# About

The MathAct is Scala toolset for modeling, simulating and analyzing of dynamic systems. It similar to [Mathlab Simulink](https://en.wikipedia.org/wiki/Simulink) but uses the Scala instead of Mathlab language and based on the messages propagation instead of state iteration. You can use it as additional toolset in your research/testing/playing projects.

Currently it contain small number of tool but I will add new ones, as soon as they will created.

# Getting Started

You need to have installed [JDK 1.8+](http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html) and [SBT 0.13+](http://www.scala-sbt.org/download.html), also some Scala IDE will be helpful.

Download or clone this repository somewhere on your local machine, for example with Git: **git clone https://github.com/AlexCAB/MathAct.git**

Navigate to MathAct folder: **cd MathAct**

Now you can run examples with a command: **sbt mathact\_examples/run examples.Simple**

In addition, you can import SBT project to your favorite IDE and run **examples.Simple** object from there.

# Demo

Running of couple of sketches:

<video>

# Project Structure

Project comprises of next subprojects:

* Core – contains common and service definitions.
* Tools – set of tools, that may be used to compose sketch.
* Examples – contains a set of examples that demonstrate a using of the MathAct toolset.

# Defining of Sketch

…типичный процесс разработки какого-либо решения (т.е. системы решающей некоторою проблему) включает этап моделирования, это когда у вас уже есть идея о том, как построить решение, но вы недостаточно уверены в этой идее чтобы начать реализацию и хотели бы оценить вашу идею «виртуально». Построить математическую модель вашей идей и выполнить её симуляцию хороший вариант для этого.

...для того чтобы определить и симулировать какую то модель MathAct вы можите создать скетч. По сути скетч это Скала класс расширеных от SimpleWorkbench, внутири корого будут размещены все определния относящиеся к вашей модели, это могут быть любые Скала определния.

<code> //Пример пустого скетча

…весьма вероятно что в процессе разаботки у вас скопится несколько скетчей потому будет удобно обьединить все скетчи относящиеся например к одному проекту в один список. Для чего вы можите определить Скала обьект, расширеный от класа Sketches. Это будет стартовый обьект приложения. После старта приложения все зарегистрированые скетчи будут отображатся в списке из котрого вы сможете запускать их.

<code> //Пример с регистрацией скетча и описание параметров

...Теперь вы имеет своё первое простое приложение и можете его запустить. После запуска вы увидите список скетчей:

<screenshot>

…И нажав не Run (зелёный триугольник) вы увидите Sketch UI (смотрите ниже описание элементов управления):

<screenshot>

...Собственно больше ничего интересного не произойдёт так как скетч пустойб так что давайте добавим что нибуть туда.

# Sketch (model) Structure

Similarly, to the Mathlab Simulink, you can compose your model from the blocks.

…В отличии от Simulink вы определяете модель в текством виде на языке Scala, лично я считаю работу с текстом более удобной, но тем не менее планирую написать графический редактор так-же.

...Также в отличии от Simulink блоки обмениваются сообщениями и в целом подход к построению модели похож на то что используется в [akka-streams](http://doc.akka.io/docs/akka/2.4/scala/stream/stream-quickstart.html). Т.е. ваша модель представляет из себя граф где узлы (блоки) это процессоры сообщений. И рёбра — это соединения между процессорами (пути) по которым распространяются сообщения.

...Каждый блок может иметь несколько входов (инлетов) и выходов (оутлетов), которые могут быть соединены.

...Образно вы можыте представить себе блоки как элестронные микросхемы, as [pneumatic automation](https://www.youtube.com/watch?v=IqIqpTwKMPI) или как елементы гидравлической системы (I prefer last one, which couse to be reflected in the paceges and class names). Между елементами (блокми) их циркулирует поток сообщений.

<рисунок электронной схемы, каой ни буть осмысленной импульсной схемы>

...Кажды инлет блока имеет свою очередь для буферизации сообщений откуда сообщения передаются в пользоватльский код на обработку, далее пользовательский код может отправлять сообщения на оутлеты блока, которые будут переданы на инлеты других блоков:

<рисунок кастрюли из заметок, но с тремя блоками (верхний и нижний частями, средний полностью) и средний блок должен иметь несколько входов и выходов>

...Типичный блок выглядит примерно следующим образом:

<код простого блока из ресунка выше внутри класа скетча из пред главы, с только входами и выходами>

...Ниже описано как это в целом устроено и как определить свой произвольный блок.

...Для соединения блоков используется DSL похожый на [akka-streams DSL](http://doc.akka.io/docs/akka/2.4.14/scala/stream/stream-composition.html), например следующий скетч

<скетч из нескольких блоков и соединения, взять из статьи о стримах>

...Представляет следующий граф:

<рисунок графа, взять из статьи о стримах>

...Если блок имеет один вход или олин выход или вход и выход вы можыте сипользовать short linking DSL to make the connections definition shorter:

<скетч демонстрирующий использоватние коротного связания, покадывать блоки квадратиками с входами/выходами как в статье про стримы>

...Каждый оутлет может быть подключен к нескольким инлетам, в этом случае он будет broadcast сообщения на все инлеты. Так же к каждому инлету может быть подключено несклько оутлетов, в этом случае сообщения от всех оутлетов будут попадать в одну очередь инлета

<скетч демонстрирующий подключение нескольких инлетов/оутлетов>

...Представляет следующий граф:

<граф подключение нескольких инлетов/оутлетов >

...Я рекомендую просмотереть примеры скетчей <ссылка на гитхаб примеры>

# Simulation Approach

… о дискретной и аналогвой симуляции

...при таком подходе все вычиления происходят мгновенно и одновременно (с точки зрения виртуального времени модели)

# Build and Run Simple Model

...пошаговое построени и запуск простой модели (пид регулятора)

<video>

# User Interface

# Common Architecture

# Compose Your Own Block

...о вайренге здесь

# Adding Block UI

# Future Work