Iogram Documentation

MESH Consultants Inc
October 17, 2016

Introduction

Iogram is a software product that accelerates and unifies the development of 3D models, apps, and games. Iogram lets you easily create and explore thousands of prototypes during the early phases of your digital design project. It enables the tools that create these prototypes to evolve naturally with the project, potentially even maturing into robust products themselves!

Getting started

Launching Iogram

1) Launch Iogram by double clicking on the icon in your OS. You are prompted to create a new project or load from a previous project.
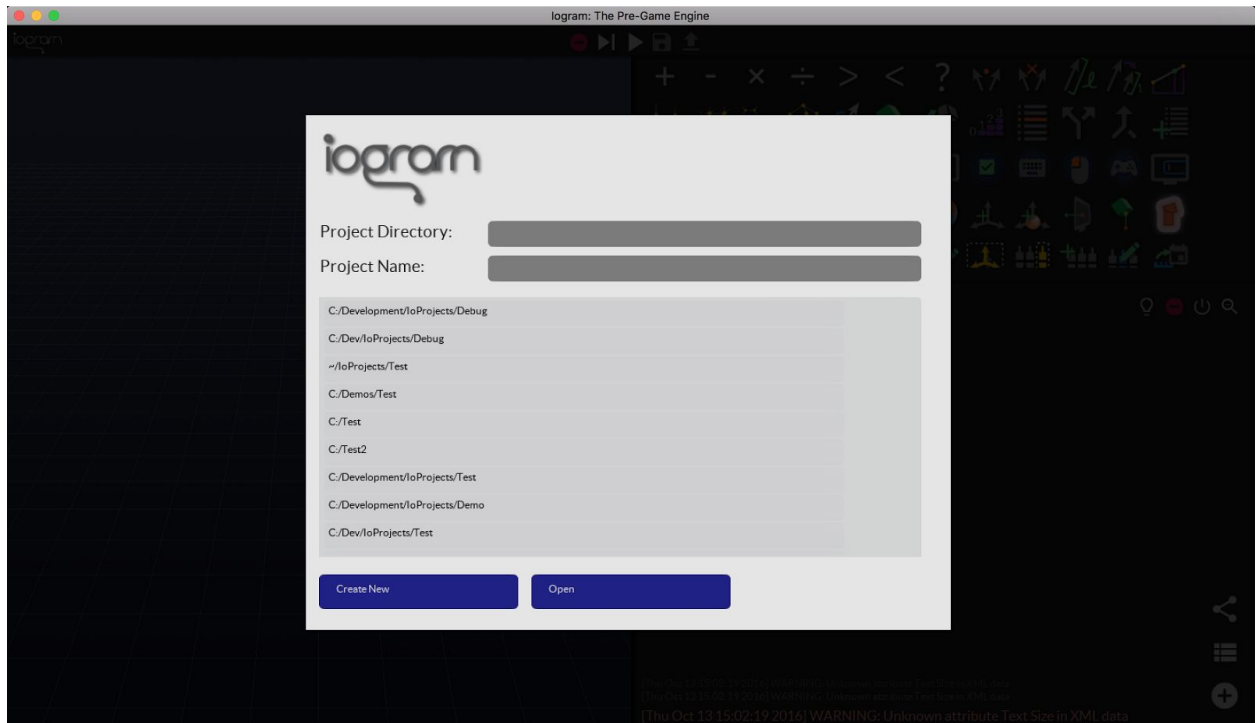
Create New Project: Enter your desired Project Directory and Project Name.
Please specify a full path to where you'd like the new project directory to be created. For example "C:/Iogram_Projects/First_Project".
Open Existing Project: Click on the relevant project from the previous projects list, or enter the project directory.

2) You will now see the main Iogram editor. If you are starting a new project, you will first need to create a new view. To do this, find the '+' button at the bottom right of the screen, and type a name to your view. Hit enter to open the view.
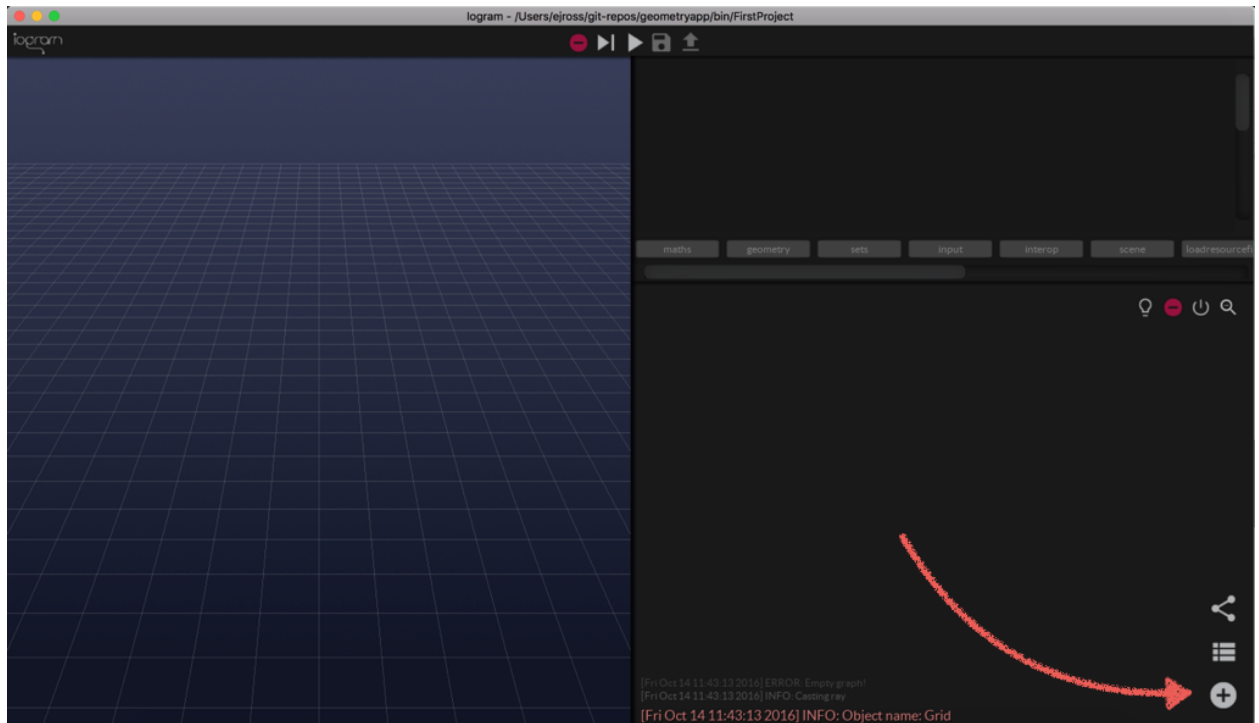
3) You are ready to start creating, loading, modifying, parameterizing, and exporting! Drag a component onto the canvas to get started. Perhaps start with a mesh cube primitive. You can also use the "ReadTriangleMesh" component to import your own geometry from obj, off, dxf, stl and others (see details below).

Example: Create a Basic App

1) Start by creating a new file, following the directions in "Getting Started" above.
2) Select the *mesh* tab on the components panel to view the available mesh-related icons. Find the **ReadTriangleMesh** icon, and drag it onto the canvas.
3) We need to give this component some input. Select the *input* tab on the components panel and find the **Panel** icon. Drag this onto the canvas. In the text box, type "Data/Models/bumpy.off", then hit enter.

4) Connect the output of the **Panel** component to the input of the **ReadTriangleMesh** component. You should see a preview of the "bumpy" mesh.
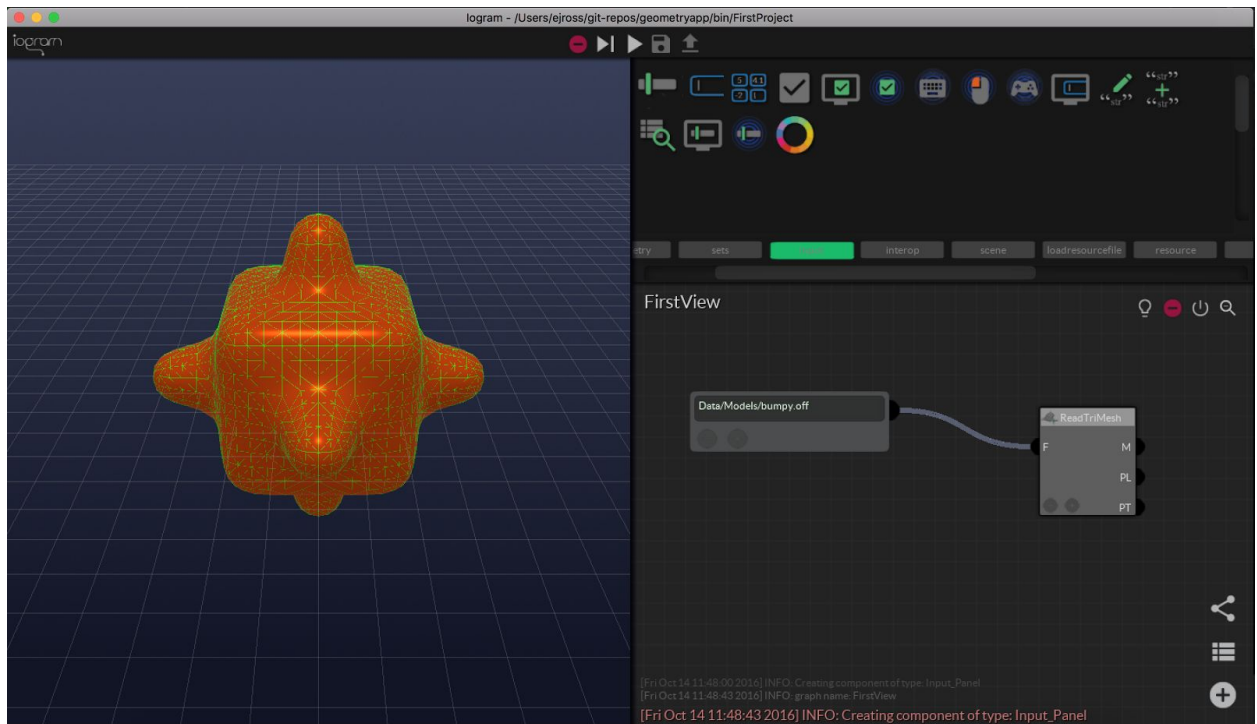
5) Now find the *scene* tab on the component panel. Drag the **MeshToModel** component onto the canvas. Connect the output of the **ReadTriangleMesh** component to the input of **MeshToModel**. This creates a pointer to the model.

6) To add this model to the scene, we'll first create a node. Find the following components (also in the *scene* tab) and drag them onto the canvas: **AddNode** and **AddStaticModel**. Connect as shown:
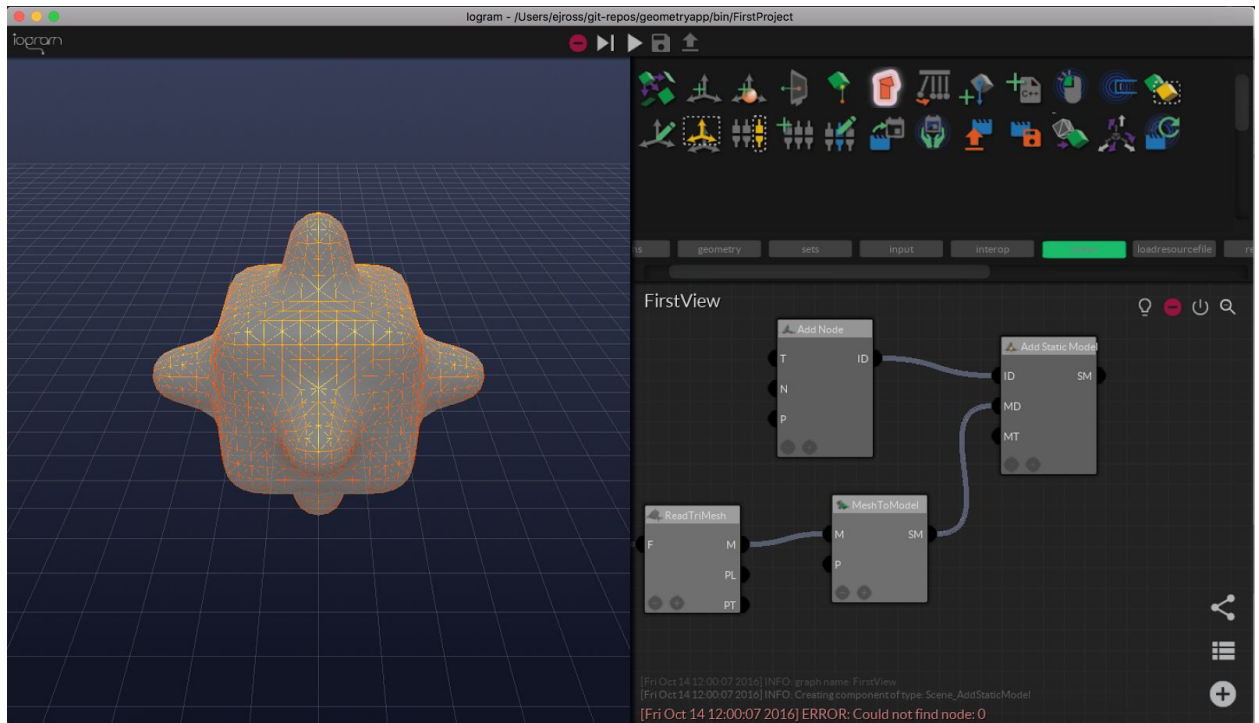
You will see some new geometry overlaid on the preview geometry. Turn off the preview on the **ReadTriangleMesh** component by selecting the component and hitting the *Preview Off* button.
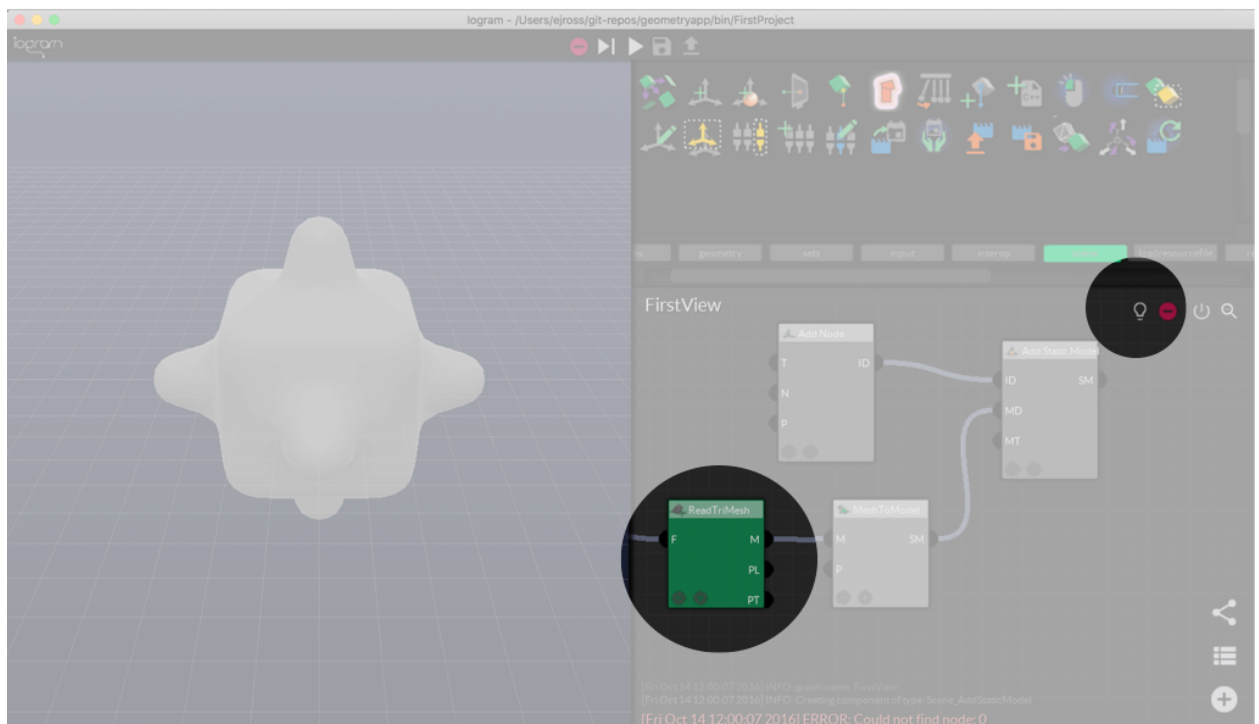

7) We'll add a very basic UI element to control the scale of the model. We need three components. From the *input* tab, drag the **ScreenSlider** and **SliderListener** components onto the canvas. From the *math* tab, find the **ConstructTransform**

Component. Hook up the components as shown, with the output from the screen slider listener entering the **S** (scale) in construct transform. Now hook up the transform to the **T** (transform) input on the **AddNode** component.
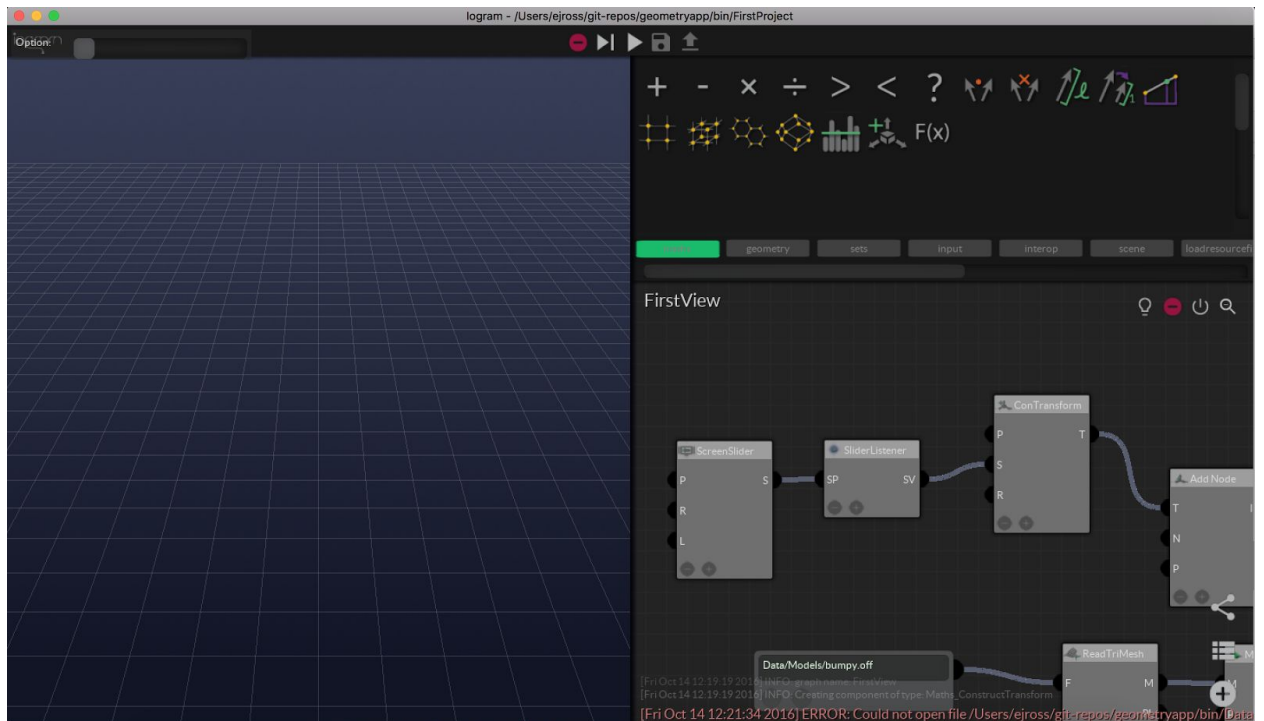
Did your model vanish? That's because the screen slider defaults to a value of zero! Move the slider to see the scaling.

8) Build the app. First save your work by clicking on the *save* icon at the top of the screen. Then find the *share* icon at the bottom right.
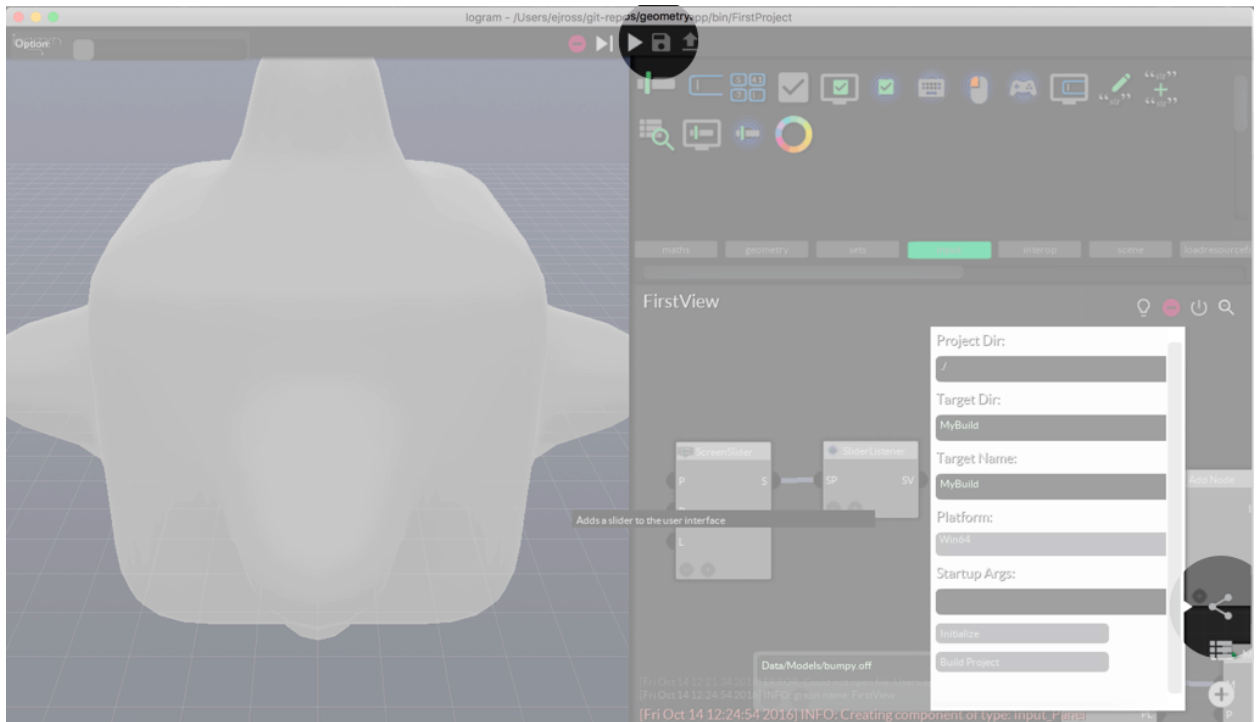

9) First select your target platform. We'll build for WebGL by selecting WebGL. Leave the TargetDir and TargetName as "MyBuild" and hit "Build".
10) There are many ways to view the WebGL app. We offer two suggestions:
    a) Open a command prompt in the MyBuild folder. Enter the following python Command (depending on your python installation):
        `'python -m SimpleHTTPServer'` or: `'python -m http.server'`
        Then in a web browser, go to localhost:8000. You should see your app!
    b) Copy your MyBuild folder into a public dropbox folder. View the app using the public link to the 'IogramPlayer.html' file in the MyBuild folder.

Elements of Iogram

The Editor View

- Components
  - Components are the basis of Iogram. All of the inputs, geometric primitives, geometric operations, and scene logic are stored as components. See "Component Index and Specifications" section for details on some of the components.
 - Components are loosely grouped into tabs

- Graph navigation (group ops on selected comps)
 - The components can be managed using the Component Manager tools.
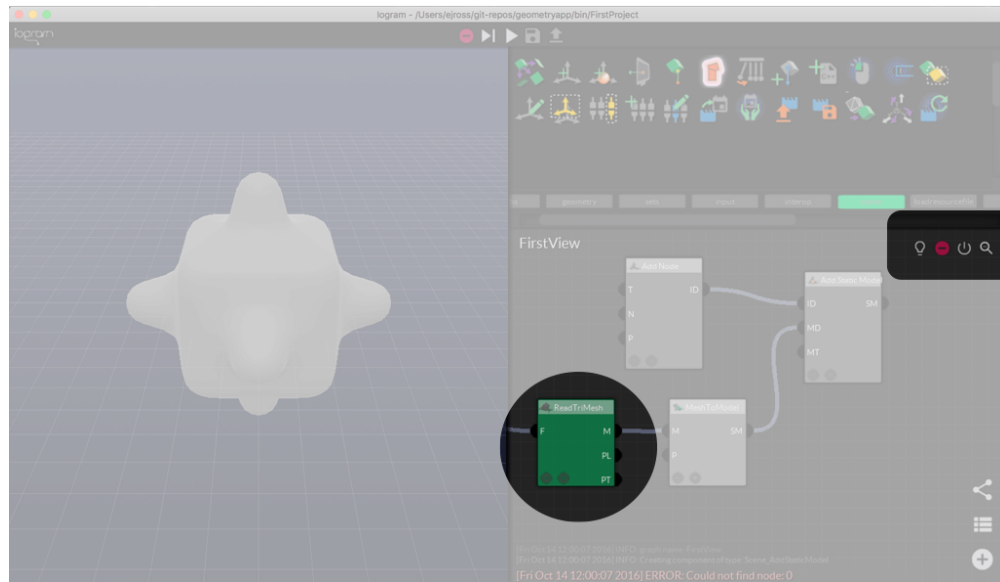
  - These tools operate on selected components, which are highlighted green.
  - If the component has **preview** geometry, this can be toggled on or off.
  - To **delete** a component, select the component and hit delete.
  - To **disable** a component from solving, use the Enable/Disable toggle button.
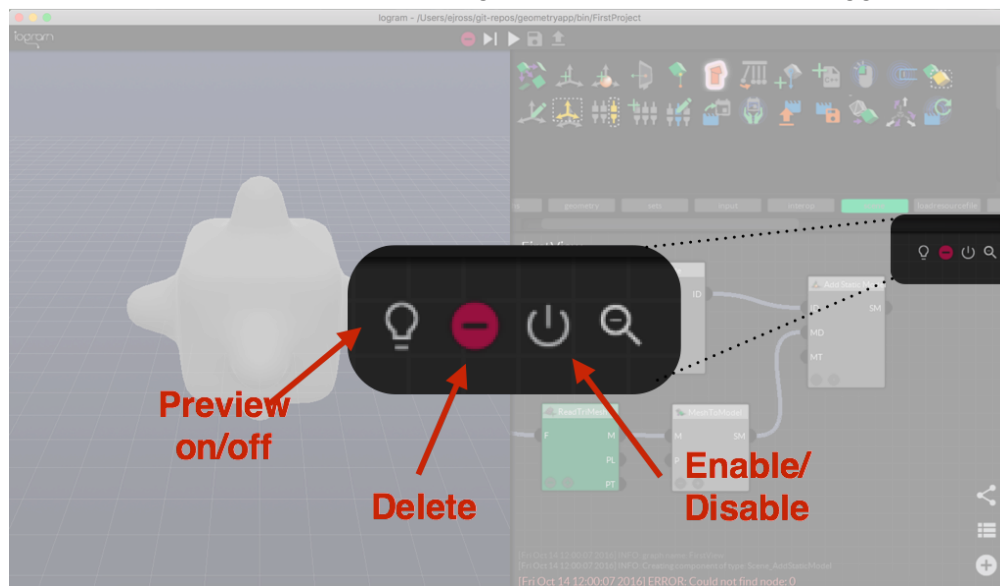
- Saving, loading, solving and project management.
  - These tools are located at the top of the display.
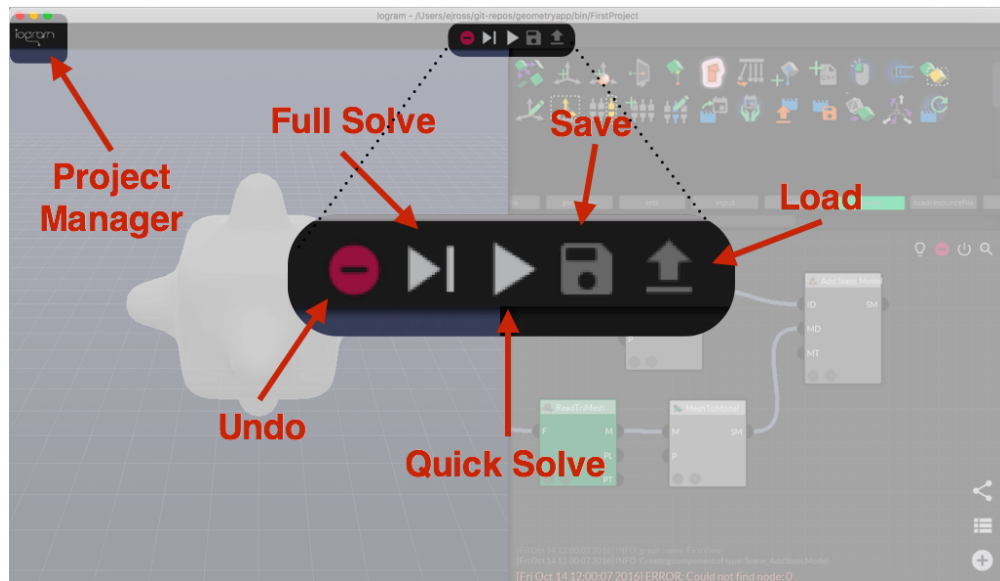
- View and Build manager.
  - These tools are at the lower left.
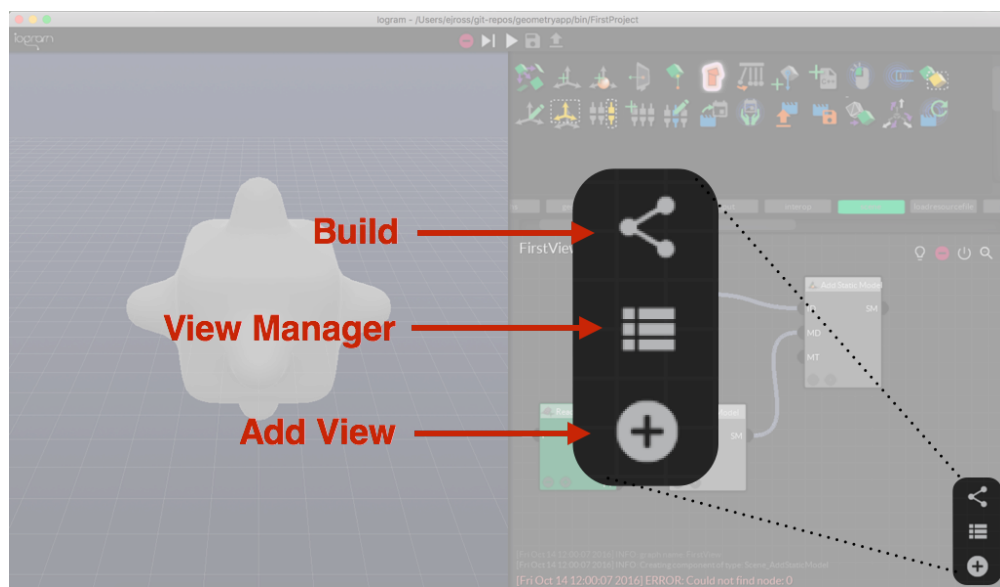
The Scene View

Navigation:
- *3D rotate:* Right mouse click and drag
- *3D translate:* Shift + Right mouse click and drag
- *Zoom:* Ctrl + Right mouse click and drag

Building a project

Iogram lets you take the functionality you've built into the Scene and export it as a standalone application. The build process constructs an executable for the platform of your choice, including Win64, OSX, Linux, and WebGl.

Use the Build manager to configure options for the targeted application:
- *Target Dir:* A folder with this name will be created inside the project directory
- *Target Name:* An executable with this name will be created inside the target directory
- *Platform:* chose from Win64, OSX, Linux, and WebGl (note OSX and Linux are disabled for this release).
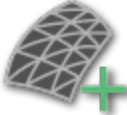- *Startup Args:* Leave startup arguments empty for now

Once all options are configured:
- *Build Project:* Builds and writes the executable for the appropriate platform to the target directory

The relevant build files (e.g. Windows executable) can now be found in the MyBuild folder.

Component Index and specifications

Below we summarize only a small selection of key components. Basic descriptions of the components and their inputs and outputs are accessible in the app by hovering on the components and their input and output slots.

Name	Description	Inputs	Outputs
ReadTriangleMesh (mesh tab)	Reads a triangle mesh from disk. *Preview Geometry:* Meshes, Polylines, Points	**F**: Path to file. Accepted inputs: .3ds .blend .dae .dxf .lwo .obj .ply .raw .stl .off	**M**: Meshes **PL**: Polylines **PT**: Points
Panel (input tab)	Flexible input to components. This will default to a string, but may accept other arguments. The input text is parsed into two section, separated by a comma. The content should precede the comma, followed by the type. Examples: 1, int 1.0, float 2.1 3.5 4.0, Vector3 - Hello, string *Preview Geometry:* none.	N/A	String, int, float, quaternion, path

iogram

Inspector (input tab)	Allows inspection of component output. Note that meshes and polylines are stored as "VariantMap"s. *Preview Geometry*: none.	Anything.	Will output exactly what it reads as input.
AddNode (scene tab)	Adds a node to the scene. This node is associated with a transform (position, scale, rotation) and an ID. We can associate models and behaviours with this ID. *Preview Geometry:* axes.	**T**: transform. Defaults to the origin. Will also accept partial information: Vector3 position, Float (uniform) scale, Quaternion rotation. Or hook up the output of the **ConstructTransform** (math tab) component. **N**: (optional) name. **P**: (optional) parent node.	**ID**: node ID.
ScreenToggle (input tab)	Places a button in the scene. Use in combination with **ButtonListener**	**P**: Button position in screen coordinates **L**: Button label (e.g. from the **Panel** component)	**Ptr**: pointer to the UI element
ButtonListener (input tab)	Listens for button input from the scene button. Translates the values back to the graph. Use in combination with **ScreenToggle**	**BP**: Pointer to screen button (output from **ScreenToggle**) **M**: Boolean to mute the listening. Defaults to true.	**V**: (Boolean) values from the button.
RigidBody (scene tab)	Adds rigid body behaviour to a node.	**ID**: node ID **PW**: Pointer to the physics world. Connect to the output of the **PhysicsWorld** component. **M**: Mass. Set to zero for fixed objects **S**: Collision shape type. Accepted types (as strings) are: "Box" "Sphere" "Convex"	**PB**: pointer to rigid body **CS**: pointer to collision shape.
PhysicsWorld (scene tab)	Initializes a physics world (simulation).	**Go**: (Boolean) toggle physics on or off. **G**: Gravity vector. Defaults to the usual Earth gravity vector.	**PW**: Pointer to static physics world.Connect this to any rigid bodies you wish to include in the simulation.

iogram