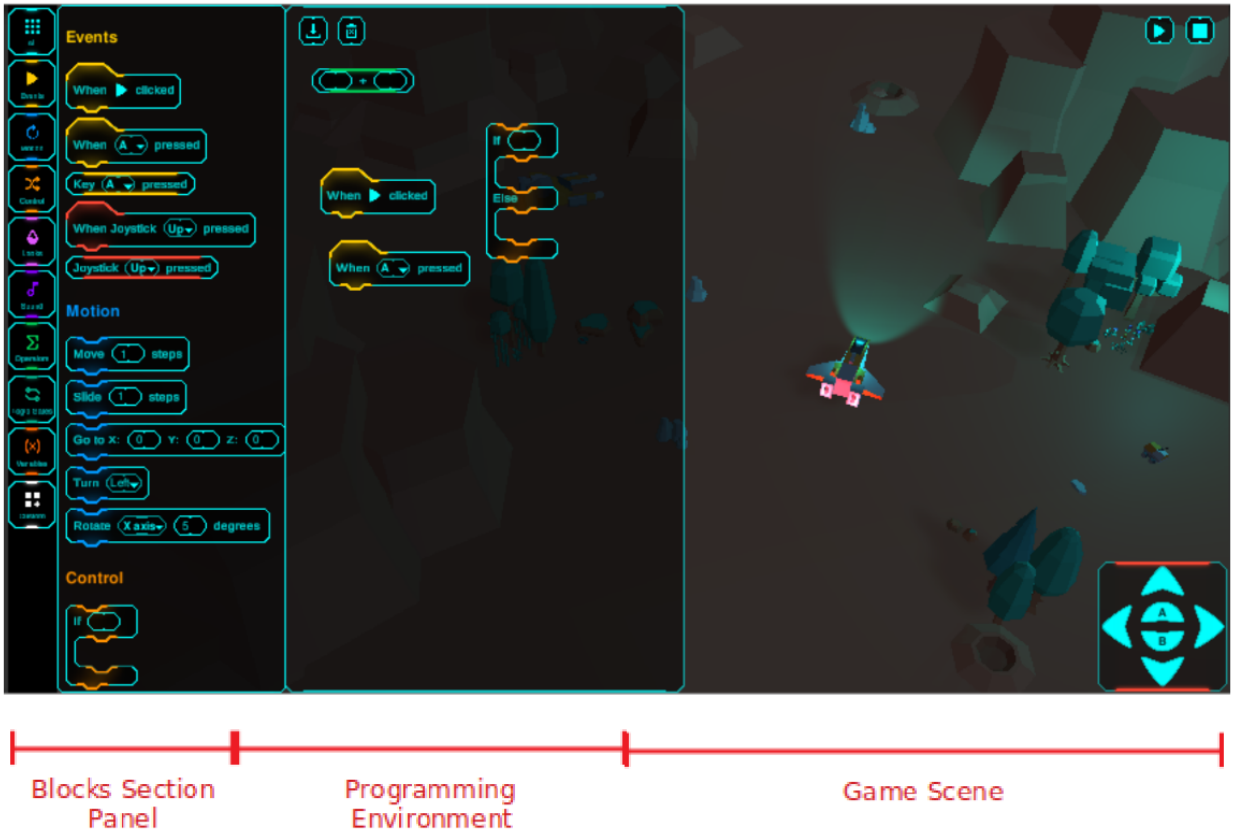
To start using the Scifi template, follow the steps below:

1. Previously **import the Blocks Engine 2**;
2. Make sure the **Text Mesh Pro package is installed**;
3. **Import the Scifi Blocks Template for Blocks Engine 2**;
4. **Open the Scifi sample scene** at: Assets/BlocksEngine2_SciFi/Scenes/ScifiBE2SampleScene.unity.

3. User Interface

The interface contains the same elements as the original Blocks Engine 2, the Blocks Selection Panel, Programming Environment and the Game Scene. Below is a print of the sample scene.

See the Blocks Engine 2 documentation for more.



The user interface can be customized to fit your unique project by changing the layout, size, position and other visual characteristics, as well as expanding the functionalities and adding game mechanics.

It is also possible to create more blocks and instructions and customize the blocks characteristics.

3.1. Blocks Layout

The Blocks have a similar layout structure as the original asset, the same interfaces are used, making them compatible to be used along with the original blocks if you want.

The Base sprites are different, with transparency and straight corners instead of rounded ones. The main difference is the addition of a neon like sprite that has the color of the block type.

3.1.1. Categories

Each category of the Block Selection View is related to the general behavior of the instruction.

- **Events:** Trigger blocks that start a sequence of blocks and operation blocks that function as event listeners. Colors #FFCC00 and #FF4835;
- **Motion:** Blocks related to movement, positioning and direction of the Target Object. Color #0099FF;
- **Control:** Blocks that can alter the run cycle, repeating the cycle of a blocks group, delaying or conditioning the execution of specific blocks group. Color #FF9200;
- **Looks:** Blocks that alter the visual aspect of the Target Object. Color #E056FD;
- **Sound:** Blocks that execute sounds. Color #8802FF;
- **Operators:** Operation blocks that perform math and comparative operations used as inputs. Color #00C855;
- **Logic Gates:** Operation blocks that perform logic operations used as inputs. Color #00AB89;
- **Variable:** Blocks that perform variable related operations, act on variables, variable manager and the variables (operations) used as inputs. Color #FF6600;
- **Custom:** Additional blocks later created and blocks that act on specific Target Objects. Color #FFFFFF.

3.1.2. Blocks

The blocks are the visual representation of the code functions, operations or variables. The user can place the function blocks in a logic sequence to build a code and insert operation blocks (and variables) as inputs in other blocks.

There are five block types (Trigger, Simple, Loop, Condition and Operation), those types can be identified by the shape of the block and for its behavior. All types except Operation are Function blocks.

Trigger

Blocks that begin each program, sequence of blocks and can behave as event listeners. They indicate how and when the program will start running.

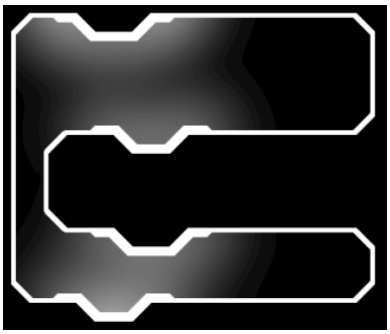
Simple

These blocks are usually responsible for the main actions of the program and execute its function once per cycle, they don't wrap children blocks.

Loop

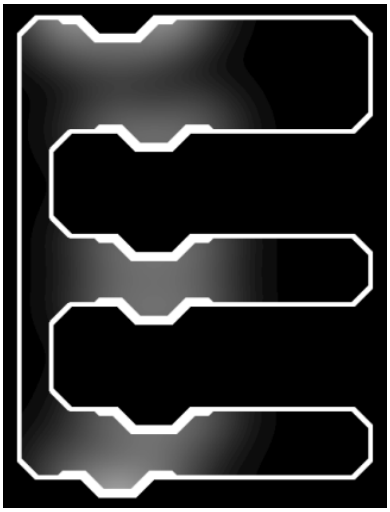
Blocks that execute loop instruction, they wrap children blocks that are going to be executed if the loop condition is still met. These blocks can have multiple sections, as an example of the If/Else block.

This type of block is by default executed once per frame due to their repetitive behavior, meaning that its ExecuteInUpdate setting of its Instruction is always true.



Conditional

Blocks that execute conditional instruction and wrap children blocks that will be executed if the condition is met. They also can have multiple sections.



Operation

Blocks that serve as inputs to be placed on the header of Function blocks, they execute operations and return a string value as result.

3.2. Game Scene

The Game Scene is where the code results are exposed to the user. In a 2D or 3D scenario, the Target Object performs the code instructions and can interact with other environment elements.

A virtual joystick is also present in that view, it can be used in composition with specific blocks for listening to the buttons.

There are a great number of possibilities for composing the Game Scene and letting your creativity flow, some examples are coding tutorials, puzzles, top down, combat, racing, coin catch games.

4. SCIFI BE2 CUSTOM COMPONENTS

The Scifi blocks use the same interfaces as the default blocks, however, it uses two components that are customized specifically for the Scifi blocks, those are:

- **BE2_ScifiBlockSectionHeader** : I_BE2_BlockSectionHeader
- **BE2_ScifiBlockSectionBody** : I_BE2_BlockSectionBody

The custom section components for the Header and Body are used to make it possible for the blocks to have a base color (cyan) and a neon detail with the block color. For this, the blocks' headers and bodies have a GameObject called "Blur" that contains an Image and a **BE2_NeonBlockColor** component.

5. Scifi Block Builder (BE2 Inspector)

The Block Builder in the Scifi sample scene is set up with the needed prefabs for creating Scifi blocks. The creation of blocks is done by the setting of variables and the use of the custom header markup to indicate the block header items.

Mind that after the creation of the block, the correspondent instruction should be implemented using C# and making use of the recommended instruction API.

For details on how to build Blocks and more, refer to the Blocks Engine 2 documentation.