Technical Specifications

Jason GROSSO

January, 2023

Abstract

This is the Technical Specifications for the project group 8 during the year 2022-23 at ALGOSUP. This document describes the practical aspects of the project such as its functionalities, which technologies were used and where, as well as the architectures used.

Contents

1 Introduction

1.1 Presentation

The objective of the project is to bind Rust to FABGen by using the client's existing C++ binding generator. The users should be able to use Harfang3D with Rust as if they were using C++..

1.2 Project Team

This project will be made over the course of 7 weeks, our Project Manager is David and is in charge of organizing the work and dividing the tasks to the team.. Max is the Program Manager, his job is to define the high level functionalities. Jason is the Tech Lead, his job is to chose how the functionalities will be implemented and which technologies will be used. Louis is in charge of the Quality Assurance, he will be testing every functionality of the project to find and point out any bug found, finally, Leo is our software engineer, and he will be the one responsible of implementing most of the functionalities.

2 Scope

Item	In Scope	Out Of Scope
Rust Binding	X	
Unit Test	X	
Reliability	X	
No Polymorphism At Run Time		X
Working Harfang3D-Rust Example		X
Ruby Binding		X

Table 1: Scope And Out Of Scope

3 Approach

We will base ourselves on the GOlang binding and unit tests to create our bindings and unit tests, because both GOlang and Rust are both statically typed.

4 Risk and Assumption

We are assuming that the software provided by Harfang3D is working perfectly and that there will be no conflict between their software and our solution.Therefore, the risk is that any issues encountered during integration could result in the failure of our bindings.

5 Process

We use the Python description of the C++ library, pass them into FABGen, which will return a .c, .h, and a .rc file; by using the clang command we will transform the .c file into a .o file usable by every language. This file will be then called by the users via the .rc file to access the C++ library(see 1).
Here we will demonstrate using the GO version of the generator; we launch FABGen we use the command line in 3. This will start the process in 2 to create the bindings.
The first part consists of setting the generator to the chosen language, to do this bind.py we will go into the language file of the chosen language, and send the info to gen.py in order to activate the GO version. Once it is done it will send the info to bind.py.
The second part is a verification of the passed file in this case,harfang.py: it will execute lots of small functions between go.py gen.py and himself and it will also check some functionalities.
The third part mainly takes place in the std.py file of GOlang it consists of transforming the GO variable types to C variable types, after that bind.py will execute the bind generator, apply the clang command and print the files in the directory passed in the starting command.

6 Success Evaluation

The success of the project will be evaluated by determining if we have successfully passed the 30 unit tests supplied by Harfang and created a binding for the supplied harfang.py file.

Glossary

Bind Allow a programming language to use a library from another programming language.

Binding generator Automatically creates a binding for any given language.

C++ Generalist programming language.

C Generalist programming language.

Clang command Transform a file into an object file.

Harfang3D A visualisation tool designed for the industry.

GOlang / GO Programming language statically typed.

Library Reusable collection of prewritten code that can be used for optimisation.

Polymorphism In programming language theory polymorphism is defined as a function that can accept multiple types.

Unit tests Tests on a specific part of the code.

Ruby Programming language focusing on simplicity and productivity.

Rust Programming language statically typed.

Statically Typed Is said of a language where the variables types are known at run time.

Diagrams

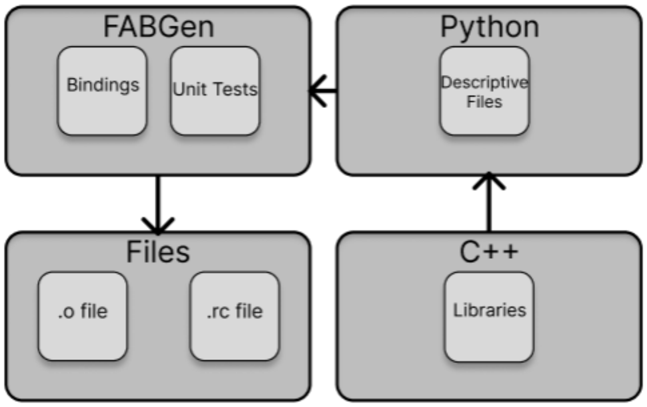


Figure 1: Project Architecture

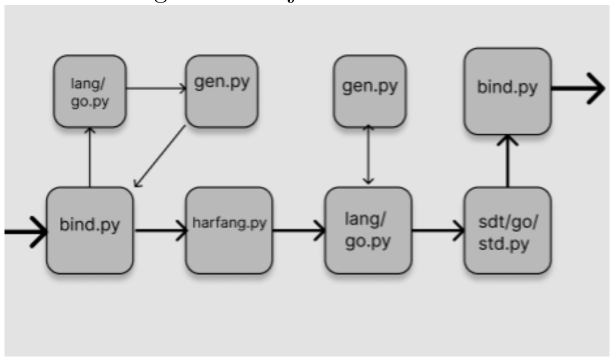
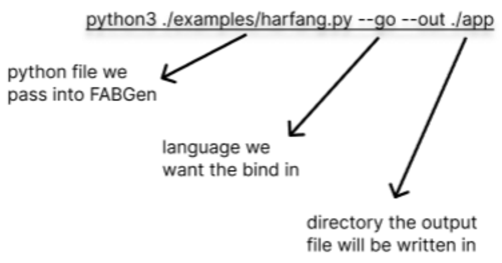


Figure 2: FABGen Architecture(GO example)



```
python3 ./examples/harfang.py --go --out ./app
```

python file we pass into FABGen
language we want the bind in
directory the output file will be written in

Figure 3: Command example