Final Report: OpenSCAD Package Manager (OSPM)

Thespians-6156

Jiaxin Su (js4722, js4722@columbia.edu)
Brennan Wallace (bgw2119, bgw2119@columbia.edu)
Stanislav Peceny (skp2140, skp2140@columbia.edu)

I. What We Have Delivered

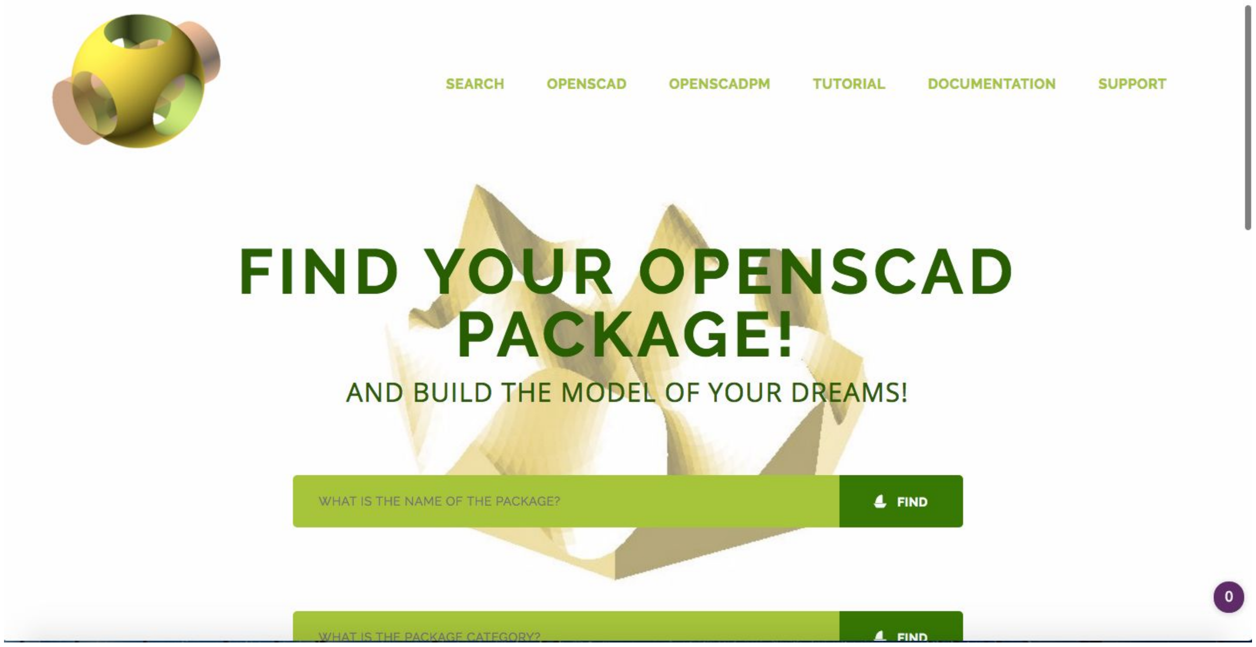
This project delivers the package manager of OpenSCAD, which includes a website and a command line tool. The command line tool is utilized for the installation and managing of modules by allowing users to use terminal commands. The website allows users to upload packages. It also serves as a way for packages to be browsed and explained with a high level description (this seems to be the norm for package managers).

Below are the detailed functionalities for the website and the command line tool respectively:

1. OSPM Website

- Search packages by package name or description.
- Search packages by package category.
- Search packages by package creator name (author name).
- View all packages.
- Download OpenSCADpm.
- Browse popular categories like geometry, architecture, mechanical, and artistic.
- Check out top packages.
- Access Github repository, OpenSCAD documentation and OpenSCAD gallery.
- Access our email contact information.

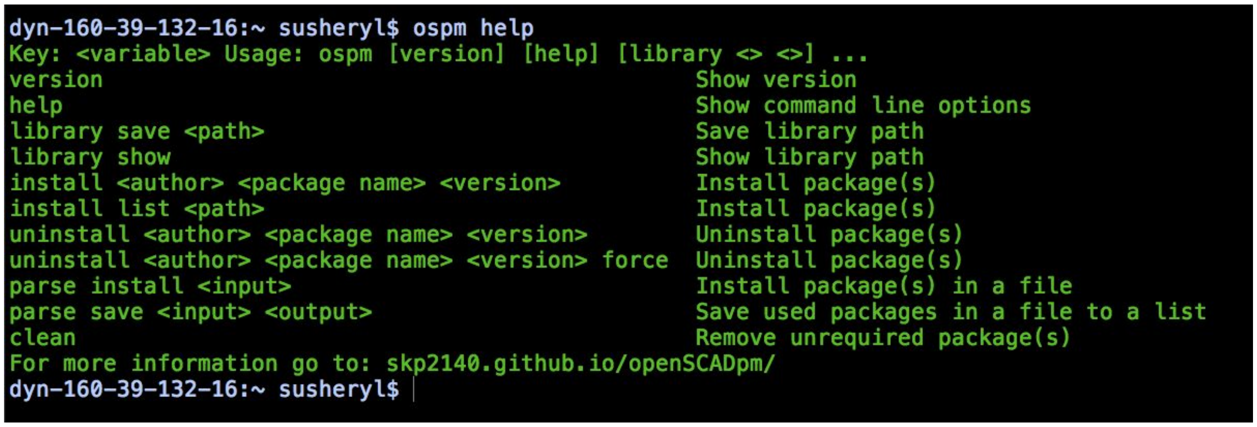
Here is an image of the index page of our website.


SEARCH
OPENSCAD
OPENSCADPM
TUTORIAL
DOCUMENTATION
SUPPORT
FIND YOUR OPENSCAD PACKAGE!
AND BUILD THE MODEL OF YOUR DREAMS!
WHAT IS THE NAME OF THE PACKAGE?
FIND
WHAT IS THE PACKAGE CATEGORY?
FIND
0

2. OSPM Command Line Tool (OSPM CLI)

- Install packages by user script that contains dependency information.
- Install packages by package name.
- Uninstall packages by package name.
- Remove dependencies for packages that have been uninstalled.
- Show library path to the local OpenSCAD library on the machine.
- Save library path to the local OpenSCAD library.
- Show version information about the ospm installed in the system.
- Show command line options, corresponding explanations and usages.
- Provide tutorial help when users input incomplete commands.
- Show warning when users input invalid commands.

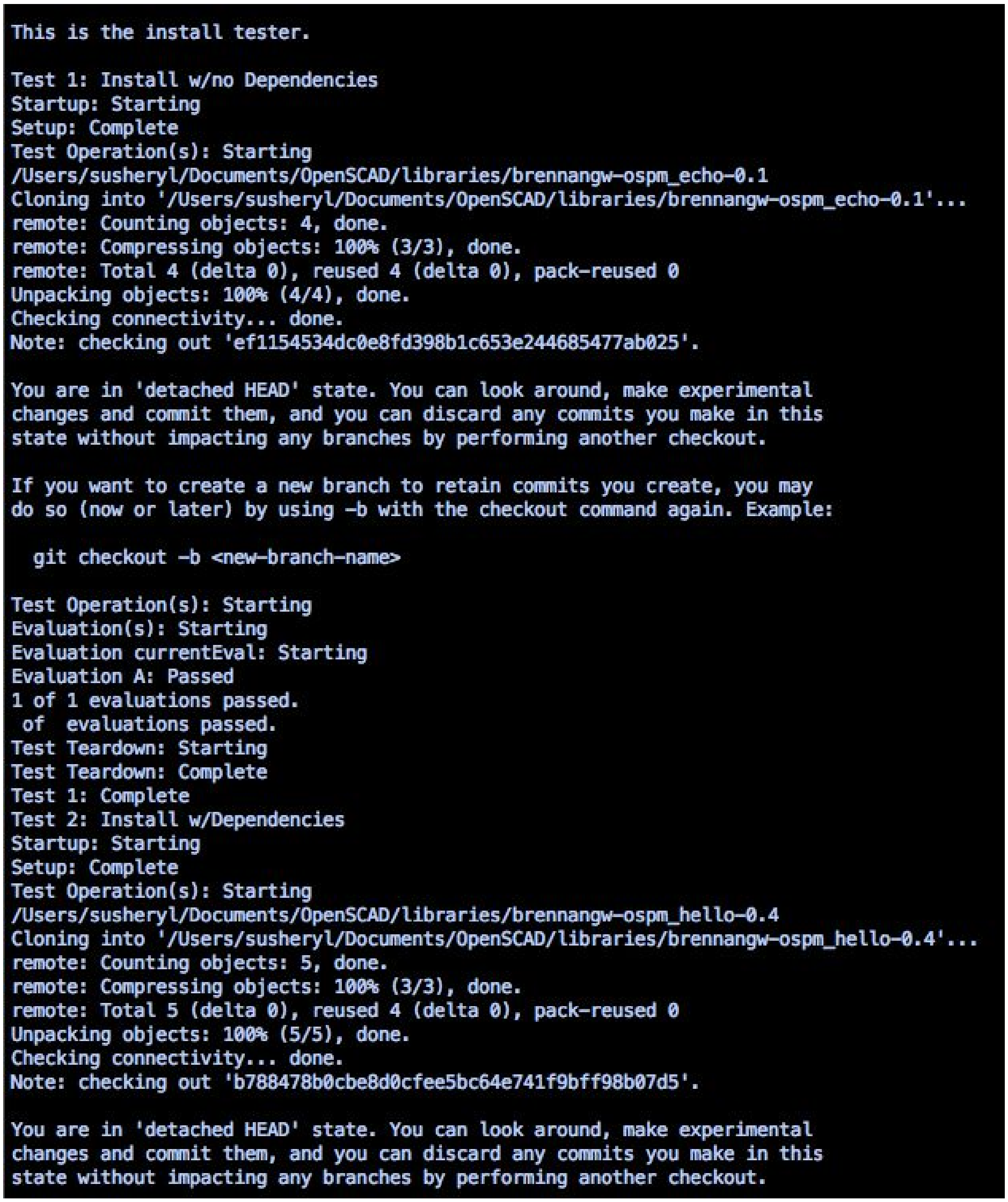
Here is an example when the CLI runs “ospm help” command.



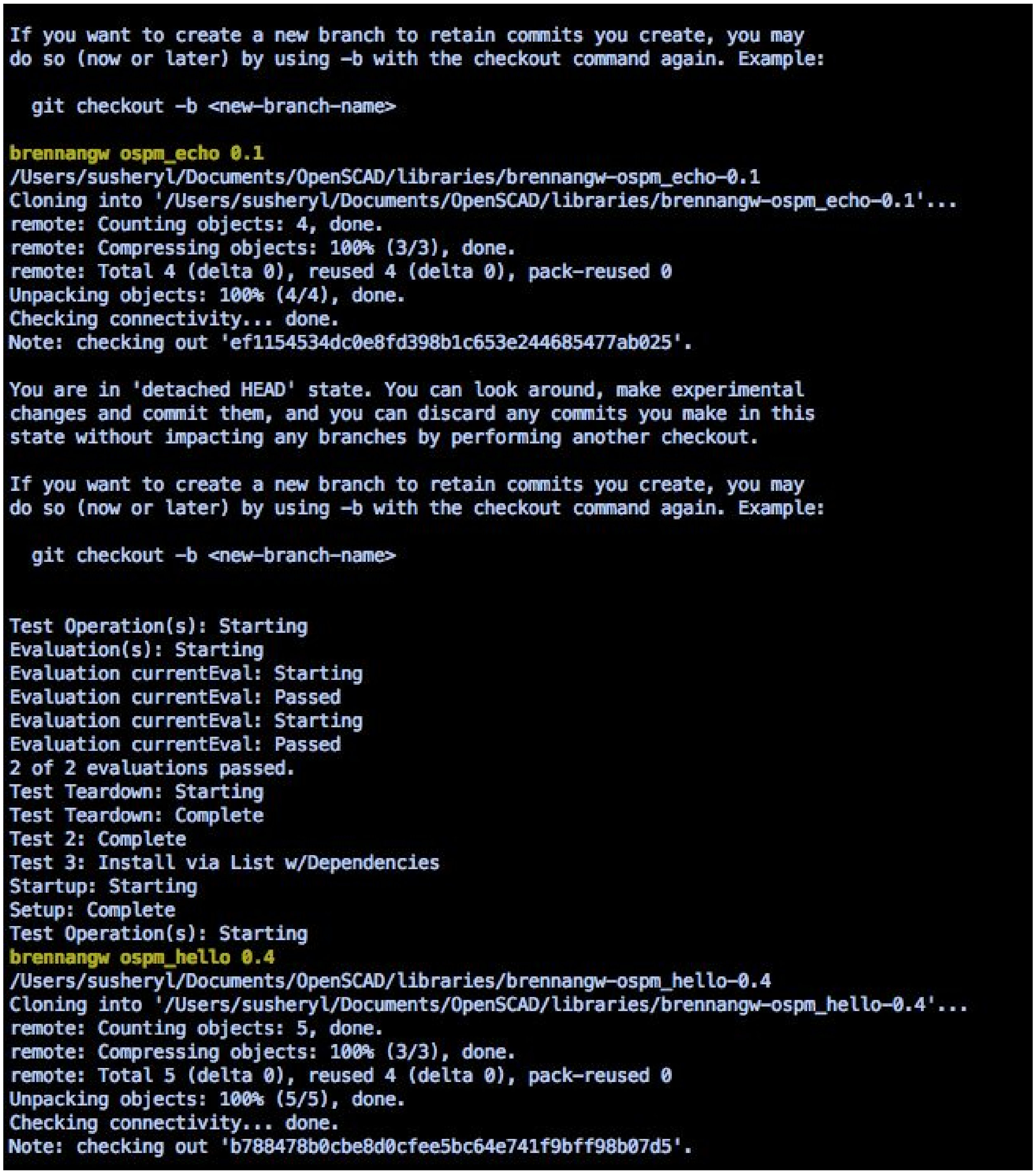
```
dyn-160-39-132-16:~ susheryl$ ospm help
Key: <variable> Usage: ospm [version] [help] [library <> <>] ...
version                                             Show version
help                                                Show command line options
library save <path>                                 Save library path
library show                                        Show library path
install <author> <package name> <version>           Install package(s)
install list <path>                                 Install package(s)
uninstall <author> <package name> <version>         Uninstall package(s)
uninstall <author> <package name> <version> force   Uninstall package(s)
parse install <input>                               Install package(s) in a file
parse save <input> <output>                         Save used packages in a file to a list
clean                                               Remove unrequired package(s)
For more information go to: skp2140.github.io/openSCADpm/
dyn-160-39-132-16:~ susheryl$
```

In addition to these two major components, we also deliver the test cases for CLI. Our test cases covers installing packages, determining library location, parsing user script, uninstalling packages, and checking version.

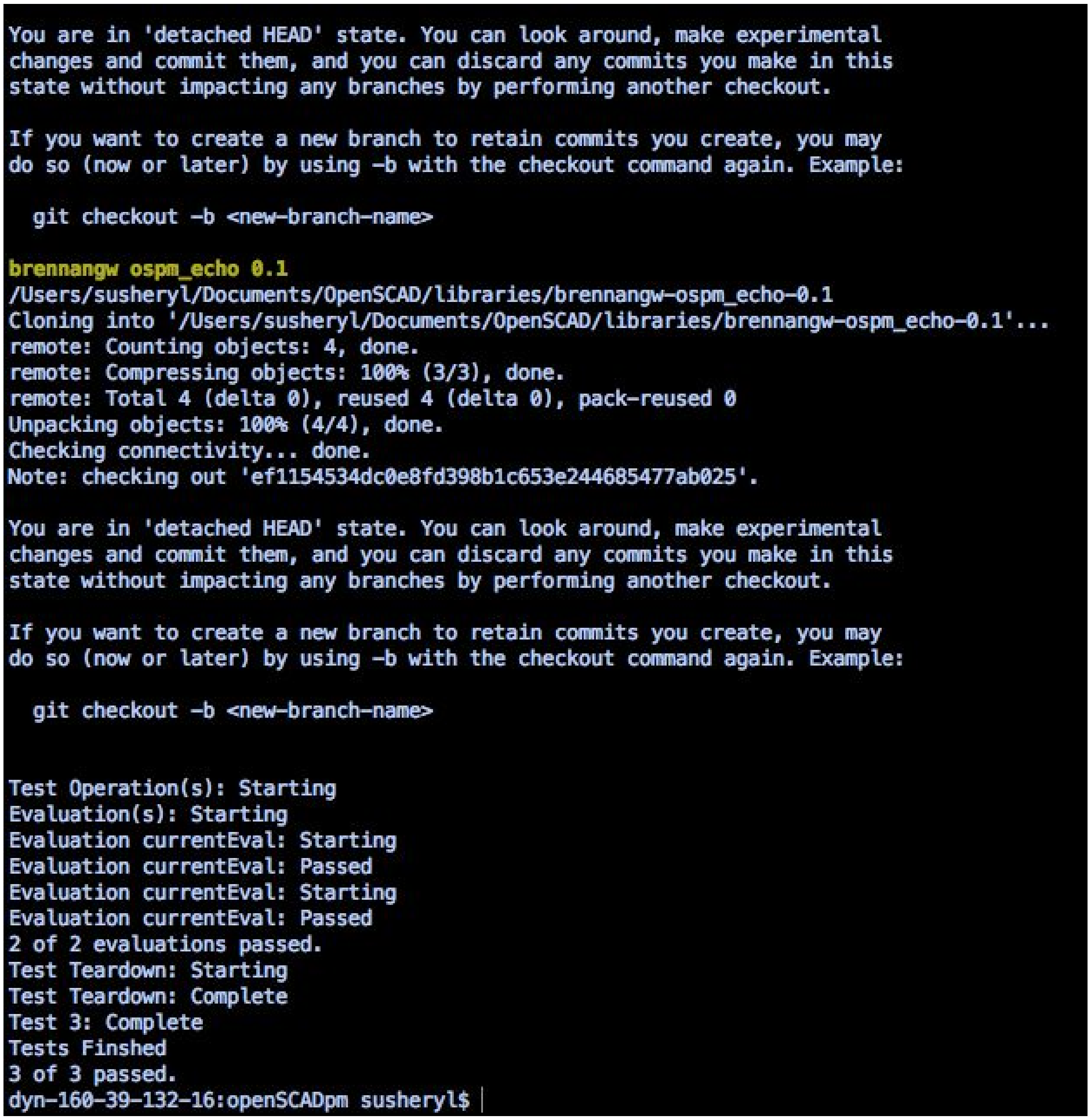
Here is an example when we runs "./installTests.sh" command.



```
This is the install tester.

Test 1: Install w/no Dependencies
Startup: Starting
Setup: Complete
Test Operation(s): Starting
/Users/susheryl/Documents/OpenSCAD/libraries/brennangw-ospm_echo-0.1
Cloning into '/Users/susheryl/Documents/OpenSCAD/libraries/brennangw-ospm_echo-0.1'...
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
Checking connectivity... done.
Note: checking out 'ef1154534dc0e8fd398b1c653e244685477ab025'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

Test Operation(s): Starting
Evaluation(s): Starting
Evaluation currentEval: Starting
Evaluation A: Passed
1 of 1 evaluations passed.
 of  evaluations passed.
Test Teardown: Starting
Test Teardown: Complete
Test 1: Complete
Test 2: Install w/Dependencies
Startup: Starting
Setup: Complete
Test Operation(s): Starting
/Users/susheryl/Documents/OpenSCAD/libraries/brennangw-ospm_hello-0.4
Cloning into '/Users/susheryl/Documents/OpenSCAD/libraries/brennangw-ospm_hello-0.4'...
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 5 (delta 0), reused 4 (delta 0), pack-reused 0
Unpacking objects: 100% (5/5), done.
Checking connectivity... done.
Note: checking out 'b788478b0cbe8d0cfee5bc64e741f9bff98b07d5'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.
```



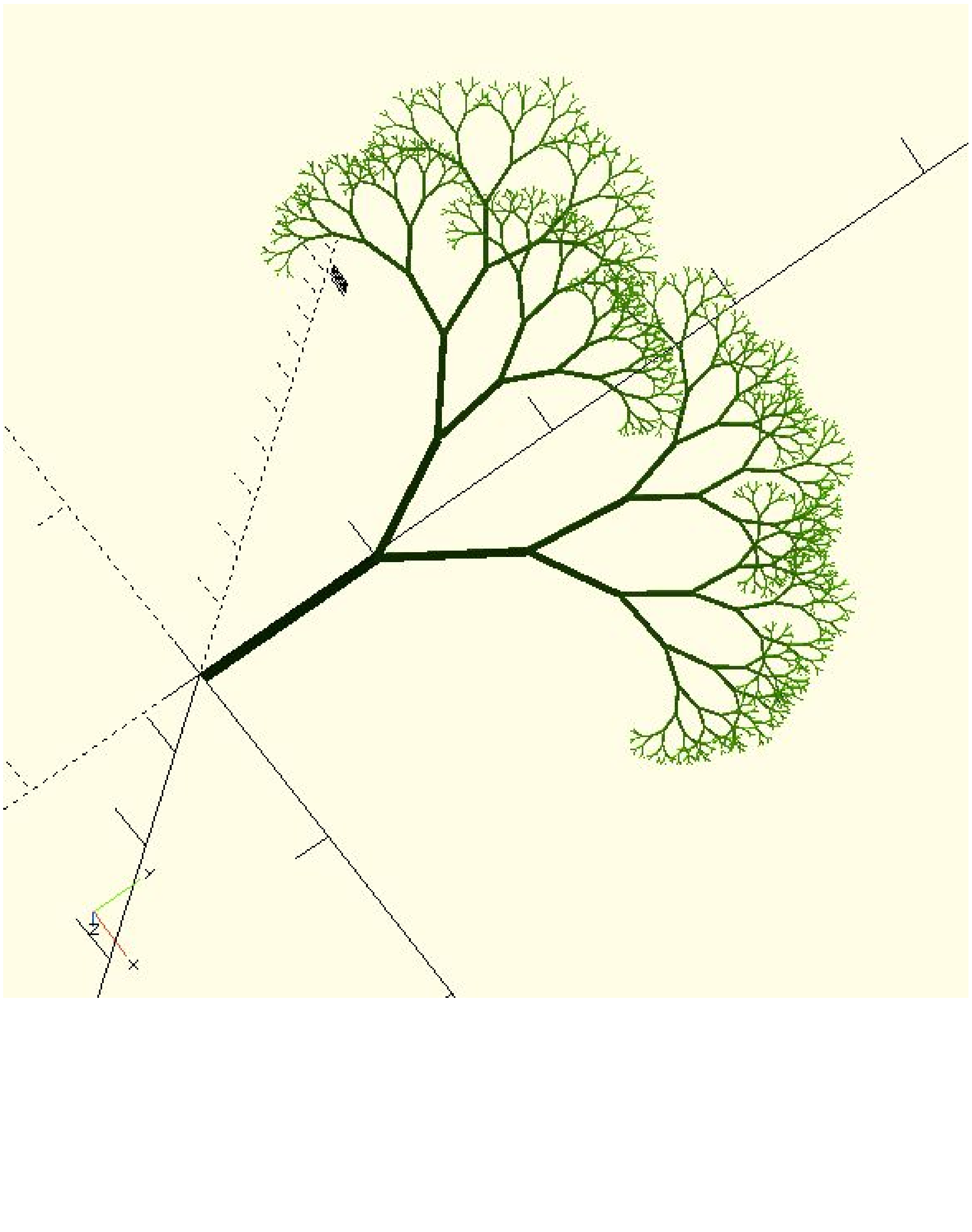
```
If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

brennangw ospm_echo 0.1
/Users/susheryl/Documents/OpenSCAD/libraries/brennangw-ospm_echo-0.1
Cloning into '/Users/susheryl/Documents/OpenSCAD/libraries/brennangw-ospm_echo-0.1'...
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
Checking connectivity... done.
Note: checking out 'ef1154534dc0e8fd398b1c653e244685477ab025'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>


Test Operation(s): Starting
Evaluation(s): Starting
Evaluation currentEval: Starting
Evaluation currentEval: Passed
Evaluation currentEval: Starting
Evaluation currentEval: Passed
2 of 2 evaluations passed.
Test Teardown: Starting
Test Teardown: Complete
Test 2: Complete
Test 3: Install via List w/Dependencies
Startup: Starting
Setup: Complete
Test Operation(s): Starting
brennangw ospm_hello 0.4
/Users/susheryl/Documents/OpenSCAD/libraries/brennangw-ospm_hello-0.4
Cloning into '/Users/susheryl/Documents/OpenSCAD/libraries/brennangw-ospm_hello-0.4'...
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 5 (delta 0), reused 4 (delta 0), pack-reused 0
Unpacking objects: 100% (5/5), done.
Checking connectivity... done.
Note: checking out 'b788478b0cbe8d0cfee5bc64e741f9bff98b07d5'.
```



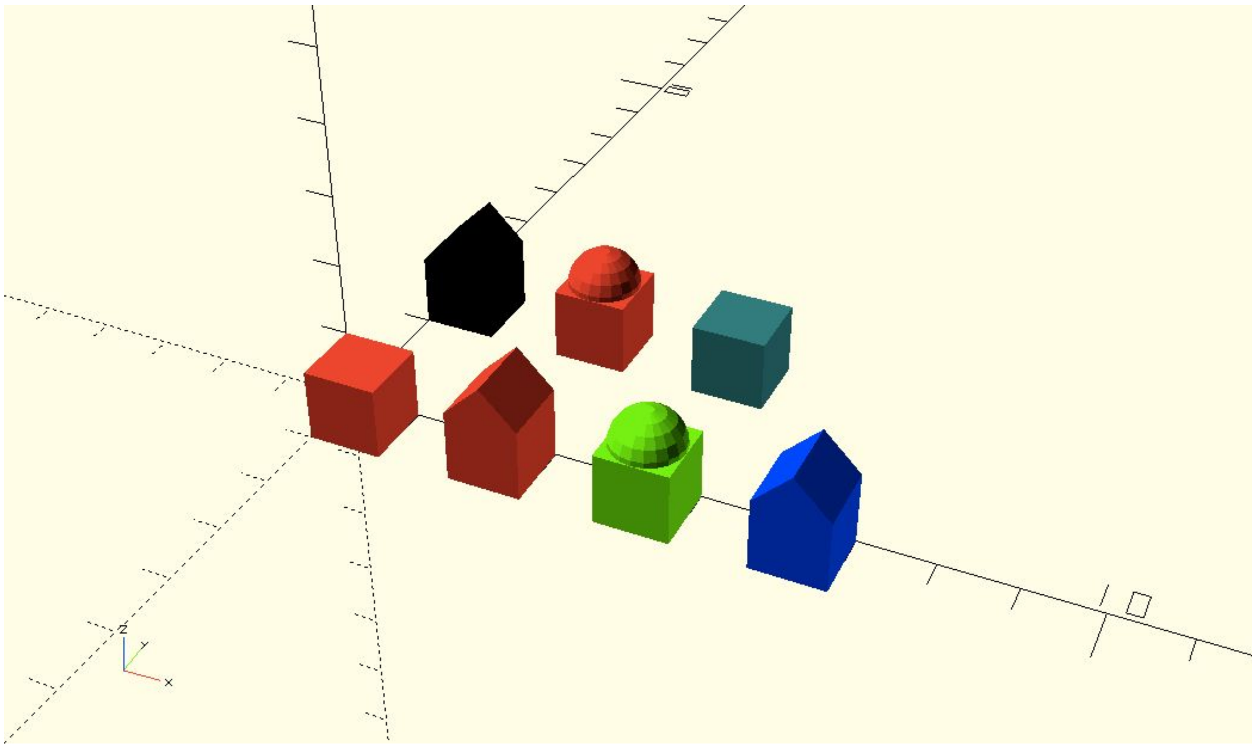
```
You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

brennangw ospm_echo 0.1
/Users/susheryl/Documents/OpenSCAD/libraries/brennangw-ospm_echo-0.1
Cloning into '/Users/susheryl/Documents/OpenSCAD/libraries/brennangw-ospm_echo-0.1'...
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
Checking connectivity... done.
Note: checking out 'ef1154534dc0e8fd398b1c653e244685477ab025'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>


Test Operation(s): Starting
Evaluation(s): Starting
Evaluation currentEval: Starting
Evaluation currentEval: Passed
Evaluation currentEval: Starting
Evaluation currentEval: Passed
2 of 2 evaluations passed.
Test Teardown: Starting
Test Teardown: Complete
Test 3: Complete
Tests Finshed
3 of 3 passed.
dyn-160-39-132-16:openSCADpm susheryl$
```

Furthermore, we created and shared the packages we created with the public. For each of the packages, it will have an index.scad file and an example.scad file. The index.scad contains the module or function codes for that specific package, and the example.scad contains an example that allows the users to know what the module provides after running it in the OpenSCAD interface. For instance, here are the examples we have for ospm_street and ospm_tree modules after we run their example files in the interface.

Here is an example of our tree module shown in the OpenSCAD interface.



Here is an example of our street module shown in the OpenSCAD interface.



In order to provide users with the best help, we also deliver documentations (wiki pages) for the project overview, how and why OpenSCADpm works, user tutorial, website re-hosting tutorial, license, and instruction about how the public can contribute and continue this project in the future.

II. Implementation Deliverables

1. How to Access

You can access the source codes for our website, CLI and corresponding CLI test cases on the following github repo:

https://github.com/skp2140/openSCADpm#contributing

You can access the website by the following link:

https://skp2140.github.io/openSCADpm/index.html

You can download the OpenSCADpm file from the website and follow the tutorial to set up the CLI in your **Macintosh or Linux machine**.

You can access the packages we created and shared by the links below:

https://github.com/jiaxinsu2016/ospm_logo

https://github.com/jiaxinsu2016/ospm_hello

https://github.com/jiaxinsu2016/ospm_street

https://github.com/jiaxinsu2016/ospm_empty_center_ball

https://github.com/jiaxinsu2016/ospm_dice

https://github.com/jiaxinsu2016/ospm_pavilion

https://github.com/jiaxinsu2016/ospm_empty_center_cube

https://github.com/jiaxinsu2016/ospm_star

https://github.com/jiaxinsu2016/ospm_tree

https://github.com/jiaxinsu2016/ospm_house

https://github.com/brennangw/ospm_echo

https://github.com/brennangw/ospm_goodbye

https://github.com/skp2140/ospm_tripod

https://github.com/skp2140/ospm_gazebo

https://github.com/skp2140/ospm_fence

You can access our documentations (wiki pages) on the following link:

https://github.com/skp2140/openSCADpm/wiki

Or wiki page for specific topics:

Project Overview
https://github.com/skp2140/openSCADpm/wiki/Project-Overview
How and Why OpenSCADpm Works
https://github.com/skp2140/openSCADpm/wiki/How-and-Why
Website Re-hosting Tutorial
https://github.com/skp2140/openSCADpm/wiki/Rehosting-The-Website
MIT Open Source License

https://github.com/skp2140/openSCADpm/wiki/License
Instruction about How the Public can Contribute and Continue the Project
https://github.com/skp2140/openSCADpm/wiki/Contributing

2. How to Set Up

2.1 Set Up OSPM CLI

This version of OpenSCADpm is only for Macintosh and Linux/Unix machines.

You can set up the CLI by following the instructions below:
1. Install the latest version of OpenSCADpm by visiting the website below and click the "DOWNLOAD OPENSCADPM" link.
https://skp2140.github.io/openSCADpm/index.html

2. A file named ospm will be downloaded.

3. Open terminal, go to **the folder that contains the "ospm" file **, and enter the following command

```
chmod +x ospm
mv ospm /usr/local/bin/
```

The “chmod” command will make the ospm file an executable and the second command will move the ospm folder to the location /usr/local/bin/ folder, which allows the users to use ospm from anywhere in the system without restriction.

4. Use the documentation below to find the path to the ospm library in your system.
https://en.wikibooks.org/wiki/OpenSCAD_User_Manual/Libraries

5. type the following in the terminal

```
ospm library save <path-to-ospm-library>
```

6. You are all set!

2.2 OSPM CLI Usages

There are two approaches to use ospm for package installing, uninstalling and other command options: User Script Approach and Package Name Approach.

- User Script Approach

You can create a user script (txt file recommended) that contains the dependencies of packages for OpenSCAD packages.
You can read the content of that txt file by the command below.

```
cat myList.txt
```

In the txt file, each row should provide the information for the needed package. The first element should be the author name, the second element should be the package name, and the last element should be the package version number. Furthermore, in this example, since ospm_hello depends on echo function, so the information for ospm_hello is one row above the information for echo.

```
<author name> <package name> <package version number>
brennangw ospm_hello 0.4
brennangw echo 0.1
```

Once you have the user script ready, type the following command in the terminal, inside the folder that contains the myList.txt.

```
ospm install list myList.txt
```

This command will help you download what is specified in txt file.

- Package Name Approach

You can also download package by using the command below.

```
ospm install <author name> <package name> <package version number>
ospm install brennangw ospm_hello 0.4
```

This sample command will install ospm_hello v0.4 in your computer.

```
ospm install <author name> <package name> latest
ospm install brennangw ospm_hello latest
```

The command above will install the latest version of the corresponding package.

- More Commands

Uninstall

```
ospm uninstall <author name> <package name> <package version number>
ospm uninstall brennangw ospm_hello 0.4
```

In each of the ospm packages, there is a required-by file, which includes the dependency information of the current package - aka, the current package depends on these extra packages, and it needs these extra packages to run. The command above will not only remove the current package information from the required-by file, but also allow the user to uninstall the package you do not want.

Clean

```
ospm clean
```

This command removes dependencies for the packages that have been uninstalled. Be more specific, the "clean" command will remove the package that has empty required-by file, which means the package is no longer needed by its original host packages, which does not exist in the system.

Library

```
ospm library <show>
ospm library <save> <path>
```

"ospm library show" shows library path, while "ospm library save " save the library path.

Version

```
ospm version
```

This command shows the version information about the ospm installed in the system.

Help

```
ospm help
```

This command shows command line options, the corresponding explanations and usages.

If the user inputs the following incomplete command in the terminal,

```
ospm
```

ospm will print out potential command options for the user.

If the user inputs commands that are not acceptable to the system, like

```
ospm ***????<or other commands>
```

ospm will print out:

```
-ospm: ***????<or other commands>: command not found
```

2.3 Rehost Website

We host our website on Github. The access method is indicated in **section II.1 How to Access** of this report. If the user would like to rehost the website, there are two steps required to re-host the website via GitHub.

1. Follow GitHub's instructions for hosting a website (index.html is the main page of the website and the css, js, img, and pages folders are necessary to run the website). The instructions can be found here. They are linked so this page does not become out of date.

2. Change the links in the website based on the new repo. Any link with "skp2140" should be changed to the corresponding new location. Otherwise, all paths in the website files are relative, and hence the transition ought to be smooth.

The website was inspired by the Nava template:

http://designhooks.com/freebies/nava-clean-responsive-html-template/

2.4 How to Run CLI Test Cases

You can download our github repo by the link in **Section II.1 How to Access**. After you unzip the pack and get into the opened folder, you can run the following commands in the terminal to run the test cases we prepared:

```
./installTests.sh

./libraryLocationTests.sh

./parserTests.sh

./uninstallTests.sh

./versionTest.sh
```

III. How We Have Delivered Our Deliverables

Live Demo at 1:15pm, Wednesday May 3rd

Our presentation covers:

- Openscad - what is the language? What is it for?
- Why is package manager needed?

- We spoke with the developers of openscad - they are interested, submitted the project to them today
- All of the project is live -
    - Github Pages
    - Github repo
    - Wiki pages with documentation and tutorial for website
    - CLI
- Website - searches - name, topics, default topics/categories
- Website download packages
- CLI Demo
    a. Install
    b. Download
    c. Show it working
    d. Parsing
    e. Cleaning
    f. Tests

Specific Procedure:
1. Download the ospm file from the website
2. Enter the following commands:

```
mv ospm.sh /usr/local/bin/
chmod 744 ospm.sh
ospm.sh install brennangw ospm_hello 0.4
ospm.sh uninstall brennangw ospm_hello 0.4
ospm.sh clean
ospm.sh version
source opsm.sh library save
```

IV. External Software Used

The CLI used Bash (pre-bash 4) and several standard *nix command line tools. The included curl, git, cut, and grep. These technologies are usually preinstalled on most machines but GNU would be a good source for downloading them. The website used CSS, HTML, and Javascript with the jQuery library and a github provided template. The website further utilized the freely available Nava template accessible via the following link:

https://www.behance.net/gallery/37366009/Nava-clean-responsive-free-HTML-template

The website was coded in the Brackets IDE on Ubuntu 16.04.

V. Presentation of Results and Findings

1. Our Findings

In terms of website, we found that static website content is more suitable for these kinds of projects as there are several options to store the content online at no cost. For example, heroku for dynamic content gives very incomprehensible links that would likely not be found by new users. Furthermore, pure HTML, CSS, and JavaScript code are easier to manipulate than complex frameworks, and hence in certain respect provide for more flexibility.

For CLI as a package manager tool, we found that bash is a much better option than python. At the very beginning of our project, we used python for the CLI. Later, we realized that even though python has good libraries for writing CLI, there is still too much to set up in terms of implementation and user environment for using our CLI tool. That's why we switched to bash after that. During our development process for CLI, we also found that a typical package manager actually just uses simple commands to manipulate and manage packages. Advanced executions might be a combination of more smaller and complicated command execution. This finding helped us to further understand the mechanism behind a package manager, which also helped us to implement our own.

For GitHub, we found that there are many tools provided such as the ability to create a Wiki and publish releases that make many parts of Websites for projects or groups of packages (like a package manager) redundant. Thus unless there is a very good reason to want to support none-github hosted packages there is little reason to reimplement the wheel as doing so is costly and leads to work that needs to be duplicated.

2.What We Learned

Jiaxin Su (js4722):
By building the CLI tool and creating ospm packages, I learnt that how a typical package manager works behind the scene. I think it is interesting to know how the package manager manipulates and manages packages just by using simple but powerful commands, and the comprehensive bash implementation at the back. I only knew simple Linux commands before I worked on this project. However, after working on this project, my bash scripting skill is drastically improved.

Brennan Wallace (bgw2119):
I learned a lot about Github and Bash scripting. I gained an appreciation for the power of Bash scripting when operating on the file system and a lot of the syntax and language features (or lack thereof). Also, I learned a lot about what Github offers its users in terms of a powerful API, wiki systems, and versioning.

Stanislav Peceny (skp2140):
First, I learned how comprehensive and useful GitHub truly is. The user is not only able to present software to the public but further can host a website for each project and create documentation with GitHub's Wiki. Hence, I learned that GitHub is much more powerful than a mere work history.

Furthermore, I learned Ruby on Rails framework as well as sharpened my knowledge of HTML, CSS, and JavaScript. I learned about the principles of package managers and primarily about npm, which served as an example to our project. Furthermore, I learned about the variety of options for deployment of software.

3. Thing we had planned to do but didn't do or didn't complete

With any CLI tool there are many features and user-interface niceties that would be nice to add but we simply didn't have the time to execute. One of the biggest was not having a more helpful error message system. We found there were simply too many possible situations and to make sure the project was complete we had to focus on adding functionality instead. More testing would have also been nice to add as there are some very complex situations that can arise.

4. Unexpected "problems" encountered

One of the unexpected problems we encountered is about the implementing the command for uninstalling packages. Some packages have dependencies, which means that when the user installs these packages, the OSPM will also install the corresponding needed packages. In the case of installing packages, that seems convenient. However, in the case of uninstalling packages, this dependency issue can be a problem: what if more than two packages depend on a few common packages? In order to solve this issue, we add a required-by field in the implementation of each of the packages. The required-by field will record if the package is required by other packages. When the CLI removes a package A that depends on package B (assume package C also depends on package B), then the required-by field of package B will be updated from 2 to 1. The CLI will only remove the package that has 0 in its required-by field. Therefore, package A will still stay in the local package library. If package C is uninstalled, then package B will have 0 in its required-by field, and it will be removed later. Overall, this issue is solved before our live demo.

Another problem we had is about hosting the website. Originally, we host our website in Ruby on Rails. Nonetheless, we realized that hosting on Ruby on Rail was not the best idea if we really want to make this project reusable and allow other developers to further develop it. Thus, in order to solve this problem, we managed to migrate the entire website from Ruby on Rails to the Github pages, where users can host their websites for free and in a long period of time. This issue is also solved before our live demo.

Due to this migration issue, we have two github repos for this project, and both of them should contain similar source codes.

Latest Github Repo: https://github.com/skp2140/openSCADpm

Older Github Repo: https://github.com/brennangw/openSCADpm

5. Who Did What

Jiaxin Su (js4722):

- Took the lead on the CLI tool, and implemented various command options
- Digged into other CLI bash projects to see if there are helpful resource for “ospm install” & “ospm uninstall”
- Created OpenSCAD example packages
- Wrote test cases for the CLI tool
- Wrote documentations and reports (Took the lead - main force)

Brennan Wallace (bgw2119):
- Rocking it on the language specific portion of the packaging methodology.
- Worked on CLI command options and helped connect the CLI interface with the website
- Advised the website for rehosting, and migrating the site from rails to github pages.
- Wrote test cases for the CLI tool
- Wrote documentations and reports

Stanislav Peceny (skp2140):
- Implemented the frontend and backend of the website in Ruby on Rails. The app is connected with a database wherein all users and packages are stored. Users can set up accounts, create packages, search for packages by package name, author name, topic, categories, descriptions, and more.
- Migrated the rails app to a Github page app, which has free hosting service in the long run
- Transformed the website entirely to a new static page
- Wrote documentations and reports

6. Questions or Concerns from Teaching Staff With Respect To Previous Project Demo

In our previous demo, Professor Kaiser suggested us to do the following adjustments:

- Release documentation for rehosting the website.
- Follow command line conventions even more
    - Instead of only having “Command not found” message when the user inputs invalid command, the CLI should display that it is OSPM CLI.
    - Add extra command options when the user input incomplete commands.
    - Set default package version and further simplify install command.
- Enable the website to search by package author name, since sometimes users have this need.

Based on Professor Kaiser’s suggestions, we added command line tool label like “ospm: <user_input_invalid_command>: ” in the CLI implementation. Hence, when the user inputs invalid command like “#$%^&*(*&^%”, ospm CLI will display “ospm: #$%^&*(*&^%: Command Not Found” error message, rather than just plain “Command Not Found.” We also added the extra command options for inputting incomplete command. After that, we worked on the CLI implementation so that the CLI tool will install the latest package by default. This step further simplifies our command options, allowing users to install packages in an easier way.

In order to provide users with the best help, we decided to have even more detailed documentations that cover the project overview, the user tutorial, how and why OpenSCADpm works, website re-hosting tutorial, MIT open source license, and instruction about how the public can contribute and continue this project in the future.

VI. Documentations Access

You can access the **user tutorial** for the website and the CLI by the link below, which is also our README.md:
https://github.com/skp2140/openSCADpm

In addition to user tutorial, we also have wiki pages for the project overview, how and why OpenSCADpm works, website re-hosting tutorial, MIT open source license, and instruction about how the public can contribute and continue this project in the future.

You can access them by the links below:

- **Project Overview**
  - https://github.com/skp2140/openSCADpm/wiki/Project-Overview

- **How and Why OpenSCADpm Works**
  - https://github.com/skp2140/openSCADpm/wiki/How-and-Why

- **Website Re-hosting Tutorial**
  - https://github.com/skp2140/openSCADpm/wiki/Rehosting-The-Website

- **MIT Open Source License**
  - https://github.com/skp2140/openSCADpm/wiki/License

- **Instruction about How the Public can Contribute and Continue the Project**
  - https://github.com/skp2140/openSCADpm/wiki/Contributing