Developing the 3D bin packing project

Cipher Kuo

October 2021

Contents

1 Contact **2**

2 Prerequisites **2**

3 Architecture introduction **3**

4 Deploy the fontend **4**

5 Deploy the backend **6**

6 About JSON data and API **6**

1 Contact

shengchikuo@gmail.com
cipherk@supermicro.com.tw

2 Prerequisites

Before getting your hand dirty, you will need to setup your development environment to make sure the system you are using is able to let you follow the instruction guide rapidly without any fine tuning.

This instruction guide will take the Linux(Ubuntu20.04) as an example, and show how to setup the environment under this Linux distribution step by step, furthermore, we have successfully tested this project under the Windows system, but this document will not mention the relevant settings on Windows.

Therefore, it is strongly recommend using the same operating system to prevent you from unknown trouble shooting. If you are currently using Windows operating system, You can still use the following methods without changing the operating system, you can activate WSL2(Windows subsystem linux)to have a Linux like environment and shell, you can open the WSL2 feature by following the Microsoft official manual., and it is default installed on Windows 11

In case have unknown problems under other operating systems, please raise an issue on Github or write to contact

Download the source code

Before download the source code, make sure there is **git** on your system.
following instruction will install the **git** on your Debian like(Ubuntu) system.

```
sudo apt-get update
sudo apt-get install git -y
```

After download the git, your are able to download the source code to current work directory.

```
git clone https://github.com/N0nent1ty/3D-Binpacking-GUI
```

Because the font end side of this project employeed the **npm** as a javascript library package management tool, you will need to install the nodejs in advance. Following instructions will installed the nodejs, and let you able to use npm.

```
sudo apt-get update
sudo apt-get install nodejs -y
```

```
sudo apt-get install npm
```

Science the back end of this project is written in **Python3**, and utilized a python package Flask(a Web framework) to develop the CGI, to install a package, you will need to install the python, and python package.

```
sudo apt-get update
sudo apt-get install python3 -y
sudo apt-get install python3-pip
```

Afterward, you are able to install flask by following instruction.

```
pip3 install Flask==2.0.2
pip3 install flask_cors==3.0.10
```

Dependencies is scanned and generated by tool pipreqs

3 Architecture introduction

The entire architecture can be basically separated into three modules.

1. USER INTERFACE
2. ALGORITHM
3. 3D-RENDERING & other interactive functionalities.

In the following paragraph we we will use abbreviation 3D-RENDERING for short.

In order to have more user friendly functionality, we combined the USER INTERFACE and 3D-RENDERING together, so that what user need to do is type the necessary input information into the USER INTERFACE and click a send button subsequently, then render image will show up.

hypothetically, let us assume what the usage scenario would looks like in case we separate this two modules. In such context, The user will use user input module to formalize the input data and let the algorithm module to generate a json or bit format data output which is definitely not human readable, then the user need to feed this output file to another application(third module) so that the output became visible.

This approach seem like cumbersome but it has its own advantage. Take our project in example, the javascript libraries and css libraries has its own dependencies, and some will conflict with each other.

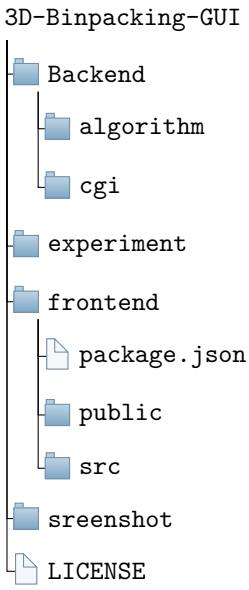
Assume that we only consider making a web services for this algorithm (there are other options like desktop app base on unity3d or unreal engine, or mobile

app, .etc), the reason we choose Vue2 as our USER INTERFACE module and Babylonjs as 3D-REDERING module for this project use not because this solution are the best option in their respective modules, but because they have the best compatibility while we want to combine these two modules together to achieve better user experience.

Therefore, instead of three module in this project, what you will see is only two moudle, frontend and backend, The frontend is merged with USER INTERFACE and 3D-RENDERING. The backend turn the alogrithm module into a CGI, take it as a web backend service.

You can take a view of architecture of the project by following instruction.

```
tree -L 2 ./3D-Binpacking-GUI/
```



In order to keep modern font-back-seperate design and under the aforementioned architecture, you also can see the algorithm folder under the backend folder, algorithm module will work as a back end service which will call by a Rest-Full API.

4 Deploy the fontend

If you have settled the environment by the previous instructions, you should have npm on your system, once you get the npm on your system, installing the required dependencies will become quite simple, what you need to do is go to the front end directory and

```
npm i
```

the dependencies listed in "package.json" will automatically install for you.
To start a developer server, type the following instruction on the terminal and enter

```
npm run serve
```

If everything goes well, it will open the default port number port 8080 and you should see the following output result,

```
  App running at:
  - Local:   http://localhost:8080/
  - Network: http://192.168.88.243:8080/

  Note that the development build is not optimized.
  To create a production build, run npm run build.
```

Now you're able to access the http://localhost:8080/, you should see the home page the user interface.

It is worth to menthion that this server is only for development and test, of course you can deploy them with Nginx or other web server like Apache httpd, but this is out of the scope.

In the front-back-separate design, the front end will need to communicate the back end service with specific url and back end port, the default is number 5000, if you want to change the back end listening port, remember to change the this setting to specify where the front end data will send to.

The configuration file for the port is under the following path, `$project_dir/frontend/src/config/index.js` Then modify the default port and IP setting to wherever you want, change the following setting will change the default POST request destination by AXIOS package.

```
module.exports = {
    API_LOCATION: 'http://localhost:5000/'
}
```

For example, you can check the source code in `\$project_dir/frontend/src/component/comfirminformation.vue` The code of this component is mainly responsible for the function of normalizing the data after the user input is completed and sending it to the back-end api.

check the sendStoredMessage function in this file, you can see the following code snippet.

```
    this.axios({
        method: "post",
        baseURL: API_LOCATION,
        url: "/api/recv/3dbinpack/info",
        "Content-Type": "application/json",
```

```
        data: {
          pallet_mode: this.pallet_mode,
          containers: this.container_infos,
          boxes: this.box_infos,
          pallets: this.pallet_infos,
        },
      }
```

In above context, if you didn't change the default setting, the API_LOCATION= 127.0.0.1:5000, then the post request will send to http://127.0.0.1:5000/api/recv/3dbinpack/info.

5 Deploy the backend

Please note that, although the algorithm module is based on py3dbp which is a open source python package credited by this repository, the same-name package in this project is modified and changed lot's of the function interface to make it can fit the need of our own project, therefore, do not install the standard package with the **pip3 install py3dbp**, install the customized package by the following instruction instead.

if you install the the py3dbp in advance please remove it first.

```
cd \$project_dir/backend/algorithm/
sudo python3 ./setup.py install
```

To startup the backend gunicorn server, there is script name under the backend folder, what you need to do is install the gunicorn server with pip then start up the server with this script.
install gunicorn

```
pip3 install gunicorn
```

run the script to start up backend server, this script will open port number 5000 on your machine. install gunicorn

```
cd  $project_dir/backend/cgi
./run_gunicorn.sh
```

6 About JSON data and API

As aforementioned in previous section, the project architecture can be viewed as as front-back-separate web service and this two module communicate with each other via rest-full API and JSON format data,

If you want to replace the one of these modules or you want to modify function behavior, you should at least understand the relevant interface specifications.

Example JSON file


```
{
    "status": 110,
    "total_container_types": 1,
    "total_pallet_types": 1,
    "total_box_types": 21,
    "containers": [
        {
            "ID": "873e56cc-d41a-4011-a6a6-d08819ee68ba",
            "TypeName": "helloContainer_0",
            "TypeIndex": 0,
            "X": 227.0,
            "Y": 225.0,
            "Z": 580.0,
            "Weight_limmit": 20000.0,
            "Fitted_items": [
                {
                    "ID": "2e027d5a-52f3-4e51-9a4d-2d1588b5a3da",
                    "TypeName": "122X102_0",
                    "X": 122.0,
                    "Y": 12.0,
                    "Z": 102.0,
                    "Weight": 64.0,
                    "position_x": 0.0,
                    "position_y": 0.0,
                    "position_z": 0.0,
                    "RotationType": 0,
                    "TypeIndex": 1,
                    "Fitted_items": [
                        {
                            "ID": "da4e09b6-4a45-4d53-abe2-8003e2ddb48d",
                            "TypeName": "box1_0",
                            "X": 10.0,
                            "Y": 10.0,
                            "Z": 10.0,
                            "Weight": 2.0,
                            "position_x": 0.0,
                            "position_y": 0.0,
                            "position_z": 0.0,
                            "RotationType": 0,
                            "TypeIndex": 0
                        },
                        {
                            "ID": "272e7b8d-2806-4bdf-9991-0081f5bd4944",
```

```
                            "TypeName": "box1_1",
                            "X": 10.0,
                            "Y": 10.0,
                            "Z": 10.0,
                            "Weight": 2.0,
                            "position_x": 10.0,
                            "position_y": 0.0,
                            "position_z": 0.0,
                            "RotationType": 0,
                            "TypeIndex": 0
                        },
                        {
                            "ID": "45e56304-6def-4c5f-ad3b-0ba33b3e5329",
                            "TypeName": "box1_2",
                            "X": 10.0,
                            "Y": 10.0,
                            "Z": 10.0,
                            "Weight": 2.0,
                            "position_x": 20.0,
                            "position_y": 0.0,
                            "position_z": 0.0,
                            "RotationType": 0,
                            "TypeIndex": 0
                        },
                    ]
                }
            ],
            "UnFitted_items": []
        }
    ]
```


It is quite complicated at first glance, but maybe we can use programming language and some structure to detailing the format, let use Python and it's own class to describe how this file were generated and send back to frontend(3D-RENDERING).
To be frank, I will just use the backend code to explain how it were generated

Code snippet from backend/cgi/main.py

```
    final_info_dictionary={
        "status":statusNumber,
        "total_container_types":total_container_types,
        "total_pallet_types":total_pallet_types,
        "total_box_types":total_box_types,
        "containers":containers_array,
        }
```

final_info_dictionary is the structure definition that will be returned by algorithm module, You can compare this structure with the example json, their structure is the same.

Recall that there are three object as input that will passed into this algorithm, they are "container", "box", "pallet", and the function return will not only turn back the basic information they have, but also the sorted information like position, rotation type,etc.

- "status":
  An integer that will tell the packing status to the 3D-rendering module.
- total_container_types
  An integer that will tell how many container types were passed into this algorithm module.
- total_pallet_types
  An integer that will tell how many pallet types were passed into this algorithm module.
- total_box_types
  An integer that will tell how many boxes types were passed into this algorithm module.
- containers
  A container object(structure) array which can be viewed as another JSON(Python) structure that contained with many different container information.

As for what the "containers" looks like, it is a array of containers object, namely, containers= container[], and we are going to detail what the "container" looks like. Code snippet from backend/algorithm/main.py

```
        return{
            "ID":self.ID,
            "TypeName":self.name,
            "TypeIndex":self.type_index,
            "X":float(self.width),
            "Y":float(self.height),
            "Z":float(self.depth),
            "Weight_limmit":float(self.max_weight),
            "Fitted_items":FittedItemArray,
            "UnFitted_items":unFittedItemArray,
    }
```

- ID
  It is a UUID version 4 for each container, this ID as a unique ID that you can search a specific information of container via this ID.

- TypeName
  TypeName is a user specific string, user named the container with this string, basically it do nothing in the entire process, it will let the user to distinguish the data he has created.

- TypeIndex
  TypeIndex is a interger that from 0 to total_container_types, while the container have different value in TypeIndex, that container will be rendered into distinct color.

  You may wondering why not just use TypeName to determine the type number of object, it is just in case the user want to create a object with different height, different width and different deep containers, but they all are in same name, because user want them to.

- x
  Width of the object

- y
  Height of the object

- z
  Depth of the object

- Weight_limmit float, the limit of weight of the container, The unit of measurement is kilograms(KG).

The name of this variable is Weigh_limmit but not the weight_limit, a extra alphabet "m" is a bug(type error while programming), but the current version (2021/10/19) has not been fixed yet.

- Fitted_items
  Fitted_items is a array of pallet object that were packed into this container, about the definition of the pallet object, it basically looks the same in structure, I believe you can find the definition by yourself in the code.

- UnFiited_items
  This variable is reserve for future use, it without a practical usage in current version.you can just ignore this value.