THIS IS A BETA FEATURE

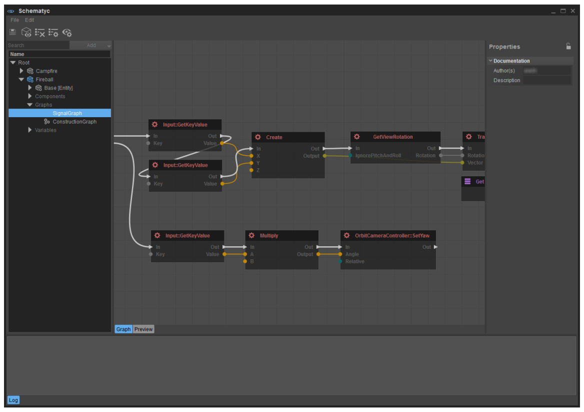
This feature is still in beta and subject to constant change. We encourage you to use it in test projects and provide your feedback to us.

However, **DO NOT** use it in production where it creates dependencies! Always back up your projects to make sure that you can go back to a previous version.

The primary goal of Schematyc is to change the way we build gameplay systems at Crytek, giving designers the power to construct new and reusable objects from a set of building blocks provided by programmers. At first glance, the editor looks a little like Flow Graph, but the two systems were built with very different purposes in mind. Whereas Flow Graphs are great for level scripting, Schematyc is designed to provide a finite control of the objects within those levels.

Visual scripting tools let you script your own logic. State machines help to break up and structure logic, making objects easier to debug and simplifying the process of network synchronization. Determinism and reduced latency make it possible to take new gameplay systems beyond the prototyping stage without the need to re-write them in C++.

Context is king in Schematyc and our aim is to *only* present users with relevant information, keeping the UI from becoming too cluttered and overwhelming.



- Basic Concepts
- The Transition Graph
- Low Latency, High Determinism
- Schematyc Assets Explained
    - Schematyc Entity
    - Schematyc Library
- How to create a Schematyc Asset
    - Asset Browser
    - Schematyc Editor
- How to Add Content to an Asset
- Intro Video

For more information on the Schematyc Editor, please refer here: **Schematyc Editor**

Basic Concepts

The basic concepts you need to understand before using Schematyc are:

- **Classes** - **https://en.wikipedia.org/wiki/Class_(computer_programming)** - A class describes an object and encompasses all of the data and logic associated with that object.
- **Components** - Components are the building blocks that define what functionality is available to an object. If you want to load geometry you will need to add a geometry component to your object, to read input from a controller you must first add an input component, and so on.
- **Variables** - **https://en.wikipedia.org/wiki/Variable_(computer_science)** - Variables store information and exist for as long as the object or state they belong to exists. By making a variable public you allow other systems access to that variable e.g. when you place a Schematyc entity in a level you can edit its public variables in the main Properties panel.
- **Finite State Machines** - **http://en.wikipedia.org/wiki/Finite-state_machine** - In Schematyc, states can be structured hierarchically and all the children of a particular state inherit behavior from that state, allowing them to share both functionality and transitions.
- **Signals** - Signals are used to communicate between objects and object states, so, for example, you could create a damage signal to send damage from one object to another. Signals can also be used to send data, so in the example of the damage signal, you might add two variables, damage type, and damage amount, in order to specify the damage being dealt.
- **Functions** - **https://en.wikipedia.org/wiki/Subroutine** - Functions in Schematyc are just like functions in any other programming language; you pass inputs, the function performs some logic, then you return the output.

- **Enumerations** - If you want to list a finite set of options with string identifiers you use an enumeration.

The Transition Graph

In addition to more traditional logic graphs for visual scripting, Schematyc also introduces the concept of a transition graph. This is where we control how and when you switch from one state to another. It is impossible to trigger transitions from anywhere else and while this may not seem quite as convenient as simply being able to link up a 'switch state' node, it makes it much easier to view the overall picture and track down issues when designs become more complex.

Low Latency, High Determinism

Two of the biggest issues we run into when scripting in Flow Graph are latency and lack of determinism. Because the flow is determined on a per-node input/output basis and programmers can implement new nodes in pretty much any way they like, it is difficult (sometimes impossible) to predict the order in which nodes will be called and perhaps more importantly, when a node will be called. This problem becomes particularly troublesome when evaluating transitions for a state machine because if you don't get an immediate response you have to defer evaluation until one or two frames later, by which time previously evaluated conditions may no longer be valid.

Schematyc solves these issues by creating a clear distinction between nodes that can trigger scripted logic and nodes that are part of scripted logic. The flow of execution must be defined explicitly and any nodes not part of the execution path will be grayed out to indicate that they will never be called.

Schematyc Assets Explained

With Schematyc 5.4 we introduce two types of Schematyc assets.

Schematyc Entity

- Can be placed in a level
- Can be used as a base for other Schematyc Entities
  - Everything in the base that was marked as public can be accessed
- Can contain following content
  - Components
  - Graphs
    - Construction Graph
    - Signal Graph
    - Function Graph
  - Variables
    - Variables
    - Timer
  - Signals
  - State Machine
    - States

Schematyc Library

- Can not be placed in a level
- A collection of functions, enumerations, and signals
- Can be accessed by Schematyc Entities and Schematyc Libraries
- Can contain following content
  - Function Graphs
  - Enumerations
  - Signals

How to create a Schematyc Asset

There are two ways to create a Schematyc Asset. You can go through the Asset Browser or do it in Schematyc.

Asset Browser

1. Open the **Asset Browser**
2. Choose a directory
3. Right click in the Search Results pane and select **New**
4. Choose either a Schematyc Entity or Library depending on what you want to create

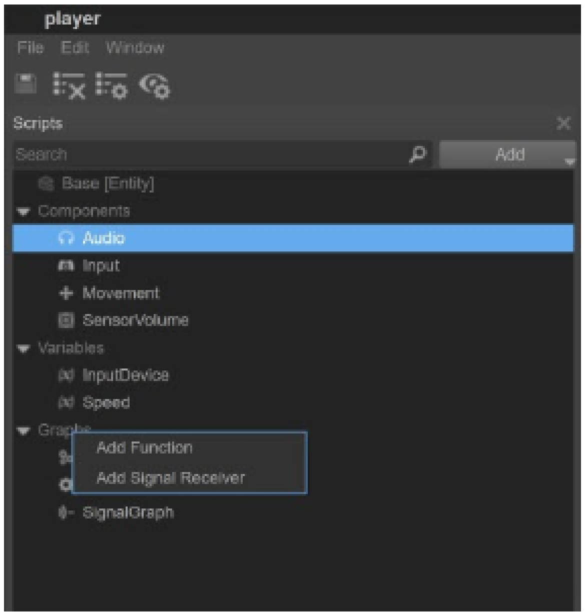
Schematyc Editor

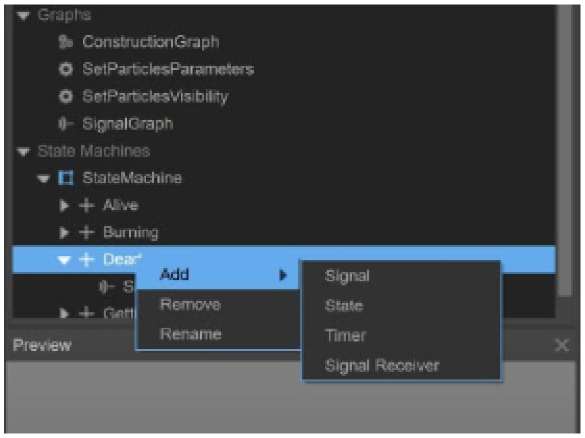
1. Go to **File** and select **New**
2. Choose either a Schematyc Entity or Library depending on what you want to create

How to Add Content to an Asset

Content can be added through the **Add** menu right to **Search bar**. It can be seen as the context menu of the whole object and therefore adds new items on the top level of the object.

Child content can be added through the item context menu:





Intro Video

For a series of videos about Schematic, follow **this link**.