1 Welcome to Golog!

Welcome to the Graphical Ontological Logger, a.k.a. Golog! In this pdf, we will describe the very high level usage of the golog program.

The purpose of an ontological log is to organize knowledge in the form of a simplicial set, golog just adds a graphical interface.

By the end of this brief introduction, you should be able to build your own gologs, as well as view the more in-depth documentation, which itself is presented as a golog.

2 Running Golog

2.1 Requirements

Before we begin, you must install the latest version of python3. Once you have done so, please install the latest version of the graphics engine Panda3d

```
pip3 install panda3d
```

To update golog or install golog, from chosen directory in terminal, run

```
git pull https://github.com/nopounch/golog master
```

2.2 Opening a Golog

To begin, run the file "run.py" from python3

```
python3 run.py
```

You will be prompted to create a new, or load a golog file. To create a new golog file, type a name for the golog and select "new", you will be prompted to select a save location. The default save location is under root/user_files/save

To load a golog, select "load". You will be prompted to choose a file location. Gologs are saved using the .golog extension. For example, this documentation is actually part of a golog file saved in

```
root/theory/Documentation/Documentation.golog
```

Try Loading the Documentation golog now.

If you wish to load the most recently used golog file (ex: in the event of a crash), there may also be a "recent" prompt, which will skip selections and open the most recent golog.

3 Controlling Golog

There 2 main processes to golog right now:
- Creating and Manipulation Simplecies (see simplicial set)

- Associating and Opening Simplicial Data

3.1 Creating Simplecies

To Create a 0-Simplex:
simply right click anywhere on the blank canvas. You will be prompted to label your simplex. You will also be prompted to associate math_data, ignore this for now. Hit "new" and you should have successfully created a 0-Simplex.

To Create a 1-Simplex:
select two 0-simplecies by left clicking them, again you will be prompted to label, hit "new" and you will create a 1-simplex between the two selected 0-simplecies.

To Delete Simpleces:
hover over a simplex with your mouse and press your backspace key. You will be prompted on what sort of data you will be destroying.

To Move Simplecies:
Simply click and drag them

3.2 Associating and Opening Data

To every simplex, there is associated data. When creating a simplex you were prompted to create data along with it. Although theoretically any data type can be used, the following data are currently handled by golog

```
[None, Golog, File, Text, Latex, Weblink]
```

The default data type to a simplex is the None Data. But if other data is provided, golog will understand how to open it. When creating a simplex, you can choose to associate data to it. If you want to change this data, you can do so, as detailed below. If you want to interact directly with the data (by opening it) golog provides handlers for all of the data types above.

To Update / Change Data:
Hover over a simplex, and press the "u" key. This will also allow you to change the label of a simplex. If the simplex has no data, you can also press the "space" key.

To Open Data:
Hover over a simplex, and press the "space" key. If the simplex has no data, this will bring up the prompt to Update the Data.

3.3 Golog Data

The original goal of the golog program was to demostrate the concept of an "ontological expansion". To this end, the most special type of data one can associate with a simplex of a golog, is a golog itself. If you associate a golog to a simplex, and open it, you will open up a new window with the golog data displayed. You can control this golog like you would control any golog,

creating simplecies and data as you wish, and even nesting further gologs into the new gologs simplecies. This is meant to represent the hierarchical nature of information. It also happens to be a very convenient way to organize knowledge.

An example usage of such an organization of knowledge in the form of a golog, is the documentation. Try opening the documentation.golog in golog, and exploring it's subgologs to learn how the program itself works.