imstk Documentation

Release 6.0.0

arikatla

Jun 22, 2022

User Guide

Interactive Medical Simulation Toolkit (iMSTK)

User Documentation

1 Introduction

iMSTK is an open-source C++-based software toolkit that aids rapid prototyping of interactive multi-modal medical simulations. It provides a highly modular and easy-to-use framework that can be extended and interfaced with third-party libraries for the development of medical simulators without restrictive licenses.

iMSTK supports all three major platforms-macOS, Linux, Windows and comes with a CMake-based automatic build system. This documentation is designed to provide an overview of iMSTK, introductory concepts needed to comprehend the framework, its component modules, and how they interact to help build simulations. For a detailed insight into the usage of various modules, you may refer to an extensive set of examples that are built by default. For implementational level details of the modules and their classes, please refer to the code documentation. The chapters that follow will describe details of how to build iMSTK, elements of the simulation scenario, and how these elements are connected using iMSTK's modular architecture followed by a detailed description of each of the major modules. The final chapter includes a walk-through of the code of an all-inclusive example to help the readers quickly build their application.

2 Setup for Development

iMSTK and its external dependencies can be configured and built from scratch using iMSTK's CMake super-build (build project dependencies and then the project) on UNIX (MAC, Linux) and Windows platforms.

2.1 Configuration and Build

CMake should be used to configure the project on every platform. Please refer to CMake's official page to read about how to configure using CMake.

Linux/MacOSx

Type the following commands from the same location you cloned the

```
>> mkdir iMSTK_build
>> cd iMSTK_build
>> cmake ../iMSTK    # path to source directory
>> make -j4 #to build using *4* cores
```

This will configure the build in a directory adjacent to the source directory. To easily change some configuration variables such as `CMAKE_BUILD_TYPE`, use ccmake instead of cmake.

One can also use Ninja for a faster build instead of Unix Makefiles. To do so, configure the cmake project with `-GNinja`

```
>> cmake -GNinja
>> ../iMSTK
>> ninja
```

This will checkout, build and link all iMSTK dependencies. When making changes to iMSTK base source code, you can then build from the Innerbuild directory.

Windows

Run CMake-GUI and follow the directions described on CMake's official page. You need to choose which version of Visual Studio that you would like to use when configuring the project. Make sure to select Microsoft Visual Studio C++ 12 2015 or later. CMake will generate a iMSTK.sln solution file for Visual Studio inside the build directory. Open this file and issue build on `ALL_BUILD` target, which will perform the superbuild.

The above procedures will configure and build iMSTK in the build directory (*iMSTK_build*). To easily change some configuration variables such as `CMAKE_BUILD_TYPE`, use ccmake instead of cmake. When using the CMake-GUI, one can edit the configuration directly.

Once the superbuild is complete, you may notice that (a) all of iMSTK's dependencies are cloned and built inside their respective folders inside `<iMSTK_build_dir>/External` (b) iMSTK itself is built inside the `<iMSTK_build_dir>/Innerbuild`. Beyond this point, when making changes to iMSTK base source code, one can then build from the "innerbuild" target.

Note that one can create and maintain multiple builds from the same source. For example, as is typical to interactive applications testing typically is performed with the Release build while debugging using the Debug build. Additionally, it is usually recommended to separate the build and source directories for cleaner development and maintenance esp. with git-based workflows.

2.2 Running Examples

The default CMake configuration builds the examples as part of the inner build. The executables including other targets required to run the executables are placed in the `<imstk build dir>/install/bin` directory. The execurables can either be run through command line or double clicking.

2.3 Options at Configure Time

Phantom Omni Support (haptics)

To support the Geomagic Touch (formerly Sensable Phantom Omni) haptic device, follow the steps below:

1. Install the OpenHaptics SDK as well as the device drivers:
    a. for Windows
    b. for Linux
2. Reboot your system.
3. Configure your CMake project with the variable `iMSTK_USE_OpenHaptics` set to ON.
4. After configuration, the CMake variable OPENHAPTICS_ROOT_DIR should be set to the OpenHaptics path on your system.

Note: The examples that depend on this option being on at configure time will not build automatically if this option is not selected.

Building Examples

The examples that demonstrate the features and the usage of iMSTK API can be optionally build. Set `BUILD_EXAMPLES` to ON the examples needs to be built.

Offscreen Rendering

Offscreen OSMesa/CPU rendering is supported for Linux/MacOSx. This allows one to build iMSTK without a screen or GPU. Useful for servers. This can be enabled in the build by using the `iMSTK_USE_VTK_OSMESA` flag to ON.

Audio Support

iMSTK has the ability to play audio streams at runtime. In order to enable Audio, set `iMSTK_ENABLE_AUDIO` to ON.

Uncrustify Support

iMSTK follows specific code formatting rules. This is enforced through Uncrustify. For convenience, iMSTK provides the option to build uncrustify as a target. To enable this set `iMSTK_USE_UNCRUSTIFY` to ON.

Multithreaded build

The build will be configured to be multi-threaded with 8 threads. This can be changed by modifying the `iMSTK_NUM_BUILD_PROCESSES` to a positive integer.

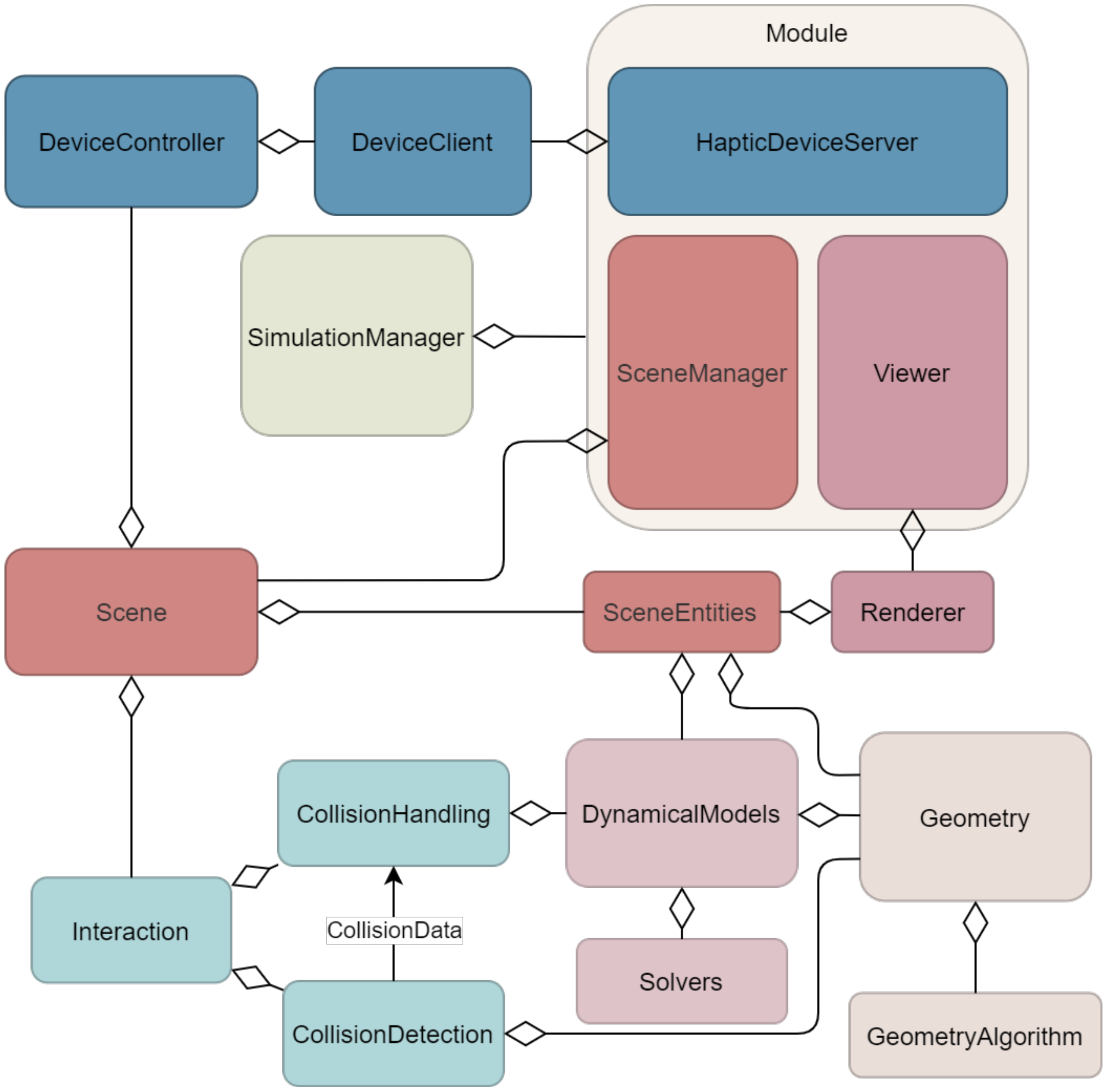
3 Overview of iMSTK

iMSTK is structured similar to many game engines with a focus on real time scene level rendering and simulating. However, iMSTK is aimed at surgical simulation. With this we can assume:

- Surgical scenes are being rendered. These are generally small, confined to an operating room at most.
- **Surgical simulation software's are often used for training. This requires a balance between how it feels and looks. Having**
    - Haptics often impose >1000 Hz performance requirements which limits rendering and other things one might do in their software.

- Interactions are fewer but far more complex in surgeries. Often requiring a wide variety of expensive dynamical models and collision detection + response to govern the physics of objects such as deformable, fluid, or rigid bodies.

The following diagram gives a high level view of iMSTK architecture. Most of the components shown in this high level view correspond to the APIs libraries. Here we will give a brief overview of every part with detailed pages for each should you choose to learn more.



3.1 Geometry

Geometries are used throughout iMSTK. The geometries themselves can be used standalone though. We have analytical, implicit, surface, volumetric, and image geometries.

3.2 Dynamical Models

Geometries are great. But you're likely going to want to do something with it. The primary use case is to advance it in time via some dynamical model. In iMSTK we provide models for dynamic and static deformable/soft bodies, fluids, & rigid bodies. We include PBD, SPH, FEM, and Rigid Bodies.

3.3 Geometric Filtering

What else can you do with geometries? Filtering! Our filtering library provides a set of geometry algorithms we have found useful for surgical simulations.

3.4 Devices

Devices are an important part to iMSTK. This is an the interactive surgical simulation toolkit after all.

- OpenVR: iMSTK, by default, builds with OpenVR. With a headset on, you may use OpenVR controllers for tools. See examples.
- Mouse & Keyboard: These are provided under the same API as our other devices for convenience.
- OpenHaptics: Allows one to use haptic tracking devices such as the Phantom Omni, these provide force feedback, under the right models we can stop your hand from moving when touching something, or give slight resistance.
- Coming Soon: VRPN, Bluetooth, Arduino ...

3.5 Controllers.

Controllers implement the controls of a device. We provide a couple of abstract base classes such as MouseControl, KeyboardControl, TrackingDeviceControl. As well as a few subclasses such as KeyboardSceneControl and MouseSceneControl which have some default behaviors such as stopping, starting, pausing a scene. But it's heavily encouraged you to subclass your own. You may also use lambdas on the devices for fast prototyping.

3.6 Collision Detection

Collision detection can be standalone in iMSTK but often finds it use through Interactions, later described in Scene. Put simply, its the act of computing "CollisionData" from two geometries. Most of the time these are "contacts" such as point normal contacts which give a point, a normal, and penetration depth. Often these are then later given to constraints to be added to a dynamical model.

3.7 Collision Handling

Collision handling implements how to consume collision data. For this reason it takes input CollisionData which is generally shared with CollisionDetection. iMSTK provides a number of handling methods, generally these call upon the functions of a DynamicalModel to immediately respond (explicit solve) or add something (such as a constraint) to later implicitly solve.

3.8 Scene

A scene contains a flat collection of SceneObjects and can fully represent the virtual environment. These SceneObjects may be something like an OR table, a tissue, a leg, a light, or even non-visual objects. Additionally a scene contains a set of interactions via it's InteractionGraph. A number of predefined InteractionPairs are available for iMSTK physics.

3.9 Mesh IO

Geometries are great. But to fully leverage them you need to be able to import from other tools which are much better at creating them. Read more about the files types supported by iMSTK. Additionally about Scene and SceneObject at the link above.

3.10 SimulationManager & Modules

Whilst scene's define the virtual environment and how to update it. They don't define how to drive it. You can certainly just call advance on the scene in a loop. That will get you decently far, but there's a bit more too it than that.

Modules in iMSTK define something that can be init'd, update'd, and uninit'd and added to a ModuleDriver. In every iMSTK example you can simply add modules to our concrete ModuleDriver called the SimulationManager to run them. It defines a special way of updating them.

3.11 Rendering

Rendering in iMSTK is done through delegation to support multiple backends. This means we setup delegate classes for each thing we want to render. And map what we want to render to what the backend allows us to render. Primarily we use VTK.

4 Miscellaneous Topics

4.1 Parallelism

Goes over loop, task, and module parallelism in iMSTK.

4.2 Events

Goes over events. Direct and message queues.

4.3 Computational Flow

Goes over the flow/advancement of a scene.

5 Releases

5.1 Release 6.0.0

Announcement: iMSTK 6.0.0

On behalf of the iMSTK community, we are pleased to announce the release of iMSTK version 6.0. The Interactive Medical Simulation Toolkit (iMSTK) is an open-source toolkit that allows faster prototyping of surgical simulators and skill trainers. iMSTK features advanced high-performance libraries for physics simulation, haptics, advanced rendering/visualization, user hardware interfacing, geometric processing, collision detection, contact modeling, and numerical solvers.

Here is a comprehensive list of changes made for this release.

Andrew Wilson

Enhancements

- ([2e7c3b1b](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/2e7c3b1b104a4673394fd81b8f2285b62b2ad85b)) REFAC: Connect & actions implemented in SWIG
- ([0f5a0f44](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/0f5a0f44e1c7f6d221bcd8913089a829b964c871)) ENH: Ghost tool for static suturing example
- ([70b922fb](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/70b922fb45f4199bb9c87824b751e8f260fb4432)) REFAC: Default to adaptive stepping in SimulationManager
- ([d5995889](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/d5995889629ddbed81c4a176a545f05625d15532)) ENH: Mouse controls for FemurCut example
- ([75d41d96](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/75d41d96afcbeef75681b01bacf51252aa4f4e84)) REFAC: PbdFunctors now created in the config with polymorphic interface
- ([2ab250d1](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/2ab250d172340f9f2eaca4404567727c60926803)) ENH: Cylinder mesh generation
- ([8f710ace](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/8f710ace59403a54b517f7a045b70b1e51438fdc)) ENH: Cloth grab example
- ([8790bb94](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/8790bb941ae8ee9b959a11c094dd93a2c2ff8956)) REFAC: TrackingDeviceController Rotational Inversions
- ([0ae58230](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/0ae582302f815d10ce0cb15b99178ce60cc11690)) ENH: Assimp reader can now read LineMesh
- ([d22d38f5](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/d22d38f588215bec24f3ee6ea5c2fe6dd12f9976)) ENH: Interactions now SceneObjects, CollisionGraph Removed
- ([8361c0ad](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/8361c0ad19949ea936c745805cae3cb82d00ad94)) REFAC: begin and end pick can be called from the interaction, getter for the node

- ([4b043ff7](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/4b043ff71143b0e89ff1c273d6938d09d26912e2)) REFAC: Change PbdPicking to occur after internal solve
- ([7deab9a9](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/7deab9a94ce3156cc964832bd6df5638c9020478)) ENH: Adds vsync toggle
- ([e89f1d3d](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/e89f1d3d0fc25864392e0cf6178d07cfc7c4aa81)) ENH: Upgrade to VTK 9.1
- ([e491885d](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/e491885d27618e06c039e714ad546f5db7afbbab)) ENH: Adds clearcoat model to iMSTK & colon fly through example rendered with clearcoat
- ([7c890690](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/7c890690f1fd14904ab0e3bbd31fbfac9d5ff74f)) ENH: OpenVR support for thumbsticks, triggers, & grips
- ([e964e167](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/e964e16710c3a983a8d65f6dbf909d05f504e29f)) REFAC: VR camera fixes, scene camera now pre multiplied to allow VR camera controls
- ([bb947658](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/bb947658ff3fe9eb4d585aa469060636fcca4203)) ENH: VR movement controls implemented
- ([2848b07f](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/2848b07fb4db9ef0aac1d19ecea0aa55db87cc94)) REFAC: Texture wrap type enum & border coloring
- ([ff2aedcf](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/ff2aedcf054ae76c47976872f5820737c7b03a2a)) ENH: Upgrade to TBB 2021.1.1
- ([5c6b0c68](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/5c6b0c681e0ae9eb4038cd2f257dc44ad89eb004)) REFAC: Split libraries off creating Materials, RenderingCore, RenderingVTK, ViewerCore, and ViewerVTK
- ([ad7e52cc](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/ad7e52cc07c7736668fd4068a0c527af22aff677)) ENH: Adds mouse controls for static suture example
- ([0a3e5e83](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/0a3e5e83cb7f2dbe055c56d2a673e0a5ad9cee33)) ENH: Adds critical damping to RigidObjectController
- ([a2dbaf81](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/a2dbaf813664343e9b06d74012cd9aabf5d9f05e)) REFAC: Adds dynamic flag in RenderMaterial avoiding GPU vertex buffer upload & CPU normal recomputation for rigid & static meshes
- ([8dcf84b8](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/8dcf84b8d902f2cf481288f19e860c1b0a87411a)) REFAC: Remove PhysX
- ([a99c1d9b](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/a99c1d9b831c4cc899963f8e9fae311f8ee41553)) ENH: Introduced delegate hints & RenderDelegateFactory
- ([f769b590](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/f769b590d9146bdae83b07e33e5eef501da78d73)) ENH: Adds ghost tool for virtual coupling example
- ([9c78db5c](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/9c78db5c2551a923b0814be502abc7e56dccbcd2)) REFAC: Remove parallel runtime for task graph
- ([865a629d](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/865a629dae0395e9cd35ec6b48e4d70f14f6bdf8)) REFAC: Add task node global ids
- ([b5ae8f71](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/b5ae8f7177a95f42b036aa0624dbdc6a13a6237e)) REFAC: Add default SceneObject name, unnamed
- ([47b66ea7](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/47b66ea7e315cd9a8c57c0c05e73a0a41cb4a614)) ENH: Refactor PbdPicking to remove collision handler & add element picking
- ([3f3e6110](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/3f3e61103b186f082f8c34f7cb9cd274d35a913c)) ENH: Generalized picking introduced

- ([6bb0ee1e](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/6bb0ee1e9c4003bd72d0be9ba7fce6bab1493ae1)) ENH: Update point picker for tetrahedral grasping
- ([4e4136f0](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/4e4136f05d005676e29fca1b3f7a065f5392044e)) REFAC: Make laprascopic angle movement relative to time
- ([ec64d6dc](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/ec64d6dcefc5ad60fbbaf55be7d5e0b96e8dd40e)) ENH: Ray cast point picker can sample first hit or N points
- ([ddceb571](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/ddceb571a16094c757ed0978d6056cf00e5456a3)) ENH: Support vertex data in cell grasper, such as with PointSetToSphereCD
- ([c70539d0](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/c70539d0b3bef6c6d7cfa95d5a90cbdec74e9edc)) ENH: OneToOneMap support for PbdObjectGrasping
- ([1d1dba48](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/1d1dba48b5817ab4062c112b2181f95dcc3bbb03)) ENH: Add PbdObjectStitching
- ([aff4c089](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/aff4c08958f9855363a7f47af16ef0cdb2f5596f)) REFAC: GeometryMapper to GeometryAlgorithm refactor
- ([f0c655a1](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/f0c655a19f8793dac8935556ef8255209814daf1)) ENH: SurfaceToTetraMap added
- ([e6d8ebd2](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/e6d8ebd23254eb6f27d83381d10693cae01f81dd)) ENH: Stitching support for thin tissues
- ([7dd6e8d6](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/7dd6e8d69053dfd4ef7681339b996dd0d8f47be2)) PERF: TriangleToTetMap now in O(n+m) instead of O(n*m)
- ([4bc001ae](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/4bc001ae697adc8c74617baf3fb3e8ad39a0d943)) REFAC: Fix dependencies. libusb, libnifalcon, & ftdxx not required when not building with VRPN
- ([c6634178](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/c66341789b43ff3d5b5f4205ff2a6de4a818276a)) REFAC: Remove globs from all imstk libraries
- ([2f14dd4b](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/2f14dd4bbcf9df5a3e0a44d07630f6c08340766a)) REFAC: Remove libnifalcon & ftd2xx
- ([e37e47c2](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/e37e47c2fba0f73eba3a8d060a7abf8e8b904f3a)) ENH: Adds build option to build without rendering
- ([1a27dfc4](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/1a27dfc44947423cb1ad8f6fa7c3e13229b7d8ce)) REFAC: Introduce VisualTesting base & reorganize test library linkage
- ([874395a4](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/874395a4fc38d4f764c6bc4e3979a9f9a3abe532)) REFAC: Add line, triangle, & tet grid geometry generation functions
- ([f1374ec5](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/f1374ec57dfb22824f046de17e76b43dea1b27c5)) REFAC: Default no friction for pbd collision
- ([41219474](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/412194744264322d9e53dac3c0280b7072481f1a)) ENH: SceneObject names optional, function to produce unique name
- ([9f998fdd](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/9f998fdda9ae412785dbf554198f76960baa9add)) ENH: Adds .vtk file writing for SurfaceMesh & LineMesh
- ([2fcd924e](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/2fcd924e4f0bc4ea316acfdcef0316e1d245710b)) ENH: Adds binary .msh reader, refactor ascii .msh reader
- ([3c59594b](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/3c59594b06e3a1c3febec2feec2fcc9838d4d710)) ENH: Capsule To Capsule Collision Detection and example

- ([dbd47504](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/dbd475044c9fd97dab3dc4d1d3c82c6420c3ffb7)) ENH: Default to 0 angle bend constraints, provide parameter in enableBendConstraints
- ([3e3551bf](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/3e3551bfb2b54fb1770c9819671d35528400165f)) ENH: Add haptic device support to PbdSutureExample

Testing

- ([97c44f84](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/97c44f847ca45f06c85ff1e0f7352743cab03acc)) TEST: Add test for initial camera looking down z
- ([46735548](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/46735548358974160f51fc6dccaf93277937e2e7)) TEST: RenderDelegateMock test
- ([dbd8779b](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/dbd8779bbaa22c6fdd92b65ebb728db1cf70a009)) TEST: Add point picker tests
- ([657c28a6](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/657c28a6464b6c2e38181cf0812ea1a85ff8a807)) TEST: Max distance boundary value test for PointPicker
- ([e03fa0b0](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/e03fa0b034ed24b701e7cab102b34eb32e0f4a35)) TEST: Barycentric and ray function tests
- ([ad96cf48](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/ad96cf48c5533c394b4543e08922ae2d31470593)) TEST: Integration testing for collision & pbd collisions
- ([39a01c2e](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/39a01c2e11d832e4726aa42c9843f1fe3d351c7d)) TEST: Add all the other collision test cases for Pbd
- ([99294cfa](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/99294cfa18c68897c2cf5b5255362eea486d14ae)) TEST: Add crevice test, tweak other PbdObjectCollisionTests
- ([478de476](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/478de4767d4f124372ab5455f03e0e82e742c96f)) TEST: addSceneObject tests
- ([4fb96509](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/4fb965090a582e7daadf1e865dc1a281e55c2773)) TEST: Add hexahedral mesh .msh IO test, tests validate contents
- ([481120f5](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/481120f57fd0b041fbca420917a066d686f95ef6)) TEST: Add PointSetToCylinderCD visual test
- ([8dd88704](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/8dd88704a7fe508f927f51e7ae647cdd71fcbf38)) TEST: RigidObjectCollisionTest visual testing

Bugs

- ([f2e7eb6a](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/f2e7eb6a99742a8b29cf8a326a374342aaf1adea)) BUG: Fix capsule sdf & bounding box func
- ([97b1d3ac](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/97b1d3ac0b87f1069fb84d4383069e30d125d392)) BUG: Basis in capsule getFunctionValue should use tranpose/inverse of rotation
- ([4ff061fd](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/4ff061fdffa785a332fef70c91f6fc34fd10e517)) BUG: Fix task graph cyclic issue, bad ordering in task graph PbdObjectCollision
- ([ddaaad0e](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/ddaaad0ed5babc2289aff3cc1fa59bc276cff77e)) BUG: Fix PbdObjectGrasping task ordering
- ([d2ef1f8e](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/d2ef1f8e982dcab0bb6321487ae909e343cb6f37)) BUG: Divide by zero case & empty else removed in constraints
- ([06ee4e5e](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/06ee4e5eade12f690465e518f3871a131861c1db)) BUG: Remove restriction on 0 capsule length

- ([dd981c74](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/dd981c74441512ccd5aa3f73c88c4c5f4f18bf88)) BUG: Fix textures when used with non PBR shading / flat, gourand, or phong
- ([4f772d35](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/4f772d353c4ed66c81cd62c1fa761d01c6e772ae)) BUG: Fix segment bc coords
- ([dd1aa416](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/dd1aa4166750738b200ead88b755cb6d969a921f)) BUG: Fix edge-edge closest point computation

Documentation

- ([0aae6ec6](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/0aae6ec64813c347d542355bb127cfb55cf42cc5)) DOC: Include dashboard link in readme, update PbdModel code example
- ([2cafd83b](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/2cafd83b23a8049997ebc2b46f818270de5e4b4f)) DOC: Add CD support matrix
- ([4804f952](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/4804f95204daaf6687ae94f4d0874a3aad943039)) DOC: Update SimulationManager documentation
- ([a37ef5e7](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/a37ef5e780614a2fbf2e87c3a964257dac5c5f75)) DOC: Update README, Contributing, & Maitnance docs
- ([6fd68759](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/6fd6875901c4f3ba5fee71db45bda441fd14c462)) DOC: Improve PbdModel documentation
- ([6a2d3355](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/6a2d335523a8be9c4be15f5889e28cfed195e15f)) DOC: Add documentation for CapsuleToCapsuleCD
- ([6508c824](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/6508c824511f2c16ce73ad60f4c753be46a9e276)) DOC: Add thread gif & description in PbdModel

Ben Boeckel

Enhancements

- ([1e262b0b](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/1e262b0b60c177e3fa5a86bf405c9dc9d0204890)) ENH: Add gitlab runner support and basic configuration for Windows

Harald Scheirich

Enhancements

- ([d9085dd2](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/d9085dd2f5e42c3a426d68cb146baa584a77a1ee)) ENH: Update Eigen to 3.4
- ([d2419bff](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/d2419bffb3ca6690def056d03f2375b4fbd14fbe)) ENH: Update VRPN to latest version
- ([2c89fa1e](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/2c89fa1ea51802a80b494c7e464323066fd36029)) ENH: Add adding of multiple nodes and edges to taskgraph
- ([a543c7d6](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/a543c7d61e284429f80161d3a0dfbfdc4175b07f)) ENH: Update C++ to 14
- ([3c8e7957](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/3c8e7957277161efcb03475915614799236bc6e0)) REFAC: Consistently use delegate hint for render delegate creation
- ([91bf9003](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/91bf900340005f51fd4f6064280d80f460d8bfcc)) ENH: Add Analog getter in DeviceClient
- ([bc047b8d](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/bc047b8d15d39bd1fde75a3e47fef5b3ebfb292c)) ENH: Split rendertests into separate CMAKE variable

- ([f2491052](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/f2491052206bdc331982dd186eba296a59d84edf)) ENH: Introduce TYPE_NAME macro
- ([b961cbcd](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/b961cbcd481345a5981768f078555f39f024a9dc)) ENH: Common Factory infrastructure
- ([805894b1](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/805894b1158012c4a37fd02ce461ff7d5d055632)) ENH: Add simple example to use for onboarding work
- ([62a03fde](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/62a03fdeb4f1a5d688b9fcf8d90ff21cae4f9a12)) ENH: Add const begin/end to data array
- ([cb0ecef5](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/cb0ecef546323c2bfdf4ab0054b139abc83af659)) ENH: Add second logger to enable unity to catch issue thrown by failed CHECK or FATAL messages
- ([9ac5d1a5](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/9ac5d1a5185a08c2b571a4039638db4b51b1c5a3)) ENH: Enable C# and Unity to catch exceptions posted by C++ code
- ([1f721fa8](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/1f721fa8bf3f5f291af2743fd5a3a5f16c9a7051)) ENH: Unity loggers log to file as well
- ([8b1cd2a3](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/8b1cd2a306a904b3a8c1c1bb22dfe28e7b4180f9)) ENH: Avoid extra DataArray allocations on assignment
- ([101bc8ee](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/101bc8eed6efb86a1b19fc2d23acb3bf0affd765)) ENH: Additional tests for testPlaneToSphere
- ([65705e50](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/65705e503c4b998f79d6254e832b9489c0432d85)) ENH: Create Confiuration file for C#
- ([c1b00ec8](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/c1b00ec8939e2666a55b75493525863517fe023c)) ENH: Add Dependent options with regards to Unity build

Testing

- ([dc9f3bf8](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/dc9f3bf84b9b0e7a823633a2a86c6d75961a6312)) TEST: Add taskgraph unit tests
- ([115f9a28](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/115f9a287ac4e9fdcb96070e7ed7bcb79a710731)) TEST: Add tests for OneToOneMap

Bugs

- ([b2da6894](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/b2da689422ead8cbadaa2e55536030368d75c862)) BUG: Fix treatment of unsafe functions when PINNED_ARRAY is on
- ([91aeb8d9](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/91aeb8d96e56662b04fe9b7482a0821ef0c6c0eb)) BUG: Correct handling of size and mapped ptrs in new copy
- ([fdb51195](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/fdb5119511cb0cb0421b6bac72f07924924ea6f3)) BUG: fix spurious failure when closing a window in visual tests on linux

Documentation

- ([d5f54e4b](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/d5f54e4b4216bcd3303ef96f4c377b5198e58805)) DOC: Add language to coding guidelines addressing published work
- ([bfee2fba](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/bfee2fba8e24af076f2679bef5438655c27c838e)) DOC: Update Coding guide to bring in better in line with current iMSTK practices
- ([47c368d1](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/47c368d1f78d29b8c136a4f9da1cc0b5b04809be)) DOC: improve documentation for factories
- ([f5a1b2f8](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/f5a1b2f83ae8ac78ef867a5df37350d2c73a782c)) DOC: Add documentation for coverage calc

- ([f397bccd](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/f397bccd1b6d8771a5ee885e7b73e923c8711c79)) DOC: Update README

Hong Li

Enhancements

- ([ff9f2cb8](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/ff9f2cb8280691bee53fb76f77324b09ef8a9c63)) ENH: Add support for texture coordinates runtime modification for SurfaceMesh
- ([98f4ddc2](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/98f4ddc29b0be7cb065daad2c667428a53b2bf47)) ENH: Add PointSet to Cylinder Collision Detection

Bugs

- ([2ace82a9](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/2ace82a9db04059d97baadec6cf17efe86402846)) BUG: Fix Plane to Sphere collision test bug

Jacob Moore

Enhancements

- ([a8676f24](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/a8676f249de6c46eebb9f560d0aae8a5107aff1c)) ENH: Pbd vs Rigid Body Haptic Grapsing
- ([28b32fad](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/28b32fadbd0a05498a5cb5337b378421d32d2271)) ENH: Adding suture constraints for penetration
- ([ac24fd7d](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/ac24fd7dc22021acdcc3291b0d315e9fd7b958ff)) ENH: Adding suture needle interaction
- ([16fb452b](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/16fb452b910694c40490a580d8876ac547ac4a75)) ENH: Adding suture needle object

Testing

- ([235ef0a9](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/235ef0a94d337ddf20e86f9ffd2d8b34161b2092)) TEST: Testing for Capsule class
- ([a171879d](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/a171879d0575e4508be0401cf7ea78c07bab0b40)) TEST: Testing for SurfaceMeshToCapsuleCD and Capsule class
- ([e693456e](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/e693456e2129d056efe2e729166972ee1376fc17)) TEST: Simplified volume test for capsule
- ([57d80e09](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/57d80e09019a0e28f58b63bc13aa32bb3050b8ea)) TEST: Adding benchmarking to PBD method
- ([c998964f](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/c998964f02fddea0089030551ef8197ca0130d9f)) TEST: Added more benchmark cases with contact and modifed based off feedback

Jean-Christophe Fillion-Robin

Enhancements

- ([e9cc8117](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/e9cc8117aec192b56e5249f7bb72c4207d3a2852)) COMP: Update build-system to be consistent and require VTK dependency
- ([051abb5d](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/051abb5d1086cc48d1cc7136396ccef428bdb2c2)) COMP: Update imstk_find_header to keep ${package}_INCLUDE_DIR in the cache
- ([086e9b05](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/086e9b0536abd721c9212cb8e77308b0d8c4494b)) COMP: Simpler config against project in given dir setting default for <package>_LIB_DIR
- ([ebfde657](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/ebfde657cd32ba490add295cb33e472b18f0827c)) COMP: Support for custom include subdirectory associated with imstk_find_header

- ([35940011](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/359400116536b2c62f7817e8769d3335fe4a7422)) ENH: Update imstkSolveDependencies module based on commontk/Artichoke@4176c39f9
- ([885c7cdf](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/885c7cdf986bd7b5a2335de02b36660b14729718)) COMP: Update imstkAddExternalProject to support specifying custom install dir
- ([fe4db2e1](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/fe4db2e1466dd07ef7ad57a81f9bf24b46b5b73c)) COMP: Add relevant vars to iMSTKConfig to streamline config against a build-tree
- ([01373ce8](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/01373ce80f171ab6ea7a134ce7b79de84ae78d2d)) COMP: Update Libusb external project to support Linux
- ([5e3a7f46](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/5e3a7f463b24dcc9c18307ad6beb1c2c87a70966)) COMP: Support setting custom SOURCE_DIR and BINARY_DIR vars
- ([a30d9fa8](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/a30d9fa8654e98244c70ae073f50abe27de69c11)) COMP: Update imstk_add_external_project to keep track of the project var
- ([a9203409](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/a920340960406c8dcc30c4c9785a121a11af2af4)) COMP: Streamline client project integration updating TBB from 2019_U2 to 2019_U9
- ([abadb943](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/abadb9438e4d7939c4deaed122cf0aefc7288159)) COMP: Update external project supporting setting a custom git repository
- ([17d0c63d](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/17d0c63d6923103c2c57555cd48024075eaa0271)) COMP: Update imstkSolveDependencies module based on commontk/Artichoke@edfc828)
- ([663b4daf](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/663b4daf081ddd937c68cdb68c6b6c6b96b1876e)) COMP: Ensure inner build is always re-configured
- ([f8a1edb0](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/f8a1edb0c97d8b4d9e022d6de4fd22bc8a46ff87)) COMP: Simplify and fix handling of PHYSX_CONFIGURATION option
- ([b21384c2](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/b21384c28e6d5e720fe62b76b996fdd712761a62)) COMP: Simplify and fix handling of iMSTK_USE_OpenHaptics option
- ([ff4d8c3d](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/ff4d8c3d4823b19d1691e590d224e82644cce7cd)) COMP: Update inner build external project to use CMAKE_CACHE_ARGS
- ([4600b6c9](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/4600b6c9ecf1b57734a14c6dc215830f312081fe)) COMP: Simplify and fix passing of options to inner build using mark_as_superbuild
- ([c55b3ed1](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/c55b3ed11e1f1cb40f1dcb64e3b6e118e6698bd1)) COMP: Remove obsolete inner build external project patch step for VTK 8.2
- ([48004598](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/4800459833fbc88649eaaa8a3653d6ec0845aa9e)) COMP: Update Assimp to simplify packaging in client projects

Shreeraj Jadhav

Enhancements

- ([c30e56e5](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/c30e56e52e23aa0d53645f1313511c2bcf32f9bd)) ENH: Create a CCD Algorithm class
- ([406c734f](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/406c734fae88a4008a8e4547ae804fb5156bd0d7)) ENH: Add CCD algorithm for LineMeshtoLineMesh collision
- ([0a69049d](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/0a69049d8bca71f497ee099b3d0a1bce3d0306c2)) ENH: Add new contraint for CCD Edge-Edge collision
- ([f5de4123](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/f5de4123d4d17bcac909bc0cd947d812443ebd32)) ENH: Collision handling for Edge-Edge CCD

- ([78d0dc32](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/78d0dc32742cbb5ce607915c50aadbc127ff1958)) ENH: Add example for LineMesh self CCD

Testing

- ([d01f18cc](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/d01f18ccea46695784871a48285b3bfd3e5050d7)) TEST: Add testing for LineMesh CCD

Documentation

- ([f1c1dced](https://gitlab.kitware.com/iMSTK/iMSTK/-/commit/f1c1dced126cf2dae755d9000fd5813374857247)) DOC: Add description for LineMeshToLineMeshCCD

Api Changes

- *PbdModel::getParameters* → *PbdModel::getConfig*
- ***PbdModelConfig::enableConstraint* now takes *PbdModelConfig::ConstraintGenType* uncoupling constraint generation scher**
  - ex: *enableConstraint(PbdConstraint::Type::Distance, 1.0e2);* -> *enableConstraint(PbdModelConfig::ConstraintGenType::Distance, 1.0e2);*
- *SimulationManager* library split into *SimulationManager*, *ViewerCore*, & *ViewerVTK*. Linkage to *ViewerVTK* may be required in dependent projects should you require that library.
- Rendering library split into *RenderingCore* & *RenderingVTK*.
- ***GeometryMap`s, now extend `GeometryAlgorithm***
  - *OneToOneMap* → *PointwiseMap*
  - *TetraToTriangleMap* → *TetraToPointSetMap*
  - *OneToOneMap::getIdx* → *OneToOneMap::getParentId*
  - *GeometryMap::apply* → *GeometryMap::update*
  - Geometry constructor inputs → *GeometryMap::setParentGeometry* & *GeometryMap::setChildGeometry*
  - GeometryMap::Type enum removed.
- Many acronym'd names now Upper case first letter only. ie: *PBD* → *Pbd*. *FEM* → *Fem*. Previously had mixed usage.
- *PbdPickingCH* removed/refactored into *PbdObjectGrasping*.
- All barycentric functions moved to *imstkMath.h*
- *DataLogger* removed
- *MeshToMeshBruteForceCD* renamed to *ClosedSurfaceMeshToMeshCD*
- All VTKOpenVR classes now VTKVR. ex: *VTKOpenVRViewer* → *VTKVRViewer*. Backend is unspecified in name. OpenXR by default.
- *IdentityMap* removed.
- Collision detection algorithm factory refactored. Get collision detection class by name with *CDObjectFactory::makeCollisionDetection("SurfaceMeshToCapsuleCD");*
- *VisualModel* constructor input removed. Use *VisualModel::setGeometry* instead.
- *CollisionPair`s & `CollisionGraph* removed. Use *SceneObject*'s added to the scene with *Scene::addInteraction*
- A few occurrences of *getTranslation* & *getRotation* changed to *getPosition* & *getOrientation* when referring to a pose, not a transformation.

- PhysX deprecated backend removed completely.
- Only utilized VTK libraries are linked to, not every built VTK library. Any user facing VTK code may need to link to required libraries.
- Update to C++14
- Update to VTK 9.1
- Update to TBB 2021.1.1
- Update Eigen to 3.4
- Update VRPN

Contributors: Andrew Wilson, Harald Scheirich, Shreeraj Jadhav, Jacob Moore, Jean-Christophe Fillion-Robin, Hong Li, Ben Boeckel

5.2 Release 5.0.0

Announcement: iMSTK 5.0.0

This release features major improvements to the collision detection and response capabilities, rendering, and hardware interface modules. Numerous refactors and bug fixes were made across the board to improve extensibility. The testing infrastructure and coverage was improved substantially. Numerous medically relevant examples were added. Most notably, v5.0 comes with a beta version of the C and C# wrappers for iMSTK along with documentation and examples.

Here is a comprehensive list of changes made for this release.

Andrew Wilson

Enhancements

- Extended PBD model to allow needle-tissue interactions (07819c60)
  - PBDTissueSurfaceNeedleContact: Simple approach to needle insertion of deformable 2d triangle mesh tissue with rigid body haptics.
  - RbdSDFNeedle: Simple approach to needle insertion of static tissue for which a rigid body needle may rotate/pivot before being fully inserted.
  - PBDTissueVolumeNeedleContact: More involved approach to needle insertion of deformable tetrahedral mesh tissues with rigid body haptics. Uses two-way embedded coupling.
  - PBDStaticSuture: Suture thread and needle vs static geometry. Uses custom arc to point constraint. Suture thread non functional yet. Only surface constrained.
- The collision detection architecture was comprehensively refactored and updated (ed212e8f)
  - PbdObject vs CollidingObject possible: PbdObject vs Rigid, SPH, FEM, any CollidingObject are now possible.
  - Pbd vs Primitives Collision: PbdCollisionConstraint's and PbdCollisionHandler now use pointers and values instead of DataArray's allowing collision between something that doesn't have a vertex buffer.
  - PbdPointToEdgeConstraint: Allows mesh to curved surface collision when used together with point-to-point and point-to-triangle.
  - MeshToMeshBruteForceCD New Implementation: Able to resolve deep contacts in manifold shapes. Stable and accurate method, not performant for large meshes.
  - Virtual constraint addition for Pbd and Rbd models: Allows redefinition of contact response models. Used for needles, drills, thread holes, cutting, etc.

    - PbdRigidCollision interaction: Two-way collision between RigidObject2 and PbdObject. Gives response between both objects. (useful for haptics, no reaction force on the RigidObject2 is felt without it). See PBDTissueContactExample.
    - CollisionDebugObject: Can be added to the scene to display CollisionData as faces, edges, points, or directions/arrows.
    - SurfaceMeshToSphereCD: Triangle mesh to sphere collision. Demonstrated in PbdClothCollisionExample & RbdSurfaceMeshToSphereCDExample.
    - Restitution and friction added to PbdModel
- Light properties can be changed at runtime, attenuation exposed (2d9a78c1)
- Textures can now be setup via ImageData instead of filenames, swapped at runtime, & pixels modified at runtime (6e79fb81)
- VisualModels can be added/remove from the scene at runtime. (f1e55106)
- Camera now provides projection matrix and eye rays (bdbffbb8)
- Added OrientedBox to geometry (replaces Cube) (845dd6c5)
- PbdConstraintFunctor was added that allows customization of constraints generated for a PbdModel. (e3a9753f)
- Added strides to PBD bending model. Bend constraints can now be generated for differing strides allowing stiffer threads/strings at lower iterations. (350adc5f)
- Added RenderMaterial::ShadingModel::None allowing shading to be turned off where necessary (e7811080)
- Added RenderMaterial::DisplayMode::Normals that displays normal directions using arrows (614e296e)
- Added Texture Projection filter that projects attributes (eg: uv coords) via closest point projection (17fb7679)
- Cell attributes were added to LineMesh (08b08212)
- LineMeshes can be read in via VTK file type (6e20e88b)
- SceneObject::visualUpdate virtual function is now called before every render for every SceneObject in the scene (fa95ff5f)
- Added disposable syringe, orthopedic drill, c6 needle 3d models to the imstk data
- Performance improvements to SPH (6a2bb089)
- Performance improvements to Dense LSM (d2a2b88f)
- Remove nonfunctional Pulse (3b41b3e1)
- Remove nonfunctional Vulkan (5e7eddbc)

Documentation Updates

- PbdModel documentation updated (6eb4a04a)
- CollisionDetection documentation updated (2d9a78c1)

Testing

- PbdBendConstraintFunctor testing (dc431620)
- CollisionDetection testing (07819c60), 3b19c782, 91d09f63)
- PbdCollisionConstraint testing (ed212e8f)
- PGSSolver testing (567f126e)

Harald Scheirich

Enhancements

- VRPN Analog, Button and Tracker devices. No limit to VRPN in iMSTK anymore (co-authored by Khalil Oumimoun) (0f02a666)
- Improved testing capabilities. Reduced overhead in testing infrastructure removing auto generated classes. (3059ffbe)
- Added Git LFS-based external data management for data required by tests and examples. (7eb3d176)
- Update gtest (a4d30ec5)

Documentation Updates

- Improve C# Wrapper Documentation (28050bd3)
- Added documentation for external Data and dependency update process (a4051249)
- Added coding style guidelines (1f450dcf)

Testing

- DataArray testing (99f18c24, 8b08c873)
- Device & Control testing (1b76fed9)
- EventObject testing (538915c8)
- GeometryAlgorithm testing (d061764a)
- SurfaceMesh tests (e7dfcb23)
- Render testing (57725ee0)

Sreekanth Arikatla

Enhancements

- Remove plotter utils (5508d197)
- Remove APIUtils (554cde05)
- Cleanup Vulkan references and remove related external dependencies imgui, glm (d6cea42b)
- Scene bounding box computation (458fa955)

Testing

- PbdConstraintFunctor testing (a55301f1)
- Geometry testing (613402e1, 84caab04)
- CollisionDetection testing (07819c60, 3a135f83, 6b5a3685, 2e6ad2aa, 266ea1b3, 64363cd1, 2f9e385f, 666467bd)
- Scene testing (cfdbe9e3)

Infrastructure

- Enabled nightly readTheDocs build.

Jianfeng Yen

Enhancements

- SWIG-based generation of C and C# wrappers along with C# examples and tests (See Source/Wrappers/csharp) (c11d712c)

- Performance Improvements to FEM (c9f002ec)

Ye Han

Enhancements

- Triangular mesh cutting via local remeshing (e7c01ead)
- Added SurfaceMeshToCapsuleCD static collision method (42a321fb)

Khalil Oumimoun

Enhancements

- VRPN Analog, Button and Tracker devices (0f02a666)

Aron Bray

Testing

- Integration tests for rendering and geometry modules (96f084a1)

Hong Li

Enhancements

- Extension of PBD Constraints to inflate tissue (3bf08161)

Furkan Dinc

Enhancements

- Screen space ambient occlusion support (ab38797d)

Ben Boeckel

Infrastructure

- Added Linux to the merge request builds
- Fixed issue on the MSVC 2017 merge request build

Api Changes

- *PbdPointDirectionConstraint* replaced with *PbdPointToPointConstraint*.
- *PbdCollisionConstraint::initConstraint* now use *VertexMassPair* structs for initialization.

```
PbdPointPointConstraint constraint;
constraint.initConstraint(
    { vertexA, invMassA, vertexVelocityA },
    { vertexB, invMassB, vertexVelocityB },
    stiffnessA, stiffnessB);

- `PbdCollisionConstraint::projectConstraint` updated, accepts no parameters anymore as
↪it uses pointer values provided during `PbdCollisionConstraint::initConstraint`.
- `PbdCollisionSolver::addConstraint` now only accepts a constraint, no buffers.
```

```
std::vector<PbdCollisionConstraint*>* constraints;
// ...Fill out constraints...
myPbdCollisionSolver->addCollisionConstraints(constraints);

- `CollisionDetection` replaced with `CollisionDetectionAlgorithm`. Now subclasses
↪`GeometryAlgorithm` with inputs given via `GeometryAlgorithm::setInput`.
```

```
 // Either order is ok (mesh, sphere) or (sphere, mesh)
 imstkNew<SurfaceMeshToSphereCD> collisionDetect;
 collisionDetect->setInput(mySurfMesh, 0);
 collisionDetect->setInput(mySphere, 1);
 collisionDetect->update();

 // Output order dependent on input order
 collisionDetect->getCollisionData()->elementsA; // CD elements for the mesh
 collisionDetect->getCollisionData()->elementsB; // CD elements for the sphere

- Collision interactions renamed, more consistent naming, removal of "Pair".
  - `PbdObjectCollisionPair` renamed to `PbdObjectCollision`.
  - `SphObjectCollisionPair` renamed to `SphObjectCollision`.
  - `RigidObjectLevelSetCollisionPair` renamed to `RigidObjectLevelSetCollision`.
  - `RigidObjectCollisionPair` renamed to `RigidObjectCollision`.
- `PbdObjectCollision` can now be constructed with both PbdObject+PbdObject or␣
↪PbdObject+CollidingObject
```

```
imstkNew<PbdObjectCollision> interaction1(myPbdObjectA, myPbdObjectB);
imstkNew<PbdObjectCollision> interaction(myPbdObjectA, myCollidingObjectB);

- `RigidObjectCollision` can now be constructed with both RigidObject2+RigidObject2 or␣
↪RigidObject2+CollidingObject
```

```
 imstkNew<RigidObjectCollision> interaction1(myRbdObjectA, myRbdObjectB);
 imstkNew<RigidObjectCollision> interaction(myRbdObjectA, myCollidingObjectB);

- `OrientedBox` should be used in place of `Cube`.
```

```
 Vec3d center = Vec3d(0.0, 0.0, 0.0);
 Vec3d extents = Vec3d(0.5, 0.5, 0.5); // Does not have to be a cube
 imstkNew<OrientedBox> box(center, extents);

- `RenderMaterial` now uses doubles, not floats.
- Removed Octree collision in `SurfaceMeshToSurfaceMeshCD`. The octree is still present␣
↪but pending touch ups.
- `RigidBodyModel2` and corresponding classes moved from expiremental to main imstk␣
↪namespace.
- `DebugRenderGeometry` replaced with `DebugGeometryObject`. Usage as follows:
```

```
 imstkNew<DebugGeometryObject> debugGeometryObj;
 scene->addSceneObject(debugGeometryObj);

 // Can be called anytime, normally during runtime
 debugTriangles->appendVertex(Vec3d(0.0, 1.0, 5.0), Color::Red);
 debugTriangles->appendLine(p1, p2, Color::Green);
 debugTriangles->appendArrow(p1, p2, Color::Orange);
 debugTriangles->appendTriangle(p1, p2, p3, Color::Blue);

- `DebugRenderDelegate`'s replaced with the already existing `RenderDelegate`'s.
- Removed Vulkan, Pulse, apiUtilities, imgui
- `NarrowPhaseCD` namespace functions moved and refactored into more general static␣
↪intersection functions in `CollisionUtils`.
```

(continues on next page)

(continued from previous page)

```
- Name was removed from the light class, light names are managed by the Scene
```

```
 imstkNew<DirectionalLight> light;
 light1->setFocalPoint(Vec3d(-1.0, -1.0, -1.0));
 light1->setIntensity(1.0);
 scene->addLight("light", light);

- Removed `Real`'s, use `double`'s instead.
- `RigidBodyModel` now optionally built, being deprecated in place of `RigidBodyModel2`␣
↪as it's more extensible.
- `PBDModelConfig::m_defaultDt` removed, use `PBDModelConfig::m_dt` instead.
- `PBDModelConfig::m_collisionIterations` removed. Iterations can be set per collision␣
↪solver, given per interaction.
```

```
 auto pbdHandler = std::dynamic_pointer_cast<PBDCollisionHandling >(interaction->
↪getHandlerA());
 pbdHandler->getCollisionSolver().setCollisionIterations(5);

- `PbdModelConfig::enableBendConstraint` should be preferred when using bend constraints␣
↪with varying strides. If `PbdModelConfig::enableConstraint(PbdConstraint::Type::Bend)`␣
↪is used, stride will always be 1.
- `PbdModel::initializeConstraints` functions removed. Replaced with extensible␣
↪`PbdConstraintFunctor`.
- `Texture` may also be constructed with an `ImageData`
```

```
 std::shared_ptr<ImageData> diffuseImage = MeshIO::read<ImageData>(iMSTK_DATA_ROOT "/
↪textures/fleshDiffuse.jpg");
 material->addTexture(std::make_shared<Texture>(diffuseImage, Texture::Type::Diffuse));

- Enums removed in places were extensibility desired.
  - `Geometry::Type` removed. Use `Geometry::getTypeName()` instead.
  - `CollisionDetection::Type` removed. Use `CollisionDetectionAlgorithm::getTypeName()`␣
↪instead.
  - `CollisionHandling::Type` removed. Use `CollisionHandling::getTypeName()` instead.
```

Contributors

Andrew Wilson Harald Scheirich Sreekanth Arikatla Jianfeng Yen Ye Han Khalil Oumimoun Aron Bray Hong Li Furkan Dinc Ben Boeckel Andinet Enquobahrie

5.3 Release 4.0.0

Announcement: iMSTK 4.0.0

This release features major improvements to the simulation execution pipeline, geometry module, virtual reality support, and haptics. Extensive refactoring and numerous bug fixes were also made across the codebase to allow extensibility of certain classes, improve clarity, separate roles for different iMSTK libraries, and enforce consistent design and conventions across the toolkit.

Here is a comprehensive list of changes made for this release.

New Features

- Addition of Implicit Geometry

  - SignedDistanceField
  - CompositeImplicitGeometry
  - All existing analytical geometries
- Addition of LevelSetModel (regular grids only)
- Addition of ImplicitGeometryCD & ImplicitGeometryCCD
- Addition of RigidBodyModel2
- Addition of Event system (addition of EventObject)
- Addition Imstk Data Arrays. DataArray+VecDataArray. Replacement throughout code.
- Addition of per cell and vertex mesh attributes. DynamicalModels use them.
- Addition of PBDPickingCH, for picking PBD vertices.
- New Geometry Filters. Including LocalMarchingCubes.
- Offscreen rendering support through VTK OSMesa
- New substepping and sequential execution mode for SimulationManager

Improvements or Refactoring

- SimulationManager and Module Refactor
- Refactor VTKRenderDelegates
- Topology changes supported
- New examples. Many fixed haptics and OpenVR examples.
  - FemurCut
  - PBDPicking
  - SDFHaptics
  - FastMarch
  - RigidBodyDynamics2
  - PBDCloth-Remap
  - SPH-Obj-SDFInteraction
  - Vessel
- imstkNew
- Refactor geometry base class transforms
- OpenVR, Keyboard, and Mouse Device Refactoring
- Control refactoring
- virtual update and visualUpdate functions for SceneObject
- virtual init and advance functions for Scene
- VRPN build toggle
- Geometry enums replaced with getTypeName polymorphic function
- DynamicalModel enums replaced with getTypeName polymorphic function
- Module unit tests

- VecDataArray + DataArray unit tests
- NRRD, MHD, & NII image file support through VTK
- Debug camera initializes to bounding box of initial scene
- Bounding box computation of many primitives added
- Laprascopic Tool Controller fixed + improved
- VisualObjectImporter can now read and flatten scene hierarchies.
- PBD performance improvements
- HapticDeviceClient accepts no name for default device
- ColorFunctions added to RenderMaterial for mapping scalars
- imstkCamera refactor, view matrix can now be set independently of focal point and position.
- VTKViewer split into VTKViewer and VTKOpenVRViewer, common base VTKAbstractViewer added.
- Mute, log, or display VTK logger options added to VTKAbstractViewer
- Shared RenderMaterials

Bug Fixes

- Capsule CD fixes
- OpenVR fixes
- Missing bounding box functions for some analytical shapes added
- Rigid body reset fixes
- Many virtual destructors added

API Changes

- OpenVR, Keyboard, and Mouse device refactoring: Mouse and Keyboard now provided under the same DeviceClient API as our haptic devices. You may acquire these from the viewer. They emit events, you can also just ask them about their state.

```
std::shared_ptr<KeyboardDeviceClient> keyboardDevice = viewer->getKeyboardDevice();
std::shared_ptr<MouseDeviceClient> mouseDevice = viewer->getMouseDevice();

std::shared_ptr<OpenVRDeviceClient> leftVRController = vrViewer->getVRDevice
```

- Controls: Our controls are now abstracted. Any control simply implements a device. You may subclass KeyboardControl or MouseControl. We also provide our own default controls:

```
// Add mouse and keyboard controls to the viewer
imstkNew<MouseSceneControl> mouseControl(viewer->getMouseDevice());
mouseControl->setSceneManager(sceneManager);
viewer->addControl(mouseControl);

imstkNew<KeyboardSceneControl> keyControl(viewer->getKeyboardDevice());
keyControl->setSceneManager(sceneManager);
keyControl->setModuleDriver(driver);
viewer->addControl(keyControl);
```

- Event System: Key, mouse, haptic, and openvr device event callback can be done like this now.

  - You may alternatively use queueConnect as long as you consume it somewhere (sceneManager consumes all events given to it).
  - Your own custom events may be defined in iMSTK subclasses with the SIGNAL macro. See [KeyboardDeviceClient](https://gitlab.kitware.com/iMSTK/iMSTK/-/blob/master/Source/Devices/imstkKeyboardDeviceClient.h) as an example.

```
connect<KeyEvent>(viewer->getKeyboardDevice(), &KeyboardDeviceClient::keyPress,
  sceneManager, [&](KeyEvent* e)
  {
    std::cout << e->m_key << " was pressed" << std::endl;
  });
```

- Imstk Data Arrays: Data arrays and multi-component data arrays provided. They are still compatible with Eigen vector math.

```
VecDataArray<double, 3> myVertices(3);
myVertices[0] = Vec3d(0.0, 1.0, 0.0);
myVertices[1] = Vec3d(0.0, 1.0, 1.0);
myVertices[2] = myVertices[0] + myVertices[1];

std::cout << myVertices[2] << std::endl;
```

- SimulationManager may now be setup and launched as follows:

```
// Setup a Viewer to render the scene
imstkNew<VTKViewer> viewer("Viewer");
viewer->setActiveScene(scene);

// Setup a SceneManager to advance the scene
imstkNew<SceneManager> sceneManager("Scene Manager");
sceneManager->setActiveScene(scene);
sceneManager->pause(); // Start simulation paused

imstkNew<SimulationManager> driver;
driver->addModule(viewer);
driver->addModule(sceneManager);
driver->start();
```

- *VisualObject* typedef removed. Just use *SceneObject*.
- *HDAPIDeviceServer* renamed to *HapticDeviceManager*
- *HDAPIDeviceClient* renamed to *HapticDeviceClient*

Contributors

Andrew Wilson, Venkata Sreekanth Arikatla, Ye Han, Harald Scheirich, Bradley Feiger, Jianfeng Yan, Johan Andruejol, Sankhesh Jhaveri

5.4 Release 3.0.0

Announcement: iMSTK 3.0.0

This release features major improvements to the computational workflow, physics, and rendering aspects of the toolkit. Major refactoring and bug fixes were made across the board to allow easy extension of classes, improve clarity and separation of roles of different imstk libraries and enforce consistency of design across the toolkit.

Here is a comprehensive list of changes made for this release.

New Features

- Introduction of configurable task-graph and task-based parallelism.
- Major upgrade to the rendering module (VTK backend)
  - Upgrade to VTK 9.0
  - Realistic fluid rendering using screen space fluids
  - Faster particular rendering of fluids
  - Addition of physically based rendering
- Addition of 3D image support and volume rendering
- Improved physics models for particle based dynamics: Addition of extended position based dynamics (xPBD)
- Addition of support for modeling 1D elastic structures with bending stiffness
- Addition of faster reduced order deformation models (Linux only)
- Addition of Reverse Cuthill–McKee algorithm (RCM) for mesh renumbering
- Major refactoring simulation manager: Improved time stepping policies, multiple scene management and scene controls, addition of async simulation mode
- Improved capabilities of the geometric utility module: addition of geometric processing filters, New tetrahedral mesh cover generation (based on ray-casting)

Improvements or Refactoring

- Upgrade external dependency from Vega 2.0 to 4.0 (finite element library backend)
- Clear majority of the warnings in imstk libraries
- Refactored examples: consistent naming, factoring out object addition into separate functions, use heart dataset, remove redundant mapping, Removed line mesh example
- New examples for scene management, volume rendering, task graph
- Renamed files to be consistent with class names
- Vulkan shader project removed for VTK backend
- Remove imstkVolumetricMesh dependency on vega volumetric mesh
- Easy configuration of finite element deformable object, viewer, renderer and simulation manager
- Concrete dynamcal models now derive from AbstractDynamicalModel
- Solvers are moved to models from scene
- Added default solvers for models
- SPHSolver is removed
- SceneObject class now has update calls

- DynamicalObject de-templatized
- Fix render window default title to imstk
- Replace external project download links with .zip versions
- Uses CHECK() instead of LOF(FATAL)/LOG_IF(FATAL) for simplicity
- imstkLogger is now a singleton
- Allow exclusion of files while building library targets
- Refactoring to use forward declarations where possible
- Templated solvers with matrix type
- Faster TetraToTriangle map
- Interactions are now specified explicitly
- PbdConstraints moved to Constraints library, PbdConstraints and PbdModel decoupled
- PbdModel performance improvements
- SPHModel performance improvements (using TaskGraph)

Bug Fixes

- Fix PhysX backend build issues on Ubuntu
- Fix imstkFind.cmake issues
- Fix imstkConfig.cmake issues
- PbdModel reset fix
- All Scene, SceneObjects reset correctly now

API Changes

- simulationManager::startSimulation() to simulationManager::start()
- CollisionGraph::addInteraction(std::shared_ptr<CollidingObject>, std::shared_ptr<CollidingObject>, CollisionDetection::Type, CollisionHandling::Type, CollisionHandling::Type) to CollisionGraph::addInteraction(std::shared_ptr<SceneObjectInteraction>())
- DynamicalModels now have default solvers

Contributors Venkata Sreekanth Arikatla, Andrew Wilson, Jianfeng Yan, Aaron Bray, Sankhesh Jhaveri, Johan Andruejol

5.5 Release 2.0.0

Announcement: iMSTK 2.0.0

This release adds major features for the physics and rendering modules. Parallel support is also added. Major improvements to the CMake build and install steps have been implemented. Many modules have been refactored for clarity and to reduce reducdency.

For more information, visit our website: http://www.imstk.org/

New Features

- Rigid body dynamics with Physx backend
- Debug rendering support
- Octree-based collision detection

- Multithreading support (using Intel TBB)
- Smoothed Particle Dynamics for fluids
- Customizable on-screen text
- New simulation modes for simulation manager to allow flexibility
- VR support for Vulkan backend
- Particle systems for visual effects
- Lens distortion for use in VR (Vulkan backend)
- Vulkan renderer compressed texture support

Improvements or Refactoring

- Improved CMake build and install
- Enable compiler flags to report W4-level warnings
- Remove cyclic dependencies between modules
- Add color to stdout on windows
- Refactored Position based dynamics classes
- Refactor rendering specification using visual model
- Modifications to the code formatting rules
- Refactor geometry mapping classes
- Remove unused files and classes
- Disable building tests for external dependencies
- Update the vrpn to the latest to fix linux build
- Update VTK backend to 8.2.0
- Remove ODE external library

Bug Fixes

- Fix undefined behaviour of PBDModelConfig
- Use vtkPolyData instead of vtkPolyLine for VTKdbgLinesRenderDelegate
- Fix compilation with BUILD_EXAMPLES Off

Contributors for this release

Venkata Sreekanth Arikatla, Nghia Truong, Nicholas Boris Milef, Aaron Bray, Ruiliang Gao, Johan Andruejol

5.6 Release 1.0.0

Announcement: iMSTK 1.0.0

We are introducing Interactive Medical Simulation Toolkit (iMSTK)-a free & open source software toolkit written in C++ that aids rapid prototyping of interactive multi-modal surgical simulations.

For more information, visit our website: http://www.imstk.org/

Features

- Cross-platform build

- CMake automated superbuild
- Test infrastructure (via google test)
- Continuous Integration
- Scene and simulation management
- Vulkan and VTK rendering backends
- Advanced rendering: Physically based rendering, Subsurface scattering, Decals, Shadows,
- Graphical overlays (Vulkan backend only)
- Standard user controls (pause, run, exit, pan-zoom-rotate)
- SteamVR support including (Oculus, HTC Vive (VTK backend only)
- Finite elements (linear, co-rotational, non-linear formulations)
- Position based dynamics
- Penalty and constraint-based collision handling
- Linear solvers: Direct and Iterative matrix solvers
- Non-linear Newton solver
- Collision detection (CCD, Spatial hash based collision, narrow phase queries)
- External device support (VRPN)
- Support for standard mesh input formats (.obj, .dae, .fbx., .stl, .vtk, .vtu, etc.)
- Asynchronous logging (using g3log)
- Audio support
- Haptic rendering (OpenHaptics)

Contributors for this release

Venkata Sreekanth Arikatla, Alexis Girault, Nicholas Boris Milef, Ricardo Ortiz, Thien Nguyen, Rachel Clipp, Mohit Tyagi, Samantha Horvath, Jean-Baptiste Vimort, Sean Radigan, David Thompson, Dženan Zukić, Mayeul Chassagnard, Tansel Halic, Hina Shah, Andinet Enquobahrie, Hong Li, Shusil Dangi

6 Apache License

Version 2.0

Date January 2004

URL http://www.apache.org/licenses/

6.1 TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

6.2 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means *(i)* the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or *(ii)* ownership of fifty percent (50%) or more of the outstanding shares, or *(iii)* beneficial ownership of such entity.

"You" (or **"Your"**) shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

6.3 2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

6.4 3. Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

6.5 4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

- You must give any other recipients of the Work or Derivative Works a copy of this License; and
- You must cause any modified files to carry prominent notices stating that You changed the files; and
- You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
- If the Work includes a `"NOTICE"` text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such `NOTICE` file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a `NOTICE` text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the `NOTICE` file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the `NOTICE` text from the Work, provided that such additional attribution notices cannot be construed as modifying the License. You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

6.6 5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6.7 6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the `NOTICE` file.

6.8 7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an **"AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND**, either express or implied, including, without limitation, any warranties or conditions of **TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE**. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

6.9 8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

6.10 9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

6.11 APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

```
Copyright 2018 iMSTK

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```