ph

A c++ library

Introduction

Ph is a software project that aims to be the facto software for both building and testing the best C++ software. It aims to cut loose from other languages in the build process, instead integrating them directly in the source code such as this:

```
import Ph.Build;
import Ph.Concepts.Integer;

auto main (int i, char** s) -> int
{

    Arguments auto args = parse_args (i, s);

    Error auto err = len (args) > 0 ? true : false;

    if (err)
    {

    }

    return err;
}
```

"Ph" is a lightning fast C++ library and also a build tool (such as Cmake or Make) for turning C++ source code into something useful.

It introduces c++ concepts and modules, by which the former it is built around, thus, making it lucrative for industrial applications where the need for precision is critical, and where most of the bugs can be catched before the c++ files are even compiled, thus simply just generating syntax errors for developers. Introducing this core language feature has huge pros for industrial-strength generic components, AKA good software. This is what we want! We dont want python's "throw in whatever type u desire", and not java's somewhat pragmatic "please specify the type". We want to be able to say "hey, other coders out there

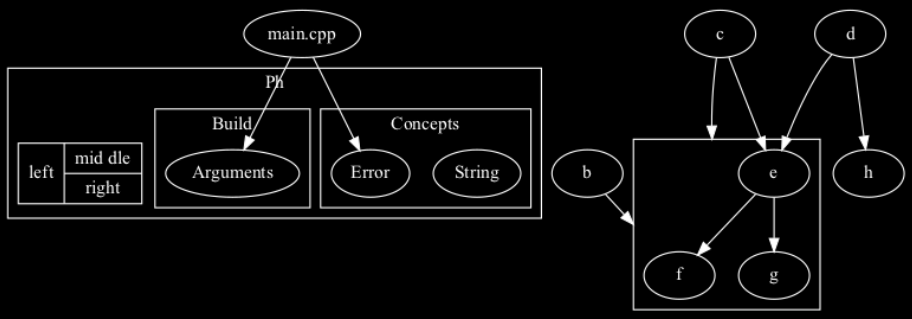

Figure 1:

using this function im about to type, just throw in a String". String is just a concept that we specify. It could be everything from "std::string" to old plain c string "char const *". We say "hey, String can be either a "std::string" or a "char const"*. OR it could be anything that we could do the following with:

```
template <typename S>
concept String = requires (S s)
{
    s.size ();
    {s [0]} -> char;
}
```

Goals with this project ==========

- ☒ Motivation
- ☐ Just c++, even for building.

  All software are built around a set of programming languages, often one for front-end and one for back-end. The reason for this, unknown. One can only guess.

  I can not stress this enough, but writing software in one language has huge upside effects, and it's much cheaper. Your teams can speek the same language, thus making it much easier for further intrigues.
- ☐ C++ interpreter

  Similar to python interpreter.

Details

file dependency

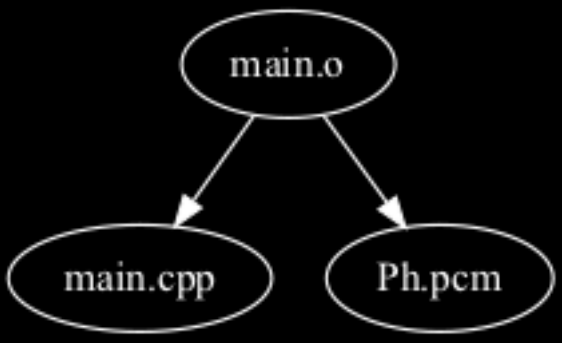

Figure 2:

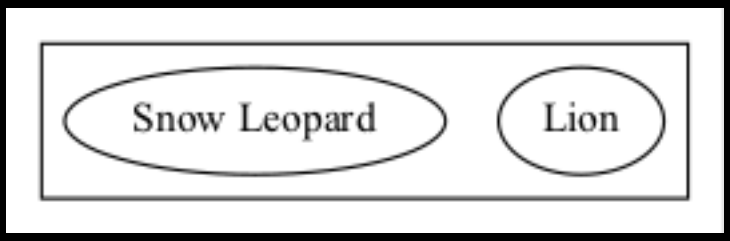

Figure 3:

Project dependencies

The following graph describes that basically “ph” is a set of files which will either be transformed into a documentation file or into the executable software.

Source code written in

- C++

and built with

- Make

About

Documentation

The documentation for Ph is dependent on the the following languages:

- Markdown

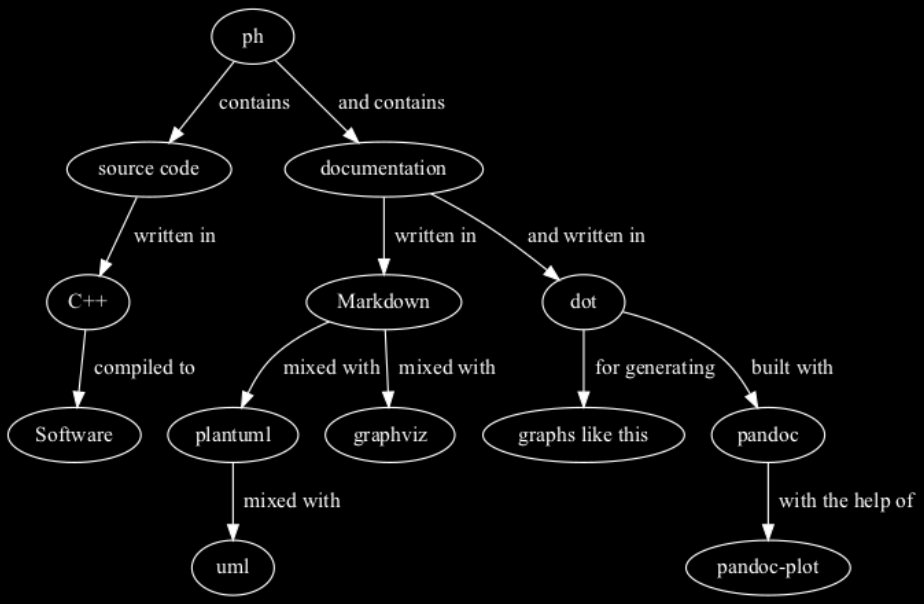

Figure 4:

- dot
- UML
- English

and built with

- pandoc
- pandoc-plot
- pandoc-plantuml-filter

Contribute

At the moment, the project is very dependent on Make (for compiling C++ files into different things and then finally everything into an executable).

Also pandoc (for documentations).

Reasons for this? It's still very young.

Please help me develop this project! At the moment there are just one developer.

Licensing

Can be either open or proprietary.

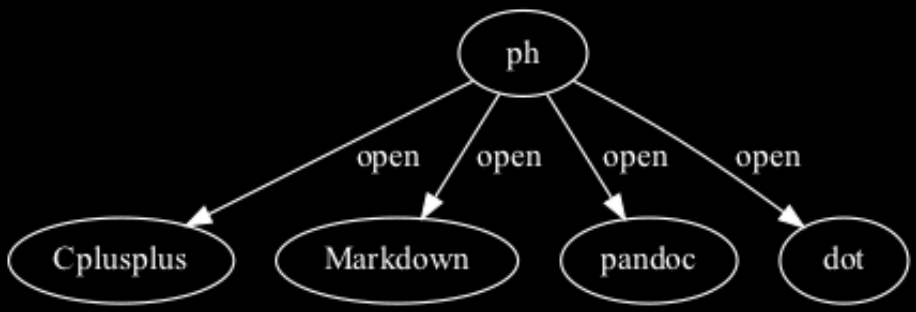

Figure 5:

Concepts architecture

Iterator-relations

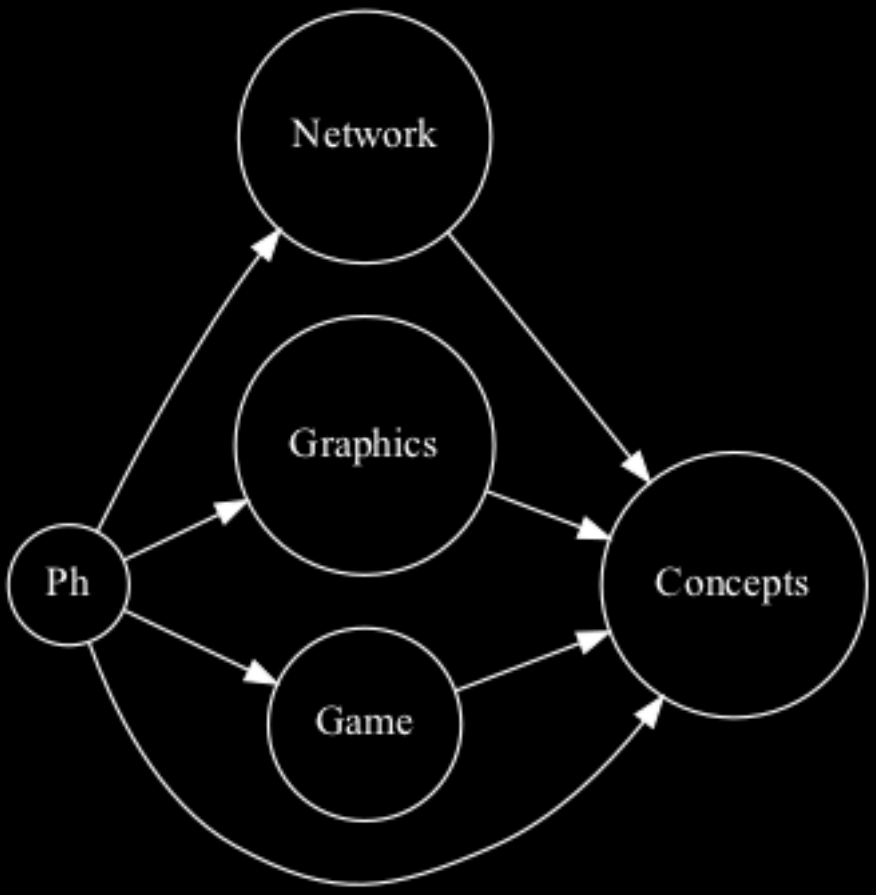

Figure 6:

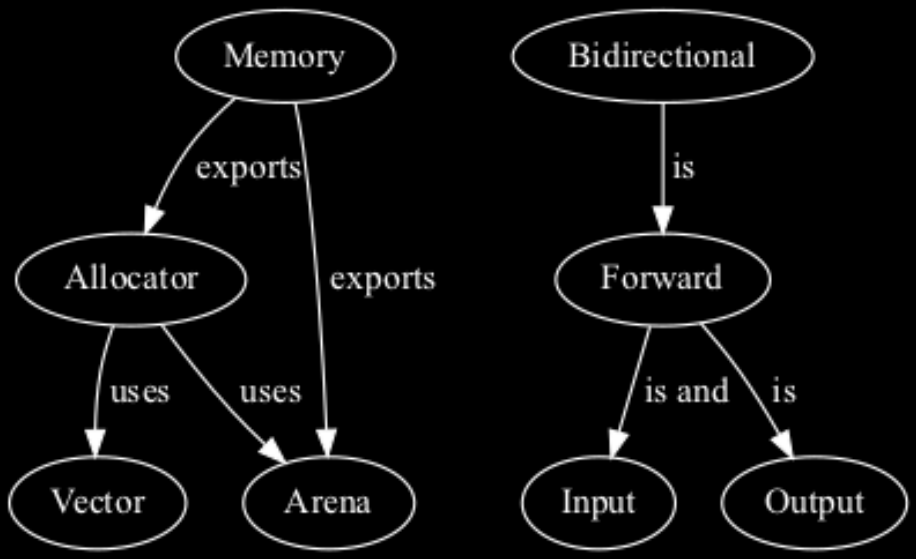
Memory
Bidirectional
exports
is
Allocator
exports
Forward
uses
uses
is and
is
Vector
Arena
Input
Output

Figure 7:

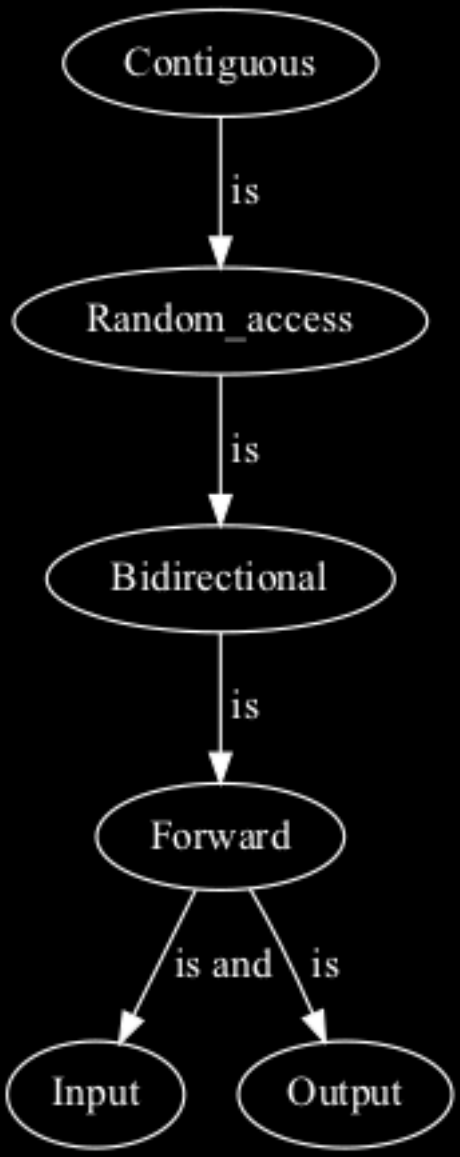
Contiguous
is
Random_access
is
Bidirectional
is
Forward
is and
is
Input
Output

Figure 8:

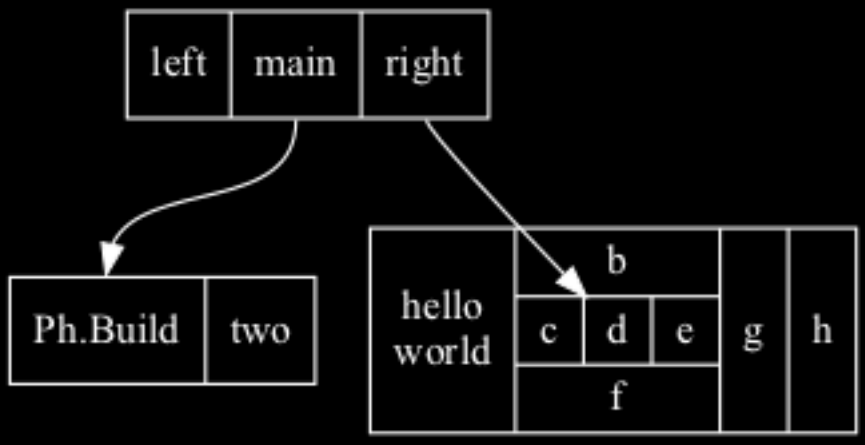

Figure 9: