



# BioGears Tutorial

Presentation for MMVR

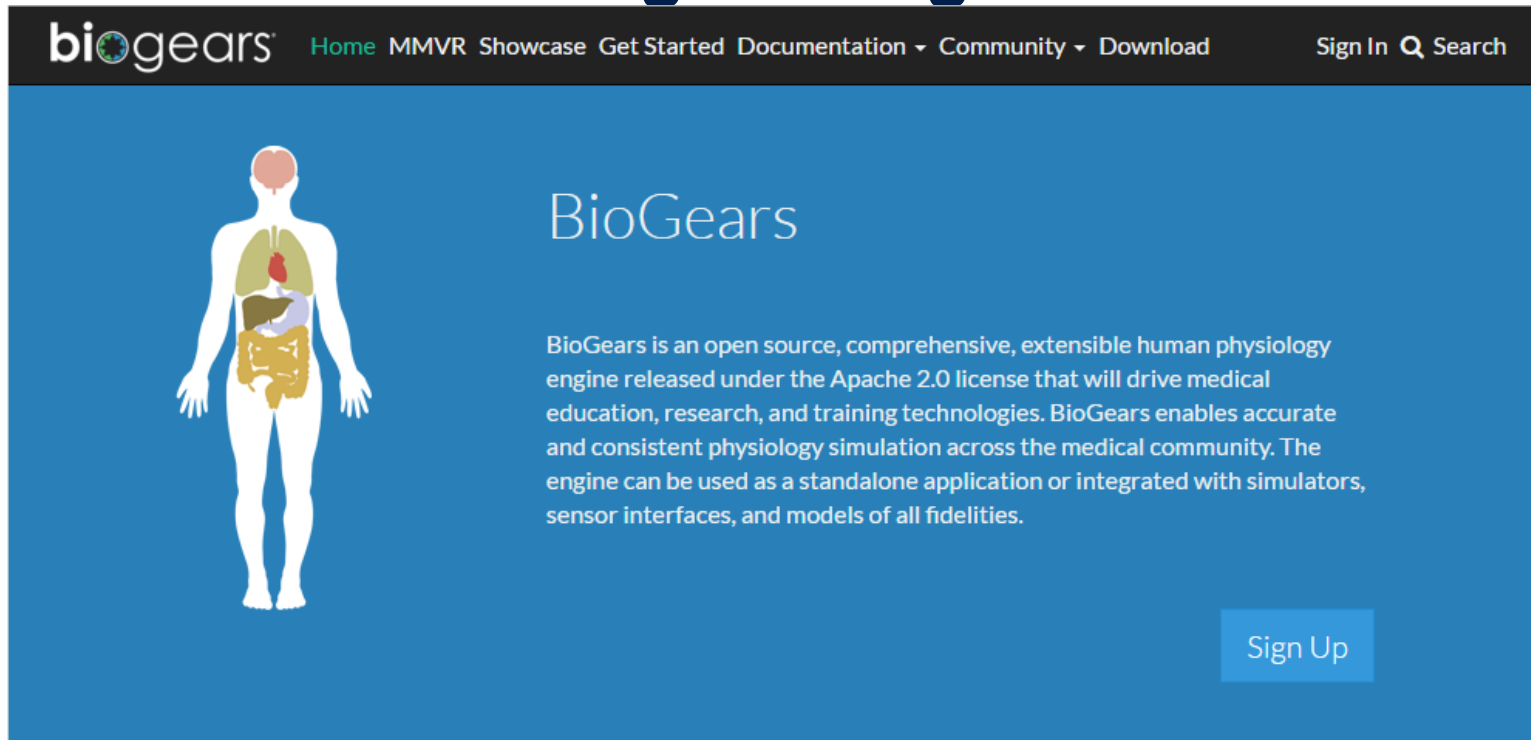
Presenters: Aaron Bray, Rachel Clipp, PhD, Jeff Webb  
Applied Research Associates, Inc. (ARA)  
7 April 2016

# Table of Contents

- Website Overview
- GUI Walkthrough
  - Running a scenario, viewing outputs, generating plots
  - Modifying a scenario, patient, drug, and environment
- Programming with the SDK
  - Examine the PhysiologyEngine Interface and how it controls the BioGears Engine via the Common Data Model
    - Apply the concepts to with an Airway Obstruction example

# WEBSITE OVERVIEW

# www.biogearsengine.com



- Showcase – Waveform data player from applying BioGears to several narratives
- Getting Started – General overview of our GUI
- Documentation – Details on how we implemented our software
- Community – Forums for any questions you may have
- Download – How to get our software products

# Showcase Scenarios for Combined Effects

[Link to Player](#)

- **Multi-Trauma, 22 y/o Male**
  - **Body Condition:** Soldier
  - **Insults:** Trauma leading to massive hemorrhage, tension pneumothorax
  - **Assessments:** Bleeding rate, heart rate, blood pressure, respiration rate, O2 saturation
  - **Interventions:** Tourniquet, Needle Decompression, narcotics, fluid resuscitation
  
- **Heat Stroke, 25 y/o Male**
  - **Body Condition:** Physically fit
  - **Insults:** Strenuous work at high altitude resulting in heat stroke
  - **Assessments:** Core temperature, sweat rate, heart rate, CBC
  - **Interventions:** Active cooling, I.V. fluids
  
- **Asthma Attack, 40 y/o Female**
  - **Body Condition:** Suffers from asthma, no other known issues
  - **Insults:** Asthma attack
  - **Assessments:** Respiration rate, EtCO2, Heart rate, BP, SPO2, PFT
  - **Interventions:** Administer beta agonist
  
- **Environment Exposure, 17 y/o Female**
  - **Body Condition:** Hypothermia, variable insulation of clothing
  - **Insults:** Cold air or water exposure
  - **Assessments:** Core and peripheral temperature, Heart Rate, Respiration Rate
  - **Interventions:** Removal from environment, active heating, increase clothing

# Getting Started

- A walk through of using our GUI to execute one of our Showcase Scenarios
  - View scenario
  - Select data to plot
  - Generated results
  - Modify a scenario

## Getting Started with The BioGears Toolkit

Let us help you get started using BioGears! In the below sections, we provide an overview of how to use the toolkit – you'll be up and running in no time!

To learn more about getting started with BioGears, visit the documentation or take a look at what people are talking about on the BioGears Forums.

Get Started

Download

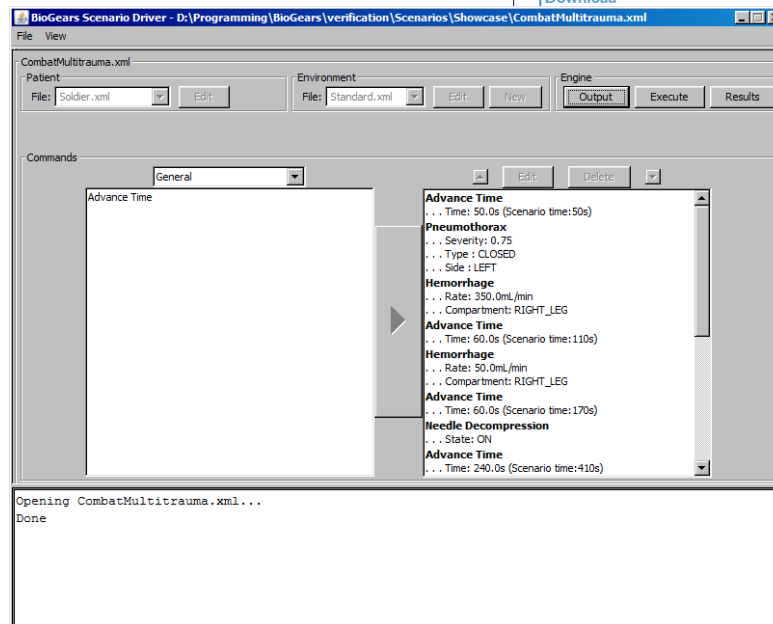
Download

For more information about how to download the toolkit, please see the [downloads](#) page.

## Toolkit

The Toolkit contains the following:

- A collection of example [Scenario XML Files](#) that can be found in the `/bin/scenarios` folder.
- A GUI provides a simple interface for creating, editing, and executing a scenario file, as well as creating a result and plots of requested data



# Documentation



- The website includes detailed documentation for physiology systems and software components (e.g., CDM, Toolkit, SDK)
- Text and tables that explain: system background, model implementation and limitations, equations used, data sources and validation matrices
- [www.biogearsengine.com/documentation/index.html](http://www.biogearsengine.com/documentation/index.html)

**Download BioGears**

We are releasing the BioGears Physiology Engine and Common Data Model under the Apache 2.0 license.

You can use BioGears as a stand-alone application or integrate BioGears into your training and immersive learning technologies. We provide comprehensive documentation, including an SDK that provides detailed tutorials and step-by-step information to help our end-user groups: simulation content developers, biomedical researchers, and technology integrators, and easier user education.

Over the next year of the project we will continue to add physiology systems and functionality to BioGears. We look forward to hearing from you and encourage feedback from our users!

Choose your platform:

Windows Mac OS Linux Raspberry Pi

**Download**

**Official Release**

Our latest deployment is intended to be an intermediate release to showcase the capabilities of the BioGears Physiology Engine and Common Data Model interface. The current version of the software includes examples of all types of engine interfaces, but does not include all of the functionality or physiologic systems that will be present in the final product. This version of the software is meant to elicit feedback and enhance community involvement in establishing end-product expectations.

**Release Notes (v5.0.0)**

The latest deployment includes the following notable updates:

- General Bug fixes
- New Heat Stroke showcase scenario
- Physiology interface changes
  - Created a Java interface for controlling the BioGears C++ engine
  - Added examples and the BioGears.jar to the BioGears SDK
  - Removed methods for executing a scenario from the Physiology Engine interface
  - Use a `IScenarioRunner` class for executing a scenario; see `HowTo-RunScenario.cpp` in the SDK
  - Urinalysis
- Baroreception
  - Nervous System responds to changes in mean arterial blood pressure by modifying the cardiovascular model

**Community**

**Download**

**General Discussion**

**Ask a Question**

**Categories**

**Recent Discussions**

**Activity**

**Best Of...**

**Unanswered**

**Category Filter**

Viewing: all categories | followed categories

**Project?**

**Overview**

**Abstract**

The BioGears Circuit Solver was created to extract the lumped parameter physics calculations from within the individual systems and centralize the generic calculations. Closed loop circuits can be defined easily within the Common Data Model (CDM). They are fully dynamic, and can be manipulated and modified at each time step. The solver can be used to analyze electrical, fluid, and thermal circuits using native units. The solver outputs were validated using the outputs from an existing known third party circuit solver. All results matched the validation data, confirming this is a sound approach for the BioGears Engine.

**Introduction**

BioGears systems use lumped parameter circuits to mimic the physiology of the human body. These circuits use fluid or thermal elements that are analogous to electrical circuit elements. The circuits have several types of feedback mechanisms that can be set and changed at every time step. Figure 1 presents a generic example of very low fidelity lumped parameter physiology circuits. Circuits can be thought of as pipe networks for fluid analysis.

**Figure 1: An example of physiology lumped parameter modeling.** This example shows very low fidelity models of specific cardiovascular compartments (left), and a respiratory combined mechanical ventilation and free breathing model (right) [1].

**Figure 2: Nodes and paths are the lowest level elements used to define all results in BioGears.** Paths correspond to fluid conductors (i.e., vessels). Nodes are placed at the intersections of paths. In fluid systems, paths can be thought of as resistances pipes and nodes as pipe junctions.

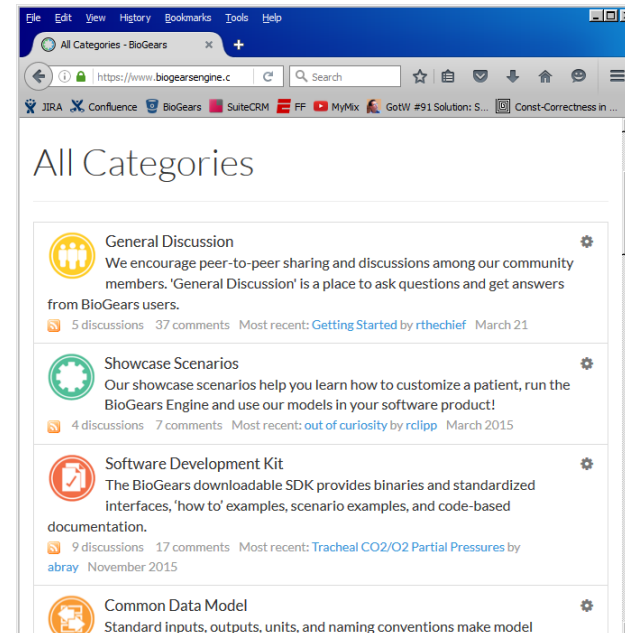
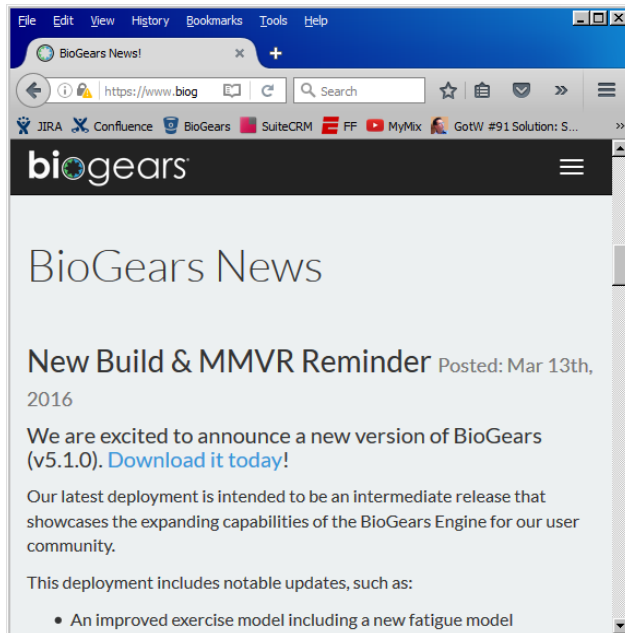
**Validation Data**

**Percent Error**

Parameter	Mean	Stdev	Min	Max	Percent Error
Heart rate (b/min)	75	10	60	90	0.0%
Mean arterial pressure (mmHg)	93	10	80	105	0.0%
Cardiac output (L/min)	5.0	0.5	4.0	6.0	0.0%
Stroke volume (mL)	70	10	60	80	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean left ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean right ventricular pressure (mmHg)	120	10	110	130	0.0%
Mean pulmonary artery pressure (mmHg)	16	2	14	18	0.0%
Mean aortic pressure (mmHg)	93	10	80	105	0.0%
Mean ventricular pressure (mmHg)	120	10	110		

# Community

- News
  - Check to see what our team is doing, where we have been and where we are going next
- Work With Us
  - Purchase support service in daily or weekly increments
  - Find more materials and information about how to work with us
- Forums
  - Have problems or questions? Ask us questions!





# Downloads

- Toolkit – Our GUI plus various command line tools and plot generation scripts
  - Available for Windows, Mac, Ubuntu Linux
- SDK – Software Development Kit – How-To examples, headers and compiled libraries
  - Available for Windows, Mac, Ubuntu Linux, Raspberry Pi
- Source Code – All of our code
- Documentation – Offline viewable version of our documentation
- Verification – Verification suite of expected results for all 150+ test cases
- [www.biogearsengine.com/download](http://www.biogearsengine.com/download)

## Download BioGears

We are releasing the BioGears Physiology Engine and Common Data Model under the [Apache 2.0](#) license.

You can use BioGears as a stand-alone application or integrate BioGears into your training and immersive learning technologies. We provide comprehensive documentation, including an SDK that provides detailed tutorials and step-by-step information to help our end user groups: simulation content developers, biomedical modelers, MedSim technology integrators, and researchers/ educators.

Over the next year of the project we will continue to add physiology systems and functionality to BioGears. We look forward to hearing from you and encourage feedback from our users!

Choose your download

*\*Please note that the below downloads are compatible only with Windows.*

Toolkit

SDK

Source Code

Documentation

Verification

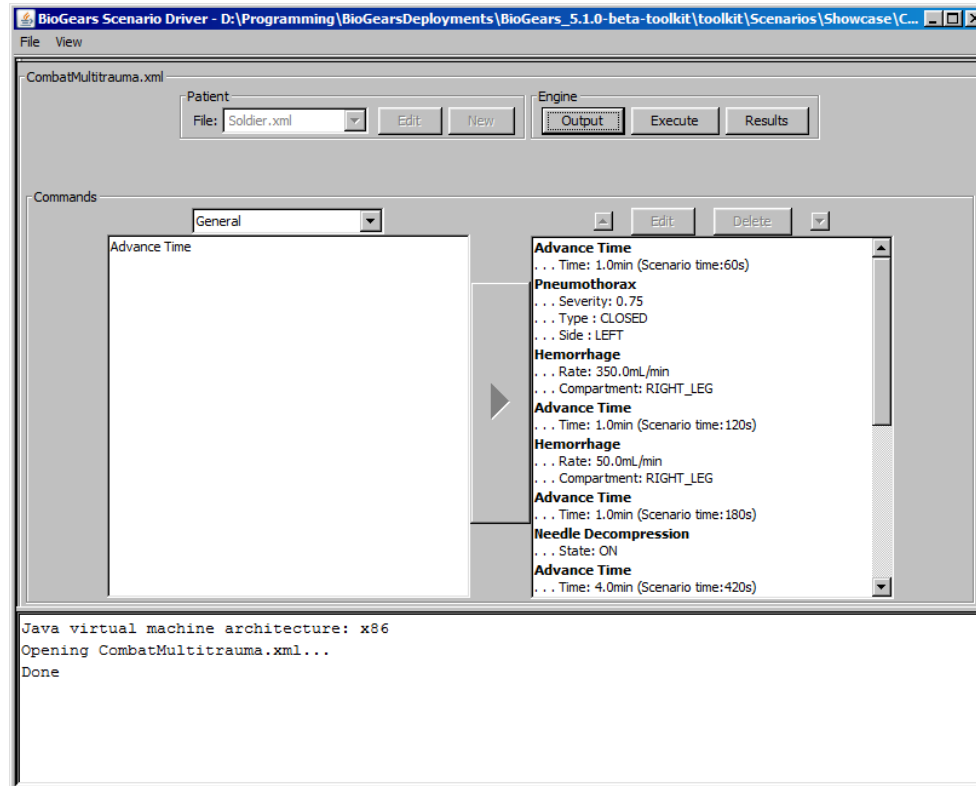
# GUI WALKTHROUGH

## BioGears GUI

- Java based user interface that can execute and plot data from a scenario xml file
- Java GUI
  - Requires Java 1.8
  - 32-bit and 64-bit available for Windows
  - GUI Run-time Problems?
    - If it fails to open, maybe a JVM mismatch
    - To ensure you properly installed the Java Runtime Environment, in your command window type: `java -version`
- The GUI is still a work in progress and is very limited in functionality and in error handling. You may run into various bugs
- Tutorial Actions for you to follow along using the GUI during this presentation are in red

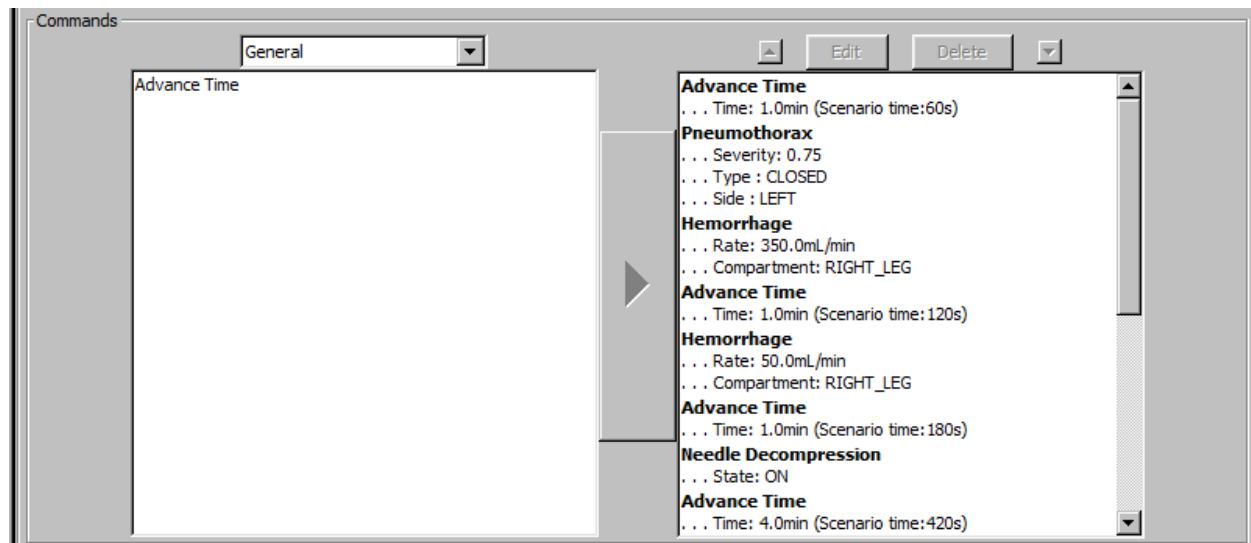
# Opening a Scenario

- Start the GUI:
  - Windows users run the BioGearsGUI.bat
  - Mac users
    - Using the DMG file, mount the DMG, drag the application to the applications folder and from there you can double click it to run the GUI
    - Using the Toolkit download, run the BioGearsGUI.sh
  - Ubuntu Linux users run the BioGearsGUI.sh
- Click File->Open and Select Scenarios/Showcase/CombatMultitrauma.xml



# Scenario Commands

- Once the scenario loads, the commands list box on the right is populated with all conditions and actions in the scenario
- The left list box provides a filterable list of the available commands you may add to the scenario (Currently disabled)
  - General (Advance Time)
  - Patient Actions, Conditions
  - Environment Actions
  - Anesthesia Machine Actions



## Patient and Log

- The patient file in the scenario is displayed in this dropdown
  - Currently editing a scenario file is not supported and is disabled

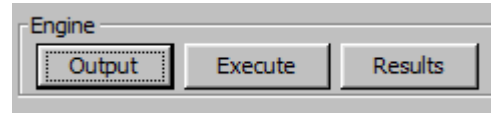


- The log window is used to communicate information, warnings, and errors that the GUI encounters



- You should see what the JVM bit architecture is being used to run the GUI. This could come in handy for troubleshooting

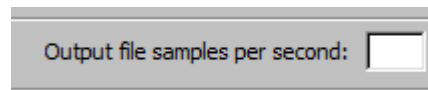
# Scenario Execution Controls



- The Engine button group provide the following functionality:
  - Output – Request data to pull and plot from the engine as the scenario executes
    - All provided scenarios have a predefined set of data requested, you do not need to open the Output dialog, you may just open a provided scenario and click execute and the GUI will plot the predefined data
  - Execute – Start a BioGears engine and execute the scenario
  - Results – View the plot images previously generated
- Click the Output button

# Output Data Request Dialog

- The Output button brings up a dialog where you can select which data you would like to pull from the engine as it runs
  - Tabs separate the various data type available
    - Physiology – Tabs for each physiology system and its available data
    - Anatomy – Compartment data
    - Environment – General Environmental/Atmospheric data
    - Equipment – Tabs for ECG, Anesthesia machine, and Inhaler compartment data
    - Substances – Whole body or compartment specific substance data
    - Patient – Patient data unique to the patient configuration itself
- If the scenario file already contains data requests, those properties are checked in the dialog
- In the bottom right of the dialog, you can specify a sampling rate by setting a value for 'Output file samples per second'
  - Default is set by the BioGears time step: 90 samples per second
  - This will create smaller output files, but the data is not interpolated in any way



Output file samples per second:



# Physiology Output Tab

**Output - CombatMultitraumaScenario**

Physiology | Anatomy | Environment | Equipment | Substances | Patient

Endocrine System | Blood Chemistry System | Drug System | Respiratory System | Renal System  
 Cardiovascular System | Energy System | Gastrointestinal System

<input checked="" type="checkbox"/> Arterial Pressure	Units: mmHg
<input checked="" type="checkbox"/> Blood Volume	Units: mL
<input checked="" type="checkbox"/> Cardiac Output	Units: L/s
<input checked="" type="checkbox"/> Central Venous Pressure	Units: mmHg
<input checked="" type="checkbox"/> Diastolic Arterial Pressure	Units: mmHg
<input type="checkbox"/> Heart Ejection Fraction	Unitless
<input checked="" type="checkbox"/> Heart Rate	Units: 1/min
<input checked="" type="checkbox"/> Heart Stroke Volume	Units: mL
<input type="checkbox"/> Mean Arterial Carbon Dioxide Partial Pressure	Units: mmHg
<input type="checkbox"/> Mean Arterial Carbon Dioxide Partial Pressure Delta	Units: mmHg
<input checked="" type="checkbox"/> Mean Arterial Pressure	Units: mmHg
<input type="checkbox"/> Mean Central Venous Pressure	Units: mmHg
<input type="checkbox"/> Mean Skin Flow	Units: L/s
<input type="checkbox"/> Pulmonary Arterial Pressure	Units: mmHg

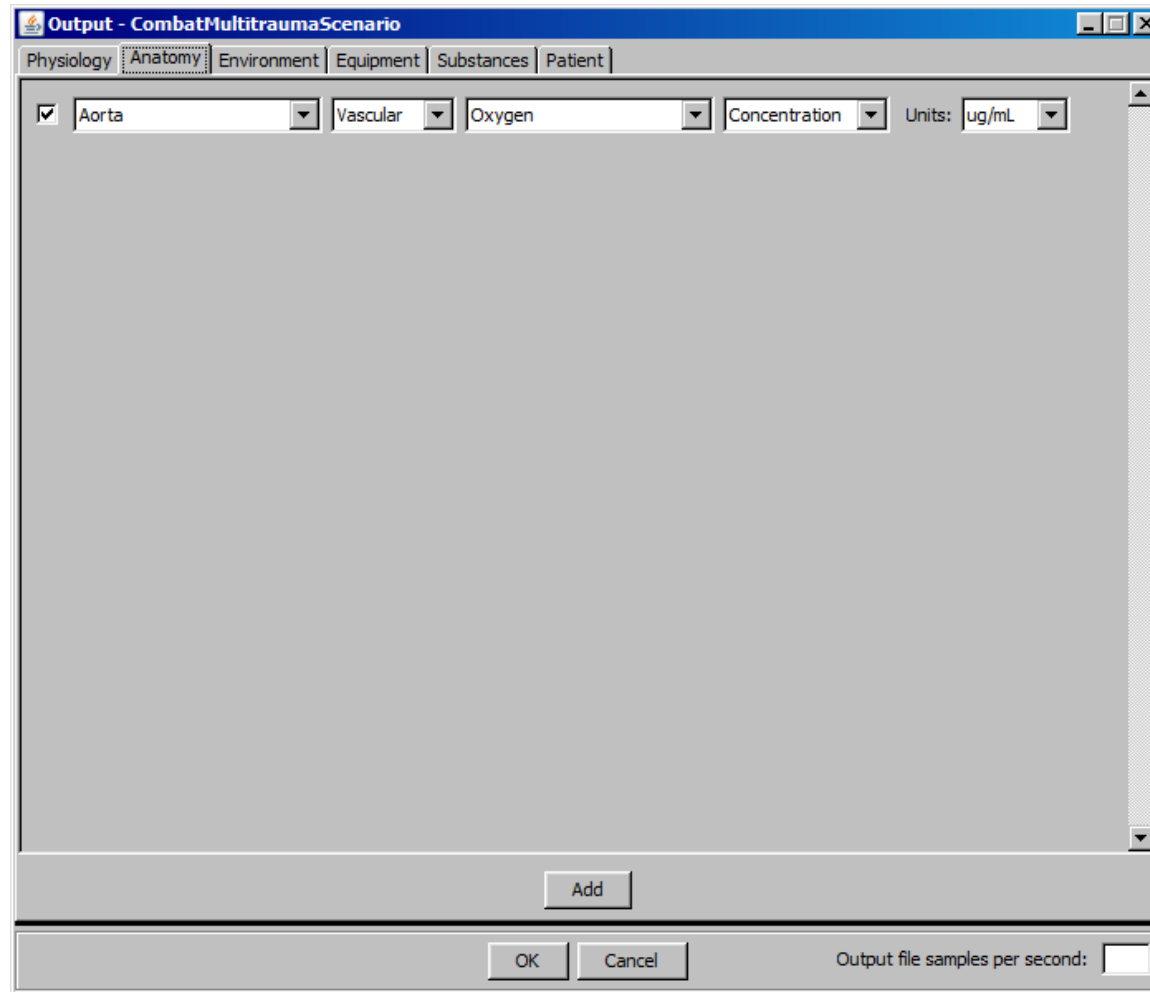
OK Cancel

Output file samples per second:

# Anatomy Output Tab

- Click the Anatomy Tab
- Click the Add button to create a new row for tracking anatomy data
  - To remove the row, uncheck its check box; when you exit the dialog via the OK button the row will be removed from the scenario file and not be shown the next time you open the Output dialog.
- First choose an anatomy compartment that you are interested in
  - Choose the Aorta
- The second column of drop-down boxes is for the type of fluid you are interested in : Vascular, Pulmonary, Tissue, or Extravascular
  - Choose Vascular
- Next choose a substance or 'Compartment'
  - The last drop down list changes based on selecting a substance or 'Compartment'
    - Compartment data is the fluid flow, volume, and pressure
  - If you want to choose a substance in the fluid note commonly used substances appear at the top of the drop-down. After the dotted line, substances appear in alphabetical order
  - Choose Oxygen
- The next drop down list will be substance or compartment properties
  - Available Substance data depends on the fluid type you chose
    - Vascular is Mass && Concentration
    - Pulmonary is Volume && Volume Fraction
    - Molarity, Partial Pressure, Saturation
      - Saturation is only available for Oxygen and CarbonDioxide
  - Choose Concentration

# Modified Anatomy Output Tab



\* Note: Not all combinations are supported

# Environment Output Tab

Output - CombatMultitraumaScenario

Physiology | Anatomy | **Environment** | Equipment | Substances | Patient

Environment

<input type="checkbox"/> Air Density	Units: <input type="text" value="ug/mL"/>
<input type="checkbox"/> Air Velocity	Units: <input type="text" value="m/s"/>
<input type="checkbox"/> Ambient Temperature	Units: <input type="text" value="degC"/>
<input type="checkbox"/> Applied Surface Area	Units: <input type="text" value="cm^2"/>
<input type="checkbox"/> Applied Surface Area Fraction	Unitless
<input type="checkbox"/> Applied Temperature	Units: <input type="text" value="degC"/>
<input type="checkbox"/> Atmospheric Pressure	Units: <input type="text" value="mmHg"/>
<input type="checkbox"/> Clothing Resistance	Units: <input type="text" value="rsi"/>
<input type="checkbox"/> Convective Heat Loss	Units: <input type="text" value="W"/>
<input type="checkbox"/> Convective Heat Transfer Coefficient	Units: <input type="text" value="W/m^2 K"/>
<input type="checkbox"/> Cooled Surface Area	Units: <input type="text" value="cm^2"/>
<input type="checkbox"/> Cooled Surface Area Fraction	Unitless
<input type="checkbox"/> Cooling Power	Units: <input type="text" value="W"/>
<input type="checkbox"/> Emissivity	Unitless
<input type="checkbox"/> Evaporative Heat Loss	Units: <input type="text" value="W"/>

OK Cancel

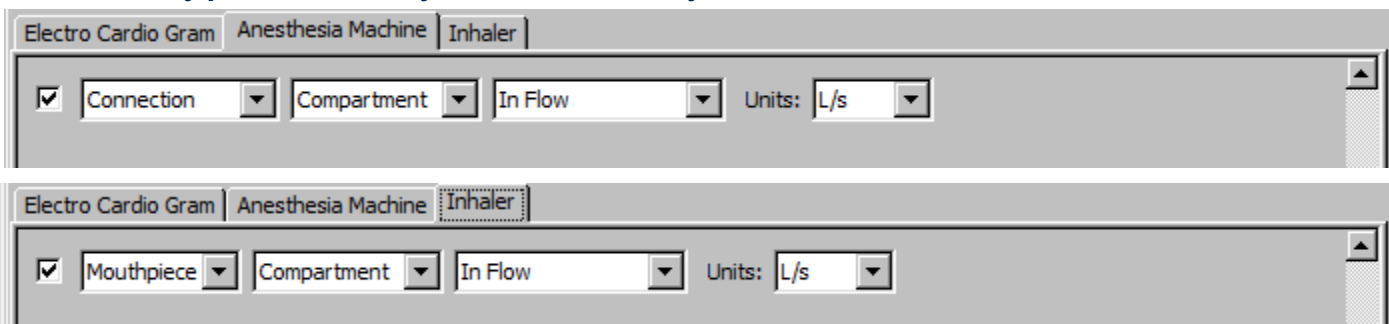
Output file samples per second:

## Equipment Output Tabs

- BioGears supports 3 types of Equipment
  - ECG Machine, Anesthesia Machine, and an Inhaler
- ECG only provides the ECG Waveform from the Third Lead

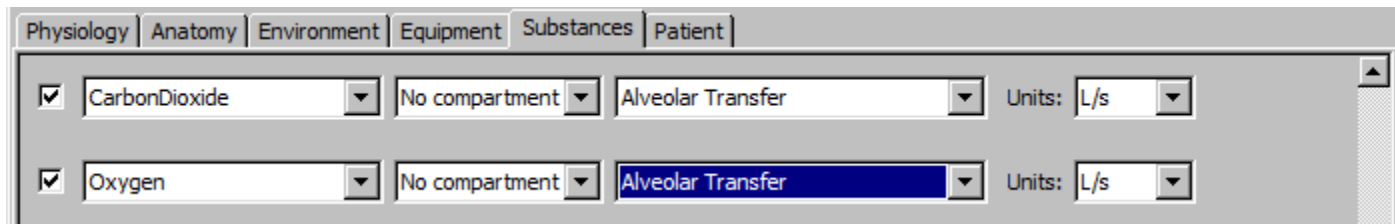


- The Anesthesia Machine and Inhaler provide compartment data and request are added just like the Anatomy tab
  - Choose Component, 'Compartment' or Substance, Property, Unit
    - Fluid type is always Pulmonary



## Substance Output Tab

- The Substance tab works much like the Anatomy tab but in a different order
  - Choose Substance, 'No Compartment' OR a specific Compartment, Property, Unit
    - If you select 'No Compartment' the data is for the whole body
    - The 3<sup>rd</sup> drop down (property) will change based on the 2<sup>nd</sup> drop down
- Click the 'Add' Button
- Choose Carbon Dioxide
- Choose 'No Compartment'
- Choose 'Alveolar Transfer'



The screenshot shows the 'Substances' tab selected in a software interface. It displays two rows of substance data with the following settings:

Substance	Compartment	Property	Units
CarbonDioxide	No compartment	Alveolar Transfer	L/s
Oxygen	No compartment	Alveolar Transfer	L/s

# Patient Output Tab

**Output - CombatMultitraumaScenario**

Physiology | Anatomy | Environment | Equipment | Substances | **Patient**

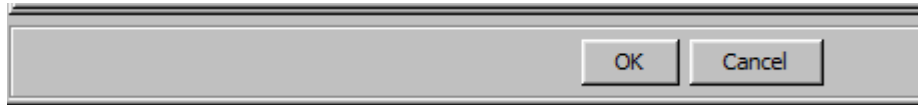
Patient

<input type="checkbox"/> Age	Units: s
<input type="checkbox"/> Basal Metabolic Rate	Units: W
<input type="checkbox"/> Body Fat Fraction	Unitless
<input type="checkbox"/> Carina To Teeth Distance	Units: m
<input type="checkbox"/> Central Controller CO2 Pressure Set Point	Units: mmHg
<input type="checkbox"/> Diastolic Arterial Pressure Baseline	Units: mmHg
<input type="checkbox"/> Expiratory Reserve Volume	Units: L
<input type="checkbox"/> Functional Residual Capacity	Units: L
<input type="checkbox"/> Heart Rate Baseline	Units: Hz
<input type="checkbox"/> Heart Rate Maximum	Units: Hz
<input type="checkbox"/> Heart Rate Minimum	Units: Hz
<input type="checkbox"/> Height	Units: m
<input type="checkbox"/> Inspiratory Capacity	Units: L
<input type="checkbox"/> Inspiratory Reserve Volume	Units: L
<input type="checkbox"/> Left Heart Elastance Maximum	Units: cmH2O

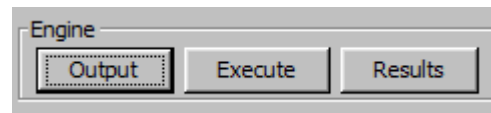
OK Cancel

Output file samples per second:

## Complete Data Requests



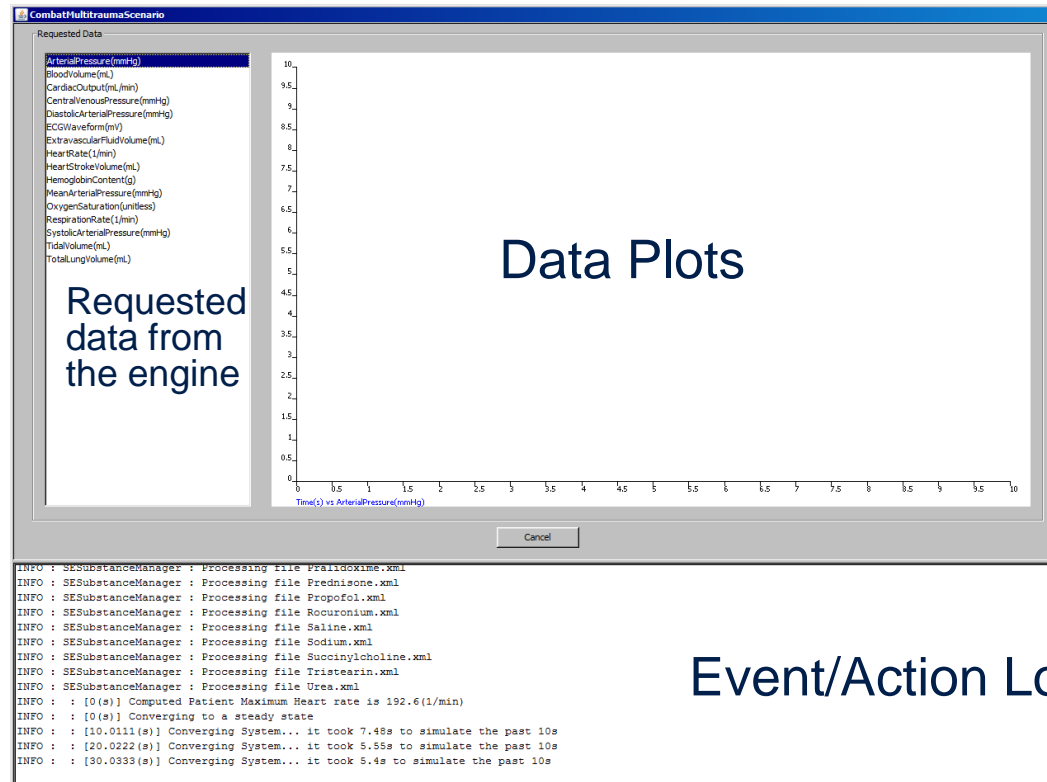
- When you are finished choosing properties, click the OK button to rewrite the scenario XML file to disk
  - Note: To exit without saving, choose Cancel or close the window via the X on the right title bar
- Click OK
- Click the Execute button





# Scenario Execution

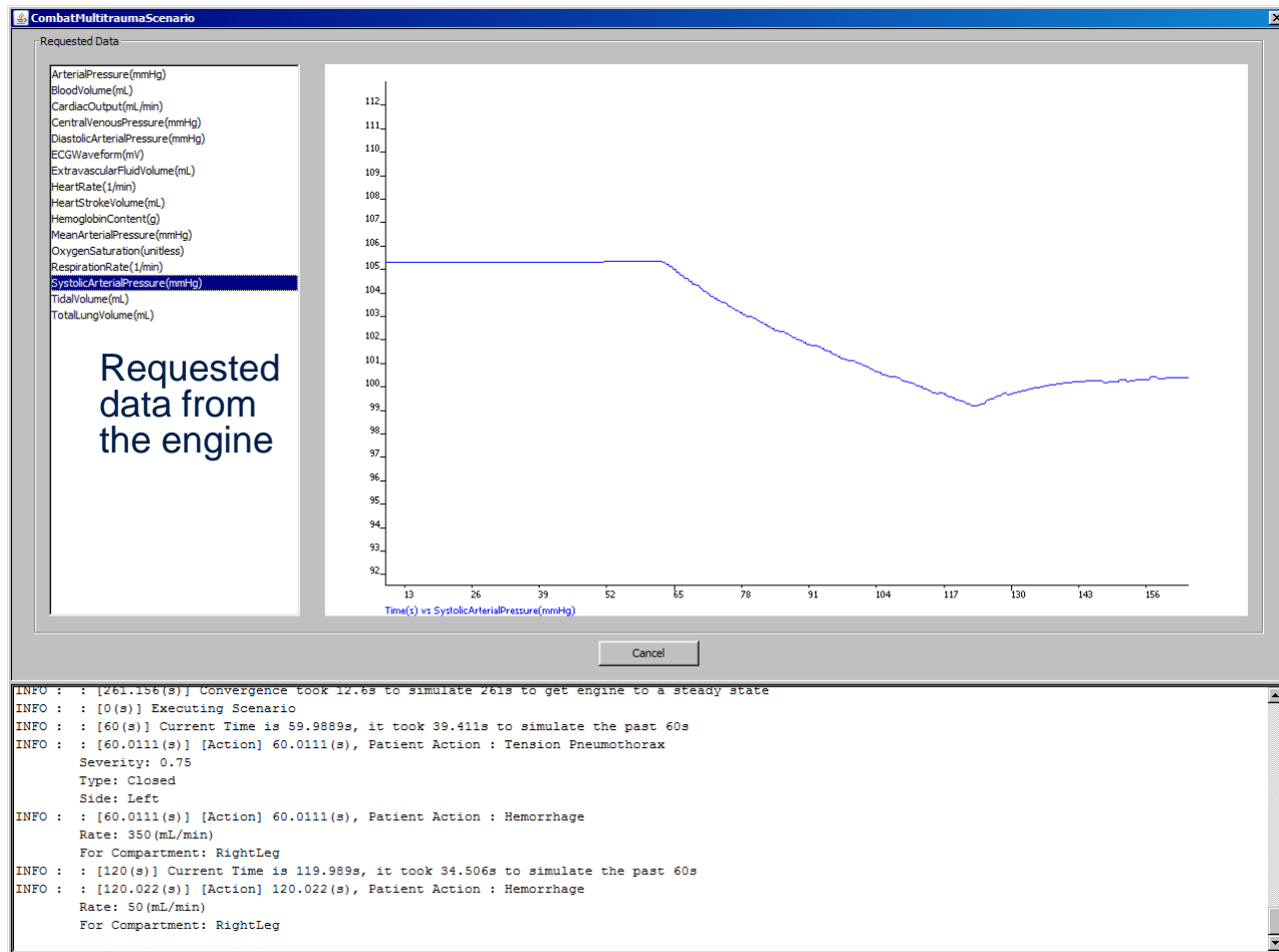
- A new child window will open for real-time plotting of data
- This plot window also has a log specific to execution
  - Note the messages for Convergence, **Wait for convergence to finish**



Event/Action Logging

# Scenario Execution

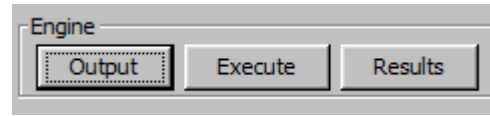
- You may click any requested data to see data plots



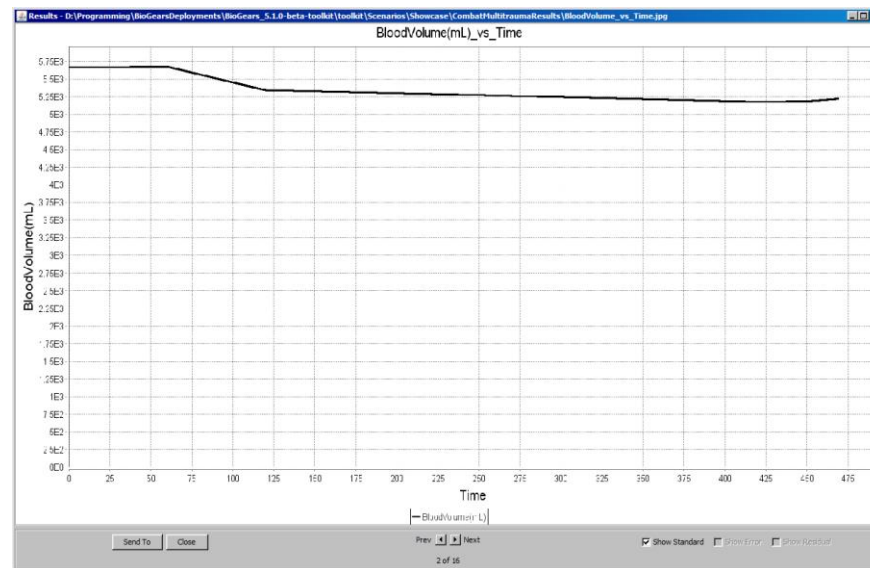
## Scenario Execution

- The execution log file as well as a comma delimited results file will be created in a directory under the location of the scenario file
  - In this case: toolkit\Scenarios\Showcase\CombatMultitraumaResults
- At any point during the execution of the scenario, you can cancel the run by clicking the cancel button
  - If the scenario is still running when you click the button, a pop up will ask you if you would like to plot the data you have run
    - If you choose yes, plot images will be created and place in the same folder as the results file
- If you let the scenario execute to completion, the GUI will automatically create plot images in the results folder
  - You can still view all data plots after the engine has finished
  - Close the execution window

# View Scenario Results



- Click the Results Button
- A simple image viewer comes up that allows you to scroll through the plot images created.
  - **Note:** The graphs are image files and do not allow for much detail or zoom capability. We recommend using a graphing application (e.g., DPLLOT) that can read comma-delimited text file.
  - Use the Send To button to open the graph image with a different viewer.



# Scenario Files

- Scenario files are written in XML and the toolkit comes with a collection in the toolkit/bin/scenarios folder
- The following is an overview of the scenario tag structure
  - <Name> - A name for the scenario
  - <Description> - A description of this scenario
  - <PatientFile> - The patient file is to be used
  - <DataRequest> - What values to put in the results file
  - <Condition> - Any conditions you want to initialize the patient with
  - <Action> - Time advancement, Insults and Interventions to execute
- The XML formatting for all Conditions and Actions is documented here: [www.biogearsengine.com/documentation/scenario\\_xml\\_file.html](http://www.biogearsengine.com/documentation/scenario_xml_file.html)
  - All actions and conditions are used at least once in a scenario in the provided toolkit/bin/scenarios folder

# Modifying a Scenario File

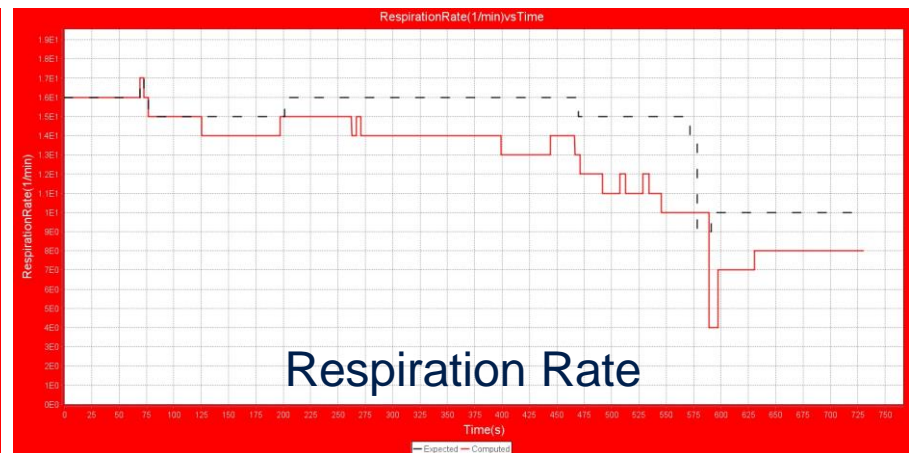
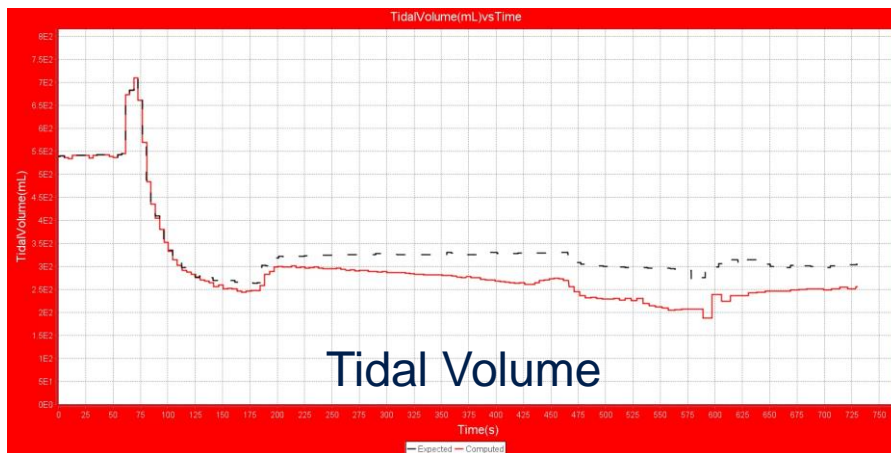
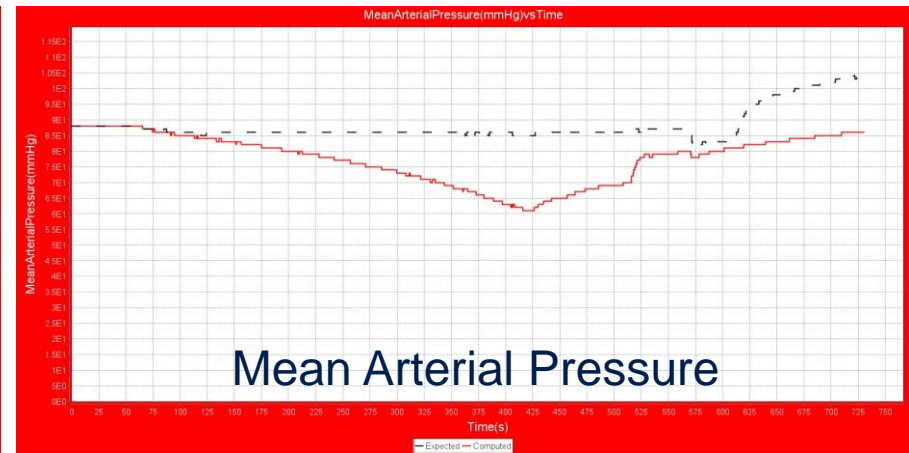
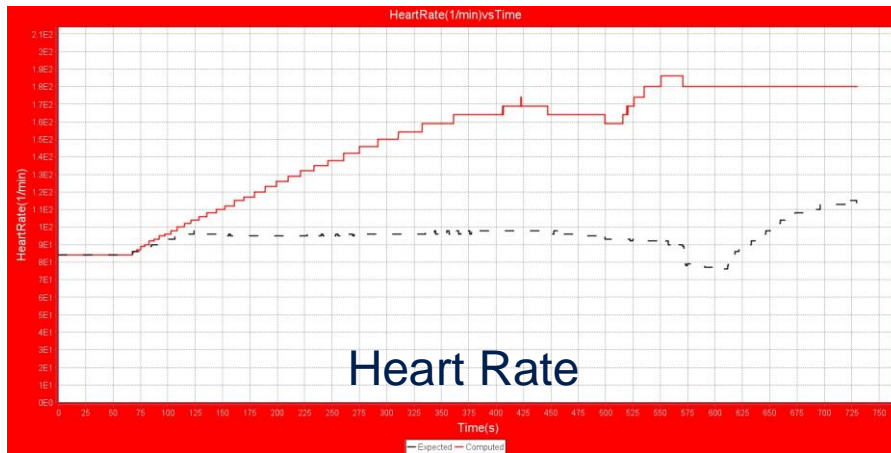
1. Open \toolkit\Scenarios\Showcase\CombatMultitrauma.xml in a text editor (Notepad, Wordpad, Notepad++, Sublime, etc.)
2. Change the rate on the hemorrhage in Segment 2 from 350 to 500
3. Remove the second hemorrhage (setting the flow to 50mL)
4. Save file and rerun the scenario in the GUI

```

<!-- Segment 1: Initialization -->
<Action xsi:type="AdvanceTimeData">
    <Time value="1" unit="min"/>
</Action>
<!-- Segment 2: Begin Pneumothorax and Hemorrhage -->
<Action xsi:type="TensionPneumothoraxData" Type="Closed" Side="Left">
    <Comment>Insult</Comment>
    <Severity value="0.75"/>
</Action>
<Action xsi:type="HemorrhageData" Compartment="RightLeg">
    <Comment>Insult: Massive bleeding</Comment>
    <Rate value="500" unit="mL/min"/>
</Action>
<Action xsi:type="AdvanceTimeData">
    <Time value="1" unit="min"/>
</Action>
    
```

# Modifying a Scenario File

- Plot images shown are the calculated (red) data compared to results data from an unmodified scenario. GUI plots will match the red line.



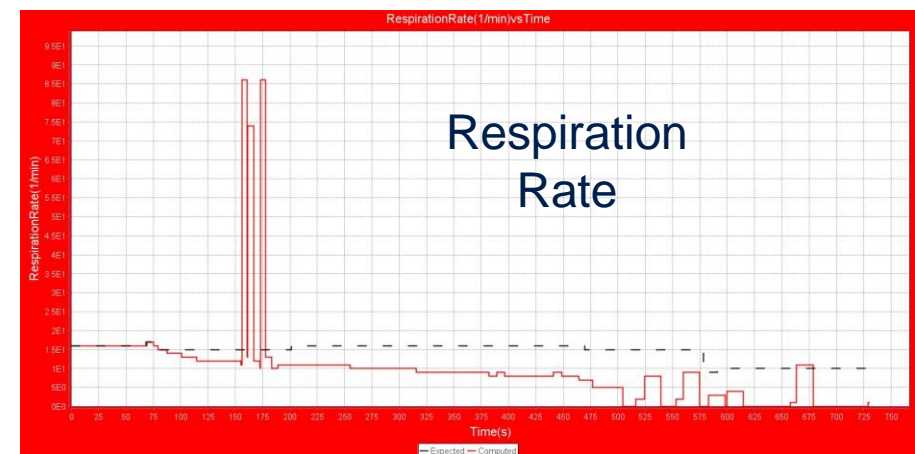
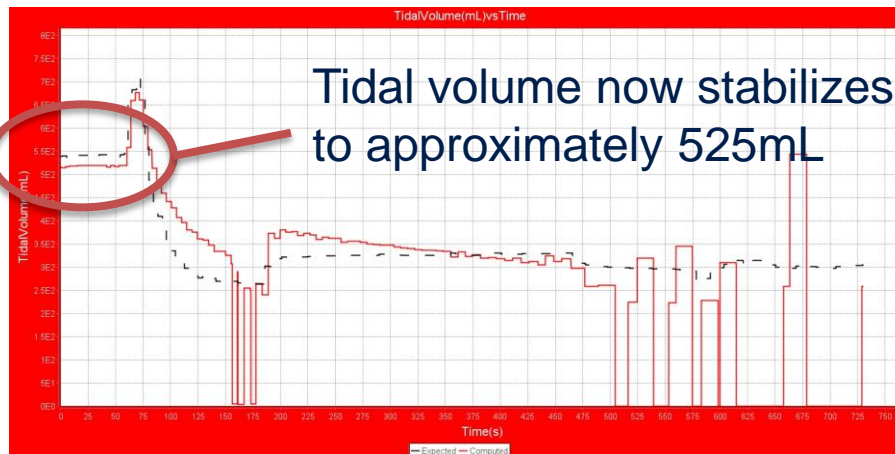
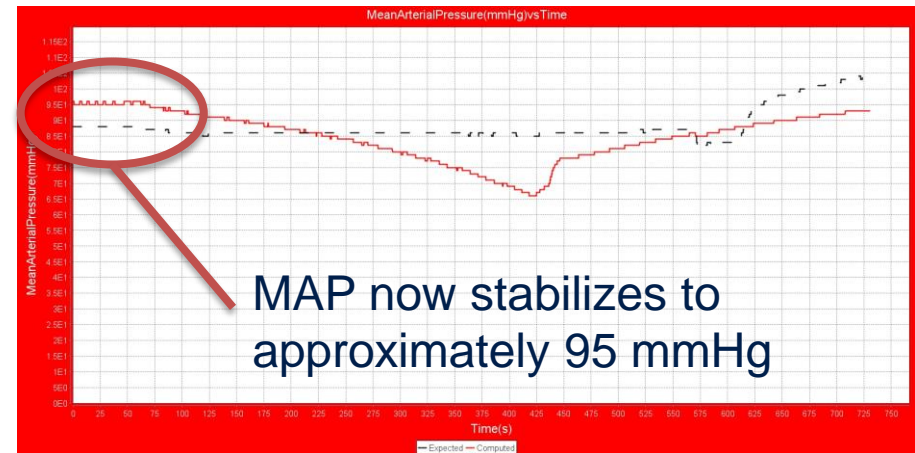
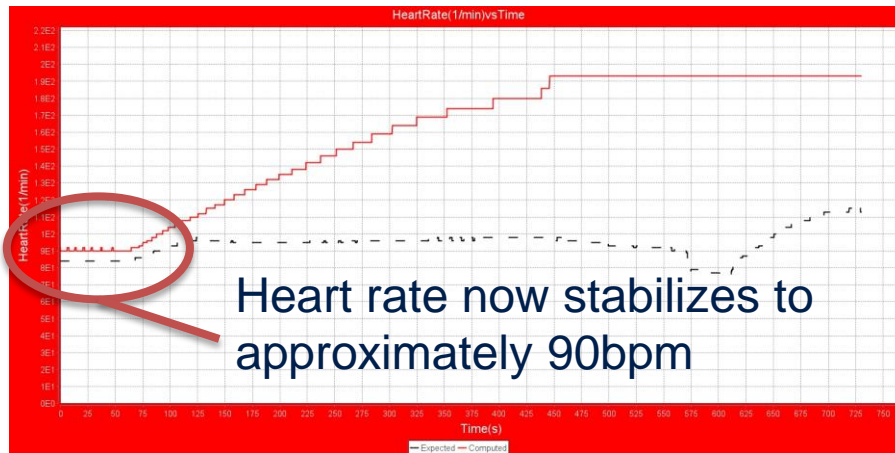
# Modifying a Patient File

1. Open \toolkit\patients\Soldier.xml
2. Change the weight from 170.0 to 145.0
3. Change the heart rate baseline from 84.0 to 90.0
4. Change the mean arterial pressure baseline from 87 to 95
5. Save file and run CombatMultitrauma.xml in the GUI

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Patient xmlns="uri:/mil/tatrc/physiology/datamodel" xmlns:xsi:
  <Name>Soldier</Name>
  <Gender>Male</Gender>
  <Age value="22.0" unit="yr"/>
  <Weight value="145.0" unit="lb"/>
  <Height value="71.0" unit="in"/>
  <BodyFatFraction value="0.18"/>
  <CarinaToTeethDistance value="22.0" unit="mm"/>
  <HeartRateBaseline value="90.0" unit="1/min"/>
  <MeanArterialPressureBaseline value="95.0" unit="mmHg"/>
  <RespirationRateBaseline value="16.0" unit="1/min"/>
  <RightLungRatio value="0.55"/>
  <TotalBloodVolumeBaseline value="5500.0" unit="mL"/>
</Patient>
```



# Modifying a Patient File

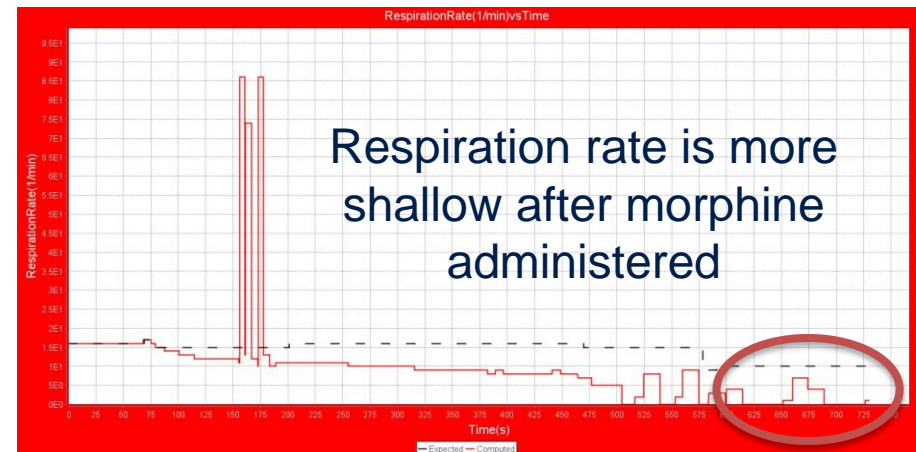
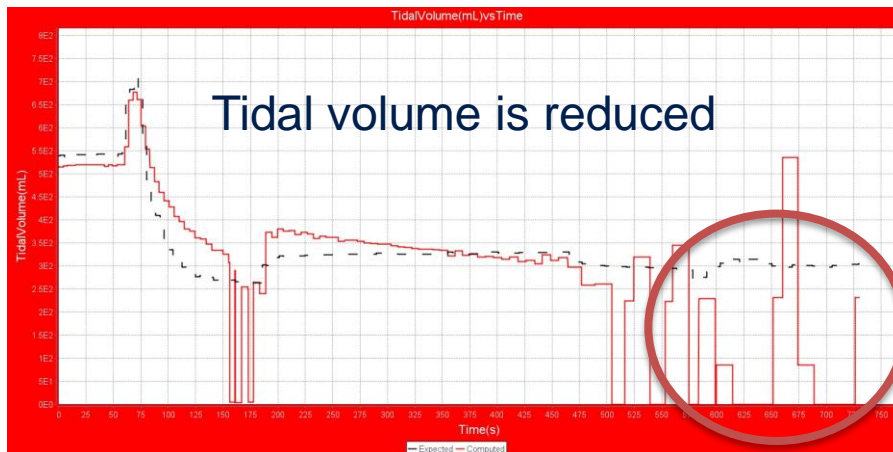
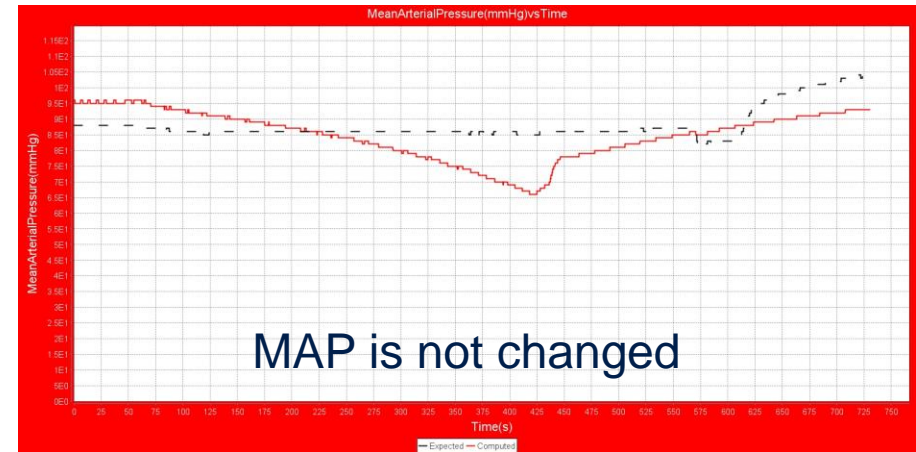
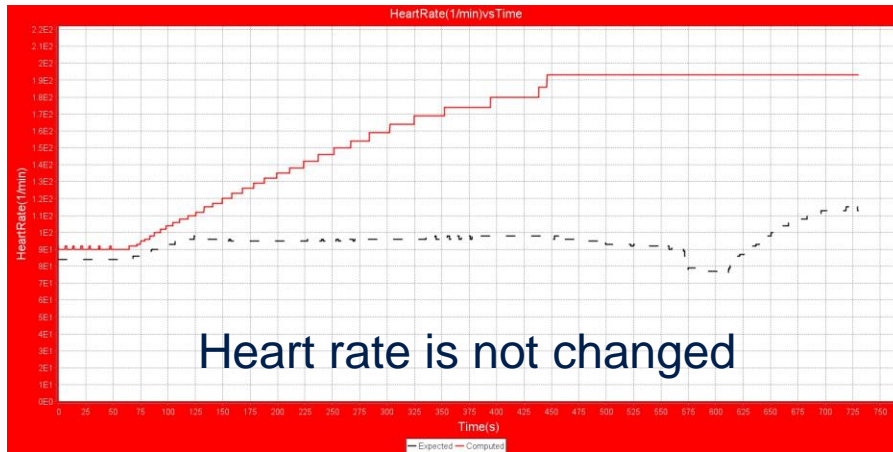


# Modifying a Drug File

1. Open \toolkit\substances\Morphine.xml
2. Change the Respiration Rate Modifier from -0.35 to -0.65
3. Save file and run CombatMultitrauma.xml in the GUI

```
<Pharmacokinetics>
  <AcidDissociationConstant value="8.9"/>
  <BindingProtein>Albumin</BindingProtein>
  <BloodPlasmaRatio value="1.02"/>
  <FractionUnboundInPlasma value="0.65"/>
  <IonicState>WeakBase</IonicState>
  <LogP value="0.89"/>
</Pharmacokinetics>
<Pharmacodynamics>
  <Bronchodilation value="0.0"/>
  <DiastolicPressureModifier value="-0.07"/>
  <EC50 value="0.0623" unit="ug/mL"/>
  <HeartRateModifier value="-0.1"/>
  <NeuromuscularBlock value="0.0"/>
  <PupilState>Constricted</PupilState>
  <RespirationRateModifier value="-0.65"/>
  <Sedation value="0.25"/>
  <SystolicPressureModifier value="-0.07"/>
  <TidalVolumeModifier value="0.0"/>
</Pharmacodynamics>
```

# Modifying a Drug File



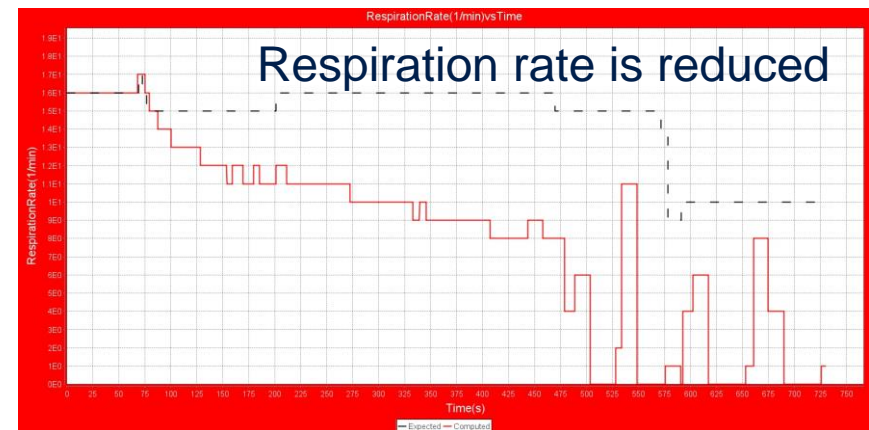
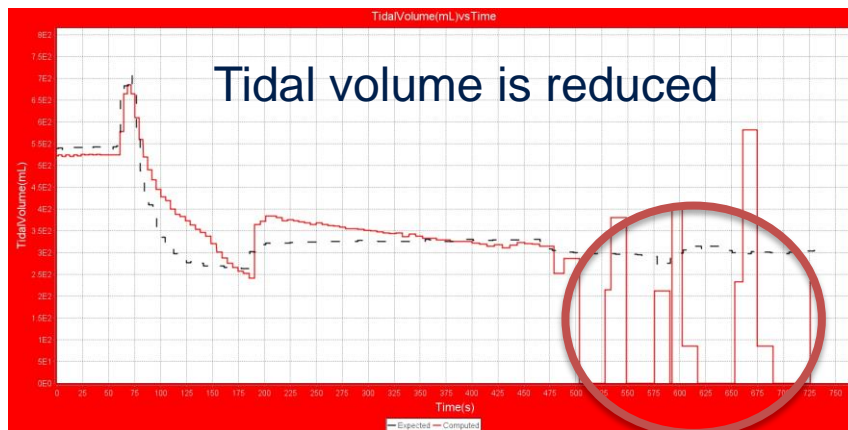
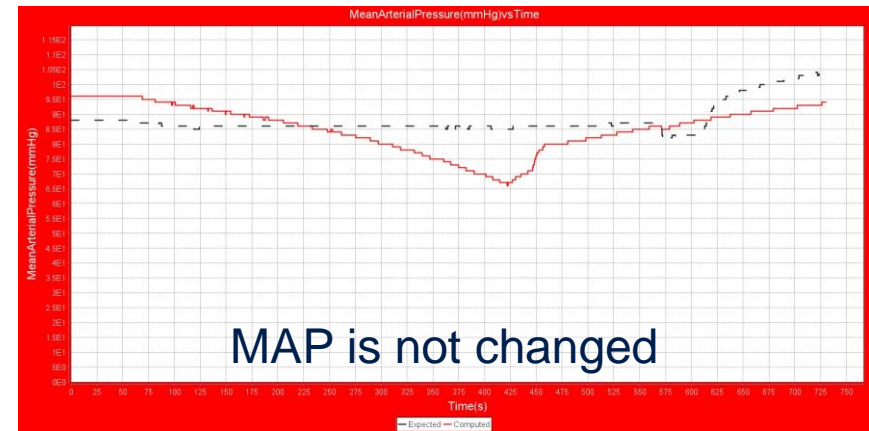
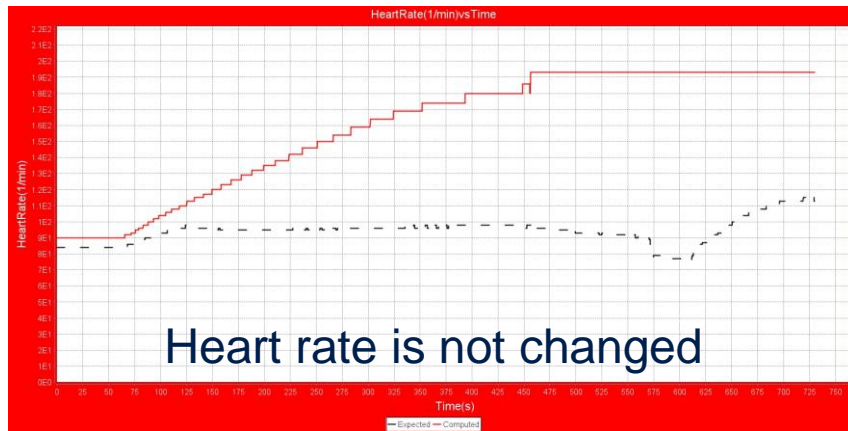
# Modifying the Environment

1. Open \toolkit\environments\Standard.xml in a text editor
  1. By default all scenarios use the Standard environment
  2. You could change the environment via a condition as well
2. Change AmbientTemperature, MeanRadiantTemperature, RespirationAmbientTemperature to 10 degC
3. Save file and run CombatMultitrauma.xml in the GUI

```
<EnvironmentalConditions xmlns="uri:/mil/tatrc/physiology/datamo
  <SurroundingType>Air</SurroundingType>
  <AirVelocity value="0.1" unit="m/s"/>
  <AmbientTemperature value="10.0" unit="degC"/>
  <AtmosphericPressure value="760.0" unit="mmHg"/>
  <ClothingResistance value="0.5" unit="clo"/>
  <Emissivity value="0.95"/>
  <MeanRadiantTemperature value="10.0" unit="degC"/>
  <RelativeHumidity value="0.6"/>
  <RespirationAmbientTemperature value="10.0" unit="degC"/>
```



# Modifying an Environment File



# USING THE BIOGEARS SDK

## BioGears SDK

- BioGears provides an SDK that contains example code files as well as headers and prebuilt binaries for:
  - Windows Visual Studio (2013)
  - Windows MinGW (GCC)
  - Mac Xcode (Clang)
  - Ubuntu Linux (GCC)
  - Rasbian for Raspberry Pi (GCC)
- There are numerous HowTo\*.cpp files that are stand alone example of creating an engine and using the CDM to interact with a BioGears engine
  - There is a MSVC solution and project provided for building an executable that will run each How-To example driver

# About the Common Data Model (CDM)

- The CDM is a large set of small classes that are used for data encapsulation and exchange
- Inputs and outputs between the user and a physiology engine are CDM objects
- CDM is defined in XML Schema
  - Can Autogenerate classes in various languages (C++/Java/C#) that can Serialize
- CDM Objects separate the data containing from the data manipulation
- CDM classes follow a hierarchical property bag design
  - Classes contain other child classes and/or properties
    - Enumeration – GetGender, SetGender, HasGender, InvalidateGender
      - Environment Surrounding Type is an Enum of Air or Water
    - String – GetName, SetName, HasName, InvalidateName
    - SEScalar – HasHeartRate, GetHeartRate
      - Basically a double, a unit and a unit conversion algorithm
        - Unless it's a unitless quantity, such as a volume fraction, severity, etc.
      - Must provide a unit when setting and getting a value to an SEScalar
      - Most properties are an SEScalar type (Heart Rate, Volumes, etc.)
    - SEFunction – HasLungVolumePlot, GetLungVolumePlot
      - A Domain/Abscissa array and Range/Ordinate array of equal size
      - Pulmonary Function Test contains the LungVolumePlot of Volume Vs. Time
- [www.biogearsengine.com/documentation/\\_c\\_d\\_m.html](http://www.biogearsengine.com/documentation/_c_d_m.html)



# Common Data Model Object Library

- Scalar/Function Types
- Patient
- Patient Actions
- Patient Conditions
- Compartment (Volume, Pressure, Flows)
  - Gas, Liquid, Thermal, and Tissue types
- Compartment Substance
  - Mass/Concentration
  - Volume/Volume Fraction
- Substance
- Substance Manager
- Physiology Systems
  - Respiratory, CV, etc.
- Equipment Systems
  - Anesthesia, ECG, Inhaler
- Equipment Actions
  - Anesthesia and Inhaler
- Environment
- Environment Actions
- Environment Conditions
- Circuits (Nodes, Paths)
- Configuration
- Scenario
  - Data Requests, Collection Containers

Complete CDM Library: [www.biogearsengine.com/documentation/modules.html](http://www.biogearsengine.com/documentation/modules.html)

# Common Data Model Algorithms

- **Generic Circuit Solver**
  - Created to extract the lumped parameter physics calculations from within the individual systems and centralize the generic calculations
- **Generic Unit Converter**
  - Extensible conversion engine that supports 50+ quantity types (mass, volume, pressure, power, volume/time, amount, length, resistances, area, etc.)
- **Generic Substance Transport**
  - Substance mass/concentration, volume/volume fraction, and/or partial pressure are calculated based on the assumption that substances travel with the fluid flow of the circuit
- **Generic Scenario Executor**
  - Given an engine, will process a scenario and execute all commands while writing data from the engine to a comma delimited file

# PhysiologyEngine Interface

- BioGears\_sdk\include\cdm\engine\PhysiologyEngine.h
- The PhysiologyEngine interface is a generic interface for controlling any physiology engine via CDM objects
  - An engine will provide a Create method that will instantiate a specific engine and return a PhysiologyEngine Interface pointer for controlling it

```
BIOGEARS_API std::unique_ptr<PhysiologyEngine> CreateBioGearsEngine(const std::string& logfile="");
BIOGEARS_API std::unique_ptr<PhysiologyEngine> CreateBioGearsEngine(Logger* logger=nullptr);
```

- std::unique\_ptr will auto delete the pointer when it goes out of scope
- If a string is passed in a log file of that name will be created in the application working directory with the provided name, if empty string is passed, no log file will be created
- You may pass in a Logger class to use as well, if you are using one for your application (more on that later)
- Example of creating a BioGears engine:

```
std::unique_ptr<PhysiologyEngine> bg = CreateBioGearsEngine("HowToAirwayObstruction.log");
```

# PhysiologyEngine Interface

- Deprecated Methods
  - These methods are being removed or replaced for state serialization

```
//-----
/// \brief
/// This will set the current time back to 0, and ready the engine to start over
/// with everthing reinitialized.
/// Input data, like the patient file, will be kept
//-----
virtual void Reset() = 0;

//-----
/// \brief
/// This will reinitialize/nullptr out ALL data associated with the engine
/// Including inputs like the patient file, substances etc
//-----
virtual void Clear() = 0;

//-----
/// \brief
/// Check to see that the engine has been configured and set up properly
//-----
virtual bool IsValid() = 0;
```

# PhysiologyEngine Interface Timing Methods

- Methods for getting engine time step size and current times

```
//-----
/// \brief
/// returns the engine time step that is used when advancing time.
///
//-----
virtual double GetTimeStep(const std::shared_ptr<CCompoundUnit>& unit) = 0;

//-----
/// \brief
/// returns the current internal engine time.
///
//-----
virtual double GetEngineTime(const std::shared_ptr<CCompoundUnit>& unit) = 0;

//-----
/// \brief
/// returns the current scenario time.
/// The engine may run a certain time to get to a steady state,
/// this does not account for that time, the scenario starts from steady state
/// THIS WILL CHANGE TO GetSimulationTime NEXT VERSION
///
//-----
virtual double GetScenarioTime(const std::shared_ptr<CCompoundUnit>& unit) = 0;
```

- Example of using this later

# PhysiologyEngine Interface Initialization

- Conditions are an optional chronic state that require the engine to restabilize the patient
- Conditions must be provided to the engine at initialization

```
//-----
/// \brief
/// locates the xml patient file and reads in the values.
///
/// This will create an engine that you can send instructions (patient,actions,conditions) to dynamically.
/// The return value will indicate success failure of the creation of the engine.
/// Some combinations of patients and conditions may prevent the engine from stabilizing
///
//-----
virtual bool InitializeEngine(const std::string& patientFile, const std::vector<const SECondition*>* conditions = nullptr) = 0;

//-----
/// \brief
///
/// This will create an engine that you can send instructions (patient,actions,conditions) to dynamically.
/// The return value will indicate success failure of the creation of the engine.
/// Some combinations of patients and conditions may prevent the engine from stabilizing
///
//-----
virtual bool InitializeEngine(const SEPatient& patient, const std::vector<const SECondition*>* conditions = nullptr) = 0;
```

- Using multiple conditions is permitted, but the engine may or may not stabilize due to combined effects

# PhysiologyEngine Interface Initialization

- Initializing the engine with patient only

```
if (!bg->InitializeEngine("Standard.xml"))
{
    std::cerr << "Could not load initialize engine, check the error\n";
    return;
}
```

- Initializing the engine with a patient and condition(s)

```
SEChronicObstructivePulmonaryDisease COPD;
COPD.GetBronchitisSeverity().SetValue(0.5);
COPD.GetEmphysemaSeverity().SetValue(0.7);
std::vector<const SECondition*> conditions;
conditions.push_back(&COPD);

if (!bg->InitializeEngine("Standard.xml", &conditions))
{
    std::cerr << "Could not load initialize engine, check the error" << std::endl;
    return;
}
```

- BioGears currently takes a minimum of 2-3 min to initialize
- Condition class headers, data definitions, and example data:
  - BioGears\_sdk\include\cdm\patient\conditions
  - BioGears\_sdk\include\cdm\system\environment\conditions
  - [www.biogearsengine.com/documentation/c\\_d\\_m.html](http://www.biogearsengine.com/documentation/c_d_m.html)
  - [www.biogearsengine.com/documentation/scenario\\_x\\_m\\_l\\_file.html](http://www.biogearsengine.com/documentation/scenario_x_m_l_file.html)
  - [www.biogearsengine.com/documentation/system\\_methodology.html](http://www.biogearsengine.com/documentation/system_methodology.html)

# PhysiologyEngine Interface Input

- Advance Time

```
//-----
/// \brief
/// executes one pass through the time loop of the engine at the fixed timestep
///
/// Events, errors, and warning as accessed via the Logger or EventHandler.
///
//-----
virtual void AdvanceModelTime() = 0;

//-----
/// \brief
/// executes time loop of the engine beginning at the current time
/// and running for the duration specified in the call at the fixed timestep
///
/// Events, errors, and warning as accessed via the Logger or EventHandler.
///
//-----
virtual void AdvanceModelTime(double time, const std::shared_ptr<CCompoundUnit>& unit) = 0;
```

- Example:

```
bg->AdvanceModelTime(300, SEScalarTime::s);
```



# PhysiologyEngine Interface Input

- Process Action

```
//-----
/// \brief
/// Execute the provided action.
/// true will be returned if the engine supports the action
/// false will be returned if the engine does not support the action
///
//-----
virtual bool ProcessAction(const SEAction& action) = 0;
```

- Example:

```
SEAirwayObstruction obstruction;
obstruction.GetSeverity().SetValue(0.6); // 0 (off) to 1 (full obstruction)
bg->ProcessAction(obstruction);
std::cout << "Giving the patient an airway obstruction.\n\n";
```

- Action class headers, data definitions, and example data:

- BioGears\_sdk\include\cdm\patient\actions
- BioGears\_sdk\include\cdm\system\environment\actions
- BioGears\_sdk\include\cdm\system\equipment\Anesthesia\actions
- BioGears\_sdk\include\cdm\system\equipment\Inhaler\actions
- [www.biogearsengine.com/documentation/\\_c\\_d\\_m.html](http://www.biogearsengine.com/documentation/_c_d_m.html)
- [www.biogearsengine.com/documentation/\\_scenario\\_x\\_m\\_l\\_file.html](http://www.biogearsengine.com/documentation/_scenario_x_m_l_file.html)
- [www.biogearsengine.com/documentation/\\_system\\_methodology.html](http://www.biogearsengine.com/documentation/_system_methodology.html)

# PhysiologyEngine Interface Output

- Get CDM objects for patient, physiology systems, equipment or the environment

```
virtual const SEGastrointestinalSystem* GetGastrointestinalSystem() = 0;
virtual const SERespiratorySystem* GetRespiratorySystem() = 0;
virtual const SEDrugSystem* GetDrugSystem() = 0;
virtual const SEAnesthesiaMachine* GetAnesthesiaMachine() = 0;
virtual const SEElectroCardioGram* GetElectroCardioGram() = 0;
virtual const SEInhaler* GetInhaler() = 0;
virtual const SEPatient& GetPatient() = 0;
virtual const SEEnvironment* GetEnvironment() = 0;
virtual const SEBloodChemistrySystem* GetBloodChemistrySystem() = 0;
virtual const SECardiovascularSystem* GetCardiovascularSystem() = 0;
virtual const SEEndocrineSystem* GetEndocrineSystem() = 0;
virtual const SEEnergySystem* GetEnergySystem() = 0;
virtual const SERenalSystem* GetRenalSystem() = 0;
```

- Example:

```
std::cout << "Respiration Rate : " << bg->GetRespiratorySystem()->GetRespirationRate(SEScalarFrequency::Per min) << "bpm" << std::endl;
```

- BioGears does support all of these objects
- Class headers, data definitions, and example data:
  - BioGears\_sdk\include\cdm\patient\
  - BioGears\_sdk\include\cdm\system\physiology
  - BioGears\_sdk\include\cdm\system\environment\
  - BioGears\_sdk\include\cdm\system\equipment\Anesthesia
  - BioGears\_sdk\include\cdm\system\equipment\ElectroCardioGram\
  - BioGears\_sdk\include\cdm\system\equipment\Inhaler\
  - [www.biogearsengine.com/documentation/\\_c\\_d\\_m.html](http://www.biogearsengine.com/documentation/_c_d_m.html)
  - [www.biogearsengine.com/documentation/\\_system\\_methodology.html](http://www.biogearsengine.com/documentation/_system_methodology.html)

# PhysiologyEngine Interface Output

- Compartments contain flows, pressure, volume, and substance amounts on patient anatomy or equipment

```
virtual const SEAnatomyCompartments* GetAnatomyCompartments() = 0;
virtual const SEAnesthesiaMachineCompartments* GetAnesthesiaMachineCompartments() = 0;
virtual const SEInhalerCompartments* GetInhalerCompartments() = 0;
```

- Example:

```
bg->GetAnatomyCompartments()->GetPulmonaryCompartment(CDM::enumAnatomy::Trachea)->GetInFlow(SESscalarVolumePerTime::mL_Per_s);
```

- Substance amounts are addressed later
- Class headers, data definitions, and example data:
  - BioGears\_sdk\include\cdm\compartment
  - [www.biogearsengine.com/documentation/\\_c\\_d\\_m.html](http://www.biogearsengine.com/documentation/_c_d_m.html)

# PhysiologyEngine Interface Output

- Assessments provide a clinical assessment object for the engine to populate data into (ex. Pulmonary Function Test, Urinalysis, etc.)

```
//-----
/// \brief
/// Determines the assessment type and fills the data object with current data.
///
/// Assessments can be queried at any point in the calculation and as many times are desired.
//-----
virtual bool GetPatientAssessment(SEPatientAssessment& assessment) = 0;
```

- Assessments can take extra computing time to fill out
- Example:

```
SEPulmonaryFunctionTest pft(bg->GetLogger());
bg->GetPatientAssessment(pft);
std::cout << "Expiratory Reserve Volume" << pft.GetExpiratoryReserveVolume() << "\n";
// BioGears does compute the LungVolumePlot Data as a function of time
SEFunctionTimeVsVolume& lungVolumePlot = pft.GetLungVolumePlot();
lungVolumePlot.GetTime(); //This is the time component of the pulmonary function test
lungVolumePlot.GetVolume(); //This is the lung volume component of the pulmonary function test
```

- Class headers, data definitions, and example data:
  - BioGears\_sdk\include\cdm\patient\assessments
  - [www.biogearsengine.com/documentation/\\_c\\_d\\_m.html](http://www.biogearsengine.com/documentation/_c_d_m.html)
  - [www.biogearsengine.com/documentation/\\_system\\_methodology.html](http://www.biogearsengine.com/documentation/_system_methodology.html)

# Complete Working Example

```
void HowToAirwayObstruction()
{
    std::unique_ptr<PhysiologyEngine> bg = CreateBioGearsEngine("HowToAirwayObstruction.log");

    SEChronicObstructivePulmonaryDisease COPD;
    COPD.GetBronchitisSeverity().SetValue(0.5);
    COPD.GetEmphysemaSeverity().SetValue(0.7);
    std::vector<const SECondition*> conditions;
    conditions.push_back(&COPD);

    if (!bg->InitializeEngine("Standard.xml", &conditions))
    {
        std::cerr << "Could not load initialize engine, check the error" << std::endl;
        return;
    }

    std::cout << "Respiration Rate : " << bg->GetRespiratorySystem()->GetRespirationRate(SESscalarFrequency::Per_min) << "bpm" << std::endl;

    SEAirwayObstruction obstruction;
    obstruction.GetSeverity().SetValue(0.6); // 0 (off) to 1 (full obstruction)
    bg->ProcessAction(obstruction);

    // Advance time to see how the obstruction affects the patient
    bg->AdvanceModelTime(90, SEScalarTime::s);
    std::cout << bg->GetAnatomyCompartment()->GetPulmonaryCompartment(CDM::enumAnatomy::Trachea)->GetInFlow(SESscalarVolumePerTime::mL_Per_s);

    obstruction.GetSeverity().SetValue(0.0); // You can remove an obstruction by setting the severity to 0
    bg->ProcessAction(obstruction); // This will remove the blockage to open the airway and the patient will recover.

    bg->AdvanceModelTime(300, SEScalarTime::s);
    std::cout << "Respiration Rate : " << bg->GetRespiratorySystem()->GetRespirationRate(SESscalarFrequency::Per_min) << "bpm" << std::endl;

    SEPulmonaryFunctionTest pft(bg->GetLogger());
    bg->GetPatientAssessment(pft);
    std::cout << "Expiratory Reserve Volume" << pft.GetExpiratoryReserveVolume() << "\n";
    // BioGears does compute the LungVolumePlot Data as a function of time
    SEFunctionTimeVsVolume& lungVolumePlot = pft.GetLungVolumePlot();
    lungVolumePlot.GetTime(); // This is the time component of the pulmonary function test
    lungVolumePlot.GetVolume(); // This is the lung volume component of the pulmonary function test
}
```

# Advanced Features of the API

- Substances

```
//-----
/// \brief
/// Retrieves the associated substance manager.
//-----
virtual SESubstanceManager& GetSubstanceManager() = 0;
```

- Logging and Error Handling

```
//-----
/// \brief
/// Retrieve the Logger associated with this engine
//-----
virtual Logger* GetLogger() = 0;
```

- Patient and Equipment Events

```
//-----
/// \brief
/// Add a callback object that will be called whenever a patient or anesthesia machine event changes state
//-----
virtual void SetEventHandler(SEEEventHandler* handler) = 0;
```

- Tracking Data and Writing a File

```
//-----
/// \brief
/// Retrieve the PhysiologyEngineTrack associated with tracking data from this engine to a file
//-----
virtual PhysiologyEngineTrack* GetEngineTrack() = 0;
```

# How To Use Substances

- Each engine has its own instance of a substance (i.e. Oxygen), you will need to get the substance object associated with the engine, the PhysiologyEngine interface has a method to get the substance manager associated with an engine

- ```
SESubstance* O2 = bg->GetSubstanceManager().GetSubstance("Oxygen");
SESubstance* CO2 = bg->GetSubstanceManager().GetSubstance("CarbonDioxide");
```
- Substances and their names can be found in the substance xml files in the /toolkit/bin/substance directory

- Substance objects are required for actions as well as if you are interested in substance data in the body and on compartments

```
O2->GetMassInBlood().GetValue(SESscalarMass::g);
double vol_mL = bg->GetAnatomyCompartments()->GetPulmonaryCompartment(CDM::enumAnatomy::Trachea)
->GetSubstanceQuantity(*O2)->GetVolume(SESscalarVolume::mL);
```

- Class headers, data definitions, and example data:
  - BioGears\_sdk\include\cdm\substance
  - [www.biogearsengine.com/documentation/\\_c\\_d\\_m.html](http://www.biogearsengine.com/documentation/_c_d_m.html)

# How To Use Logging

- You have the option to pass in a string or your own Logger class when creating a BioGears Engine
  - The string is used as the file name when creating the log file
  - You can share a Logger between your app and an engine
  - Might be confusing to share a Logger between engines
- The Logger is based on Log4cpp, which is based on Log4j
  - LogLevels: Debug, Info (default), Warn, Error, Fatal
- You can grab the Logger\* from your engine and log information to the log directly
  - `bg->GetLogger()->Info("Your Message Here");`
- All BioGears logging will go to that file as well as the console, if specified
- Class headers, data definitions, and example data:
  - BioGears\_sdk\include\cdm\utils\Logger.h
  - [www.biogearsengine.com/documentation/\\_c\\_d\\_m.html](http://www.biogearsengine.com/documentation/_c_d_m.html)



# How To Do Error Handling

- As BioGears logs all errors and warnings, you can programmatically listen to the Logger and perform custom code when the engine logs these errors/warnings
- Simply extend the LoggerForward class

```
class MyForwarder : public LoggerForward
{
public:
    virtual void ForwardDebug (const std::string& msg, const std::string& origin) { /* Your code here*/ }
    virtual void ForwardInfo(const std::string& msg, const std::string& origin) { /* Your code here*/ }
    virtual void ForwardWarning(const std::string& msg, const std::string& origin) { /* Your code here*/ }
    virtual void ForwardError(const std::string& msg, const std::string& origin) { /* Your code here*/ }
    virtual void ForwardFatal(const std::string& msg, const std::string& origin) { /* Your code here*/ }
};
```

- Add your class to the Logger, and it will call your methods

```
MyForwarder MyErrorHandler;
bg->GetLogger()->SetForward(&MyErrorHandler);
```

- Class headers, data definitions, and example data:
  - BioGears\_sdk\include\cdm\utils\Logger.h
  - [www.biogearsengine.com/documentation/\\_c\\_d\\_m.html](http://www.biogearsengine.com/documentation/_c_d_m.html)

# How To Get Events

- During the execution of the engine, depending on what you do, the patient and the anesthesia machine can go into and out of certain states.
  - Hypercapnia, asystole, etc.  
[https://www.biogearsengine.com/documentation/group\\_patient\\_enum\\_patient\\_event.html](https://www.biogearsengine.com/documentation/group_patient_enum_patient_event.html)
  - O2 Bottle Exhausted, Relief Valve is Active, etc.  
[https://www.biogearsengine.com/documentation/group\\_anesthesia\\_enum\\_anesthesia\\_machine\\_event.html](https://www.biogearsengine.com/documentation/group_anesthesia_enum_anesthesia_machine_event.html)
- An 'event' is the act of entering or exiting a specific state, you can query for, or be notified of events by extending sdk\cdm\utils\SEEventHandler.h
- Example:

```
// There are specific events that can occur while the engine runs and you submit various actions
// You can either poll/query the patient object to see if it is in a specific state
bool b = bg->GetPatient().IsEventActive(CDM::enumPatientEvent::CardiacArrest);
// You can also derive a callback class that will be called whenever an Event is entered or exited by the patient
MyEventHandler myEventHandler(bg->GetLogger());
bg->SetEventHandler(&myEventHandler);
```

```
class MyEventHandler : public SEEventHandler
{
public:
    MyEventHandler(Logger *logger) : SEEventHandler(logger) {}
    virtual void HandlePatientEvent(CDM::enumPatientEvent::value type, bool active, const SEScalarTime* time = nullptr) {}
    virtual void HandleAnesthesiaMachineEvent(CDM::enumAnesthesiaMachineEvent::value type, bool active, const SEScalarTime* time = nullptr) {}
};
```

# How To Create A Results File

- The PhysiologyEngine Interface provides a method to get a PhysiologyEngineTrack object from the engine
- The PhysiologyEngineTrack takes in a list of SEDataRequest objects and will automatically pull the values of those properties from the engine and store them and/or write them to a comma delimited file on disk

```
SEPhysiologySystemDataRequest* physRequest = new SEPhysiologySystemDataRequest();
physRequest->Set("OxygenSaturation");
tracker.m_Requests.push_back(physRequest);
bg->GetEngineTrack()->RequestData(tracker.m_Requests, "HowToPulmonaryFunctionTest.txt");
```

- The tracker object is an instance of the HowToTracker class in the How-To examples used to advance the engine in time and as it does so, also directs the PhysiologyEngineTrack to write data to a file per time step

```
void HowToTracker::AdvanceModelTime(double time_s)
{
    int count = static_cast<int>(time_s / m_dT_s); // This samples the engine at each time step
    for (int i = 0; i <= count; i++)
    {
        m_Engine.AdvanceModelTime(); // Compute 1 time step
        // Pull Track will pull data from the engine and append it to the file
        m_Engine.GetEngineTrack()->TrackData(m_Engine.GetScenarioTime(SESscalarTime::s));
    }
}
```

## How-To Library

- Everything discussed is demonstrated in the `HowTo-EngineUse.cpp` file with more detailed comments
- As we add new functionality to BioGears and have time we add more HowTo files demonstrating how to use specific functionality through the CDM and PhysiologyEngine API
- The Current Library :
  - `HowTo-Airway Obstruction`
  - `HowTo-BolusDrug`
  - `HowTo-ConsumeNutrients`
  - `HowTo-COPD`
  - `HowTo-CPR`
  - `HowTo-EnvironmentChange`
  - `HowTo-Exercise`
  - `HowTo-Hemorrhage`
  - `HowTo-LobarPnumonia`
  - `HowTo-PulmonaryFunctionTest`
  - `HowTo-RunScenario`
  - `HowTo-TensionPneumothorax`
  - `HowTo-ConcurrentEngines`

## Contact

[www.biogearsengine.com/workwithus](http://www.biogearsengine.com/workwithus)

### **Aaron Bray**

*Lead Software Architect*

[abray@ara.com](mailto:abray@ara.com)

919-582-3333

### **Rachel Clipp, PhD**

*Lead Physiology Modeler*

[rclipp@ara.com](mailto:rclipp@ara.com)

919-582-3330

### **Jeff Webb**

*Principal Investigator*

[jwebb@ara.com](mailto:jwebb@ara.com)

919-582-3435