Sprint 2 - Writeup

Behnam Saeedi, Margaux Masson, Kamilla Aslami, Nghi Duong, Dhruv Jawalkar
CS561: Software Engineering Methods
Due Nov 12th 11:59pm
Fall 2018
Repository: https://github.com/BeNsAeI/PADSR

CONTENTS

1 INTRODUCTION

This documentation is for the second stage of our project Sprint 2. In this sprint we continued on adding more features for the PAD-SR API. For this sprint we selected 4 new user stories to implement. Our repository is available at the following link(https://github.com/BeNsAeI/PADSR). In this sprint we dive deep into world of computer graphics and break the barrier between 2D videos and 3D scenes. The following documentation is the required documentation for the second sprint of this course.

2 GITHUB REPOSITORY INSTRUCTIONS

2.1 Cloning

The repository is located at "https://github.com/BeNsAeI/PADSR" and contains many different folders. The files that we developed for the purpose of this project are all located in the Integration folder. Furthermore, We can find all of our documentation in the Documentation folder. We also have few other folders in the repository that are meant for testing, development and figuring out what is achievable. Likewise, in the documentation folder we have a folder for user stories. The User stories for this sprint and potential user stories for any future development are located in "Documentation/User stories/PAD".

2.2 Trying the code:

The code intended for this class is in the "PATH/Integration" folder. You can try this sprints code by typing the following to lunch our UI:

```
python CLI.py
```

Please Note: There are several python packages that need to be installed for our UI to work properly, The following commands will install such packages:

```
sudo pip install Pympler
sudo apt update
sudo pip install hg+http://bitbucket.org/pygame/pygame
sudo pip install pyyaml
sudo pip install Pillow
sudo pip install ftputil
sudo pip install svn+https://svn.code.sf.net/p/python-xlib/code/trunk/
sudo pip install PyOpenGL PyOpenGL_accelerate
sudo apt install python-opengl
sudo apt install python-pygame
sudo easy_install PyOpenGL
sudo easy_install -f . PyOpenGL
sudo easy_install -U PyOpenGL
sudo apt install python-numpy
    python-scipy python-matplotlib
    ipython ipython-notebook
    python-pandas python-sympy
    python-nose python-sklearn
    python-opencv
sudo pip install --upgrade pip
```

```
sudo pip install numpy
sudo pip install scipy
sudo pip install matplotlib
sudo pip install opencv-python
sudo pip install opencv-contrib-python
sudo pip install -U scikit-learn
sudo pip install prompt_toolkit
```

3 MEET THE TEAM

The following is the least of our group members and their expertise and background.

- **Behnam Saeedi:** Behnam Saeedi is a master student with AI and computer graphics options. His skills are mainly in embedded programming, machine learning, image processing and computer graphics.
- **Margaux Masson (Scrum Master):** Margaux Masson is an exchange Computer Science's student at OSU from France. Her school back in France is CPE Lyon Engineering school. Her major is Computer Vision, so all the fields related to image processing and computer graphics. In OSU, she is taking the Computer Graphics, Cyber Security and Software Development Methods' courses.
- **Nghi Duong (Product Owner):** Nghi is a new master student concentrating computer vision and 3D modeling. His background is in mathematics and computer vision.
- **Kamilla Aslami:** Kamilla is a first year Master's student participating in the Software Innovation Track. She is interested in Machine Learning and has used Python for 3 years in a professional setting.
- **Dhruv Jawalkar:** Dhruv is a master's student at OSU with an interest in Computer Vision, has experience in programming in Python, training neural nets to do simple vision tasks like Classification, Localization and Object Detection.

4 USER STORIES

Here is the list of User stories which we attempted to accomplish. Being more confident in our capabilities this time we budgeted more of our 105 hours of time into the project. This time around we allocated all the time we though we might need for solving issues and fixing bugs into the allocated estimated time and we managed to successfully carry out the project in the correct budgeted time.
In this section we will cover our user stories and also comment on why we decided such specifications, priorities and estimated completion times for each one. Please note, our actual user stories are concise and do not include in depth explanations on them. These user stories could be viewed on the "PAD-SR GitHub: Documentation/User stories/ PAD/" path. In this write up, texts that are written in **bold** are the texts that are in our user story 5×3 cards.

4.1 Spike (Process requirement):

Take time to learn and setup openGL

This step was added to make all of our team members setup and get familiar with OpenGL using a first simple project.

4.1.1 Priority

High: Next 3 User stories depend on this task.

4.1.2 Estimated time

25: Setting up testing and learning OpenGL is very time consuming and all members need to do this.

4.1.3 Details

OpenGL requires a setup time, furthermore, some of the members need to familiarize themselves with the environment in which we will be developing in.

4.1.4 Tasks

- Set up openGL environment
- Run and study OpenCV examples (2)
- Create the first 3D scene and draw the first 3D object (2)

4.2 Make a 3d video (Functional, Product, Emergent requirement):

Use tools to take the depthmap stream and the full video stream and create a 3D video

4.2.1 Priority

Low: Nothing else in our system depends on this feature.

4.2.2 Estimated time

20: This is a challenging task however, the code is not so complicated to implement.

4.2.3 Details

Use OpenGL or thirdparty software to make a 3d movie using the depthmap

4.2.4 Tasks

- Unit test (5)
- Boilerplate (5)
- create a vertex shader that takes in depth map stream and uses it to set height (5)
- create a fragment shader that takes in the video stream (5)

4.3 Embedded platform (Functional, Product, Derived, System requirement):

Move the program to the AI embedded platform and test the performance

4.3.1 Priority

Low: Nothing else in our system depends on this feature.

4.3.2 Estimated time

25: Import all the libraries and test the code can be time consuming because some libraries need to be well imported and if they are not, it can take some time to debug it and install the right modules.

4.3.3 Details

The implemented API needs to be compatible with Nvidia Jetson TX2.

This might seem trivial however, the issue is that many libraries that we are looking for and using are simply not compiled for the particular ARM architecture that Nvidia Jetson TX2 uses. We need to make sure that the ones we need are supported on aptitude and pip or be compiled. Furthermore, the Performance will be improved by installing the GPU compliant version of some of the python packages we are using.

4.3.4 Tasks

- Import everything (1)
- Install all the libraries (1)
- Modify platform-dependent code (3)
- Unit tests (10)
- Performance tests (10)

4.4 UI Code integration API (Derived, System, Emergent, Process requirement):

Collect all of the functions into one program and check conflicts in a UI

4.4.1 Priority

Medium: This user story is actually an important feature since the user will use the system through this API.

4.4.2 Estimated time

35: Designing and implementing the terminal user interface and the API can be challenging.

4.4.3 Details

Make sure the python codes have no conflicting elements. Put everything together and rerun the test cases in a UI

4.4.4 Tasks

- Unit test (5)
- Import each function into a python file (10)
- Create API function calls (10)
- test the API (5)
- look for potential optimization (5)

5 INTEGRATION TEST PROCEDURE:

The following tests are all performed as one integration test. However, each test in our integration procedure aims to address one or more of the user stories working together as one complete software. For this particular set of user stories we simply could not implement an automated integration test. The reason to this issue is that certain aspects of our tool were visually dependant on the configuration and behavior of the UI. Therefore, we had to perform a manual integration test to assure that the behavior of the API is not different when launched standalone versus launched through our user interface.

5.1 User story 1 Integration: Spike

The user story 1 was a spike to setup OpenGL and did not require any integration.

5.2 User story 2 Integration: Make a 3d video

The following procedure is for testing the depth map to 3D scene feature of the tool.

- Task: Import the API
  Criteria:
  - The API needs to be imported without any errors
  - There shouldn't be any overlapping global variables outside of the scope
- Task: UI call to 3D Scene generator
  Criteria:
  - The Integrated tool shall make call to the 3D scene generator.
  - The software needs to run by the UI call and start asking for 3D scene generator parameters
  - The UI needs to resume or terminate after the execution of the API call is done.
- Task: Clean Up
  Criteria:
  - Check memory after completion and performance to make sure all heap variables are returned to the memory
  - Look for memory leaks and missing object destruction (Test code available in our repository)

5.3 User story 3 Integration: Embedded platform

- Task: Run the integration test on the embedded platform
  Criteria:
  - Same integration test needs to be successful for all attributes running on our embedded system Nvidia Jetson TX2.

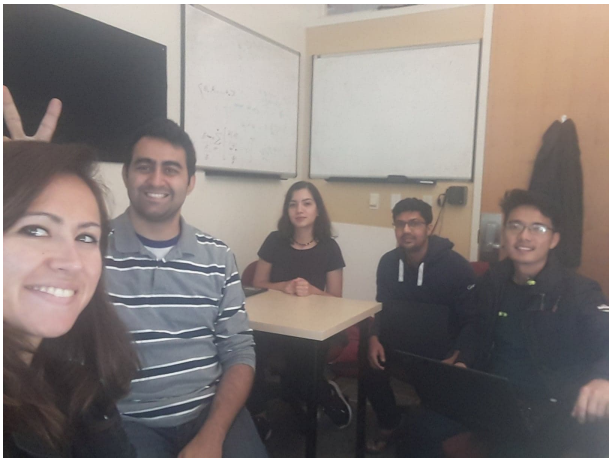
5.4 User story 4 Integration: UI Code integration API

- Task: import all API methods
  Criteria:
  - All methods need to import without any errors
  - Check memory for any memory usage and leaks
- Task: UI Call to Video to Depth map
  Criteria:
  - Make UI call Depth map function without any errors
  - Program not crash during receiving configuration input
  - UI recover after video to depth map function is done
  - Check for memory leaks
  - Check for all objects from video to depth map to be destroyed and cleaned up
  - UI prompt choices for next procedure
- Task: UI Call to Video to 3D scene
  Criteria:
  - Make UI call Video to 3D scene function without any errors
  - Program not crash during receiving configuration input
  - UI recover after Video to 3D scene function is done
  - Check for memory leaks
  - Check for all objects from Video to 3D scene to be destroyed and cleaned up
  - UI exit after window closes

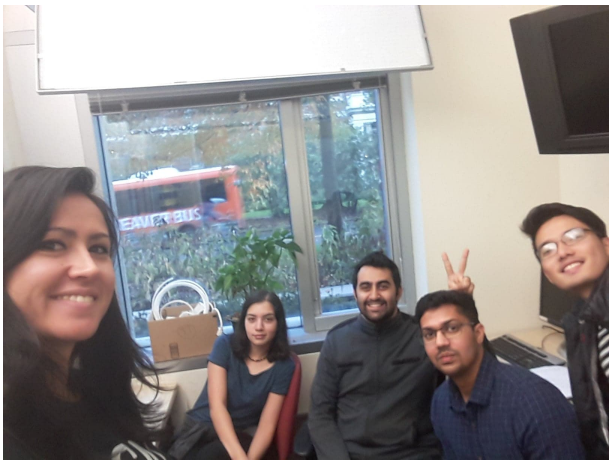
6 Daily Scrum Photo Gallery

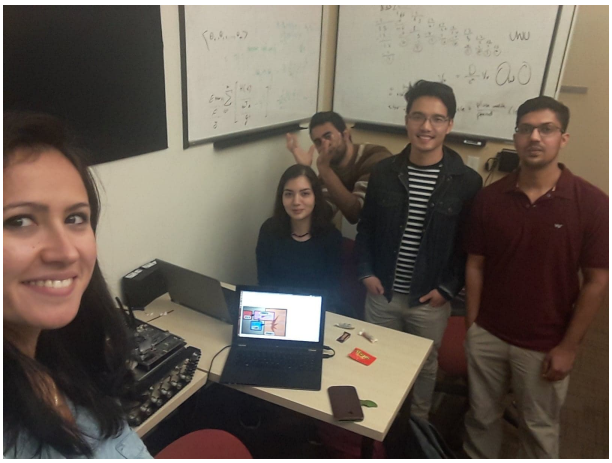
6.1 Week 1


6.2 Week 2



6.3 Week 3





7 GitHub Integration evidence

For more information on integration test and procedure, please see section 5.

```
4 hours ago : cleaned up, user test done and bugs fixed
4 hours ago : Integration test performed
22 hours ago : Finished integration. Updated unit tests. Re-factoring
6 days ago : Added unit tests for cli_helpers
6 days ago : Fixed bugs after integration testing
...
6 days ago : [WIP] Added cli helpers file and unit tests
...
30 hours ago : modded function added pre integration, unit test to follow
```

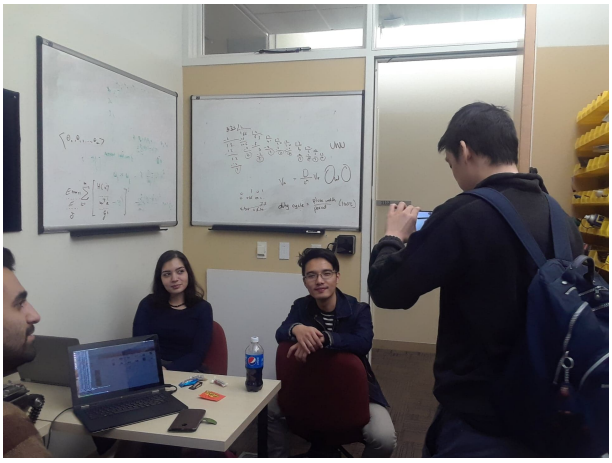
```
30 hours ago : modded function added pre integration, unit test to follow
...
4 days ago : Testings on python opengl. The depth GL with depth map visualization is in place
...
6 days ago : new example in python added with scene setup and camera control, Unit test added
7 days ago : added monitoring plots for gpu, cpu and memory usage
...
10 days ago : Merge branch 'master' of https://github.com/BeNsAeI/PADSR
...
2 weeks ago : Added CLI examples
...
3 weeks ago : Added images, files integrated, Tested
3 weeks ago :  Added new unit tests and fixed integration tests for depth map functions
...
```

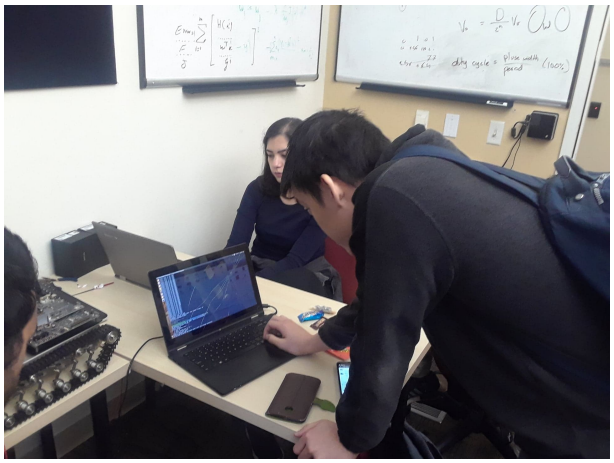
8 Sprint Review Logs and Notes

For this Sprint review we had 4 users use our system. Our 4 users tested our software in 2 ways. Each user was given the following instructions:

- Take a video with an android device
- Transfer the video to a computer
- Launch the application
- Follow the onscreen instructions

8.1 User Tests





- **User story 1: Create a 3D video:**
  All 4 users managed to produce 3d Scene
  - As mentioned earlier, the users were only given 4 simple instructions.
  - **What went well:** All 4 users successfully managed to convert their videos into a 3D scene. Users were very excited about the results since the result is visually appealing and the user understands what they are looking at.
  - **What went wrong:** Certain configurations were resulting in sub optimal image results and the UI instructions did not communicate the effect of those configurations very well. This was improved on their second attempt.
  - **Comments:** Overall the users enjoyed the tool and were excited to see more of it.
- **User story 2: Embedded platform:**
  This section is not intended to be used by users. However, the operating procedure is exactly the same as the testing on any other system. The unit and optimization tests ran on the embedded system guaranteed that the system performs identically on the embedded system to any other computer. The embedded system too connects to an HDMI monitor and has normal keyboard and mouse attached. From a user standpoint, there is no identifiable difference in operation of the tool.
- **User story 3: UI Code integration API:**
  - A good UI fills the gap between user and the API and allows the user to have a seamless experience.
  - **What went well:** All instructions in the UI were clear enough for users to understand. They had a chance to test the program with default and custom settings to see how these options affect output.
  - **What went wrong:** Some users were not comfortable with the command line, and thus we had to provide additional instructions on how to work with command-line interface. Nevertheless, we had expected that and therefore simplified the UI as much as possible by providing default values for all settings.
  - **Comments:** The target audience of our project are engineers, so we expect them to be comfortable enough with the command-line interface. Therefore, we will stick to the text-based UI but will try to find more ways to simplify user interaction.

8.2 Product owner

Nghi Duong was the product owner for this sprint. He is in charge of creating and recording this sprint backblog. He also gets the feedback from users at the end of this sprint.

8.3 Scrum Master

Margaux Masson was the scrum master for this sprint. During this sprint she was in charge of the daily meetings' hours, get to know if all the members were able to come, take pictures to have evidence of the meetings' attendance, make sure that every member of the team had something to work on and was happy about it.

8.4 Attendees

The attendees were limited to the minimum number of people necessary in order to have an effective Sprint review.

- Developers: All the members of the team were present during this meeting.
- Product owner
- Scrum master
- User

8.5 System strengths and weaknesses

8.5.1 Strengths

- The program execution is fast enough to create a smooth 3D scene.
- The User interface is easy to use for those who are comfortable with terminal.
- The system is actually converting a 2D video into a 3D scene and satisfies our claims.

8.5.2 Weaknesses

- The 3D video can be smoother by removing some noise.
- Difficult to cover with automatic tests.
- The UI is not designed for those users who is not familiar with a command-line interface.

8.6 Evidence

9 Sprint Retrospective Notes

9.1 Notes

What went well:

- **Timing:** This sprint we used our time very efficiently and effectively. Overall we managed to stay true to the timing we allocated for our procedures.
- **Meetings' time:** This Sprint we nailed one of our weaknesses from last sprint which was bad scrum meeting times. This sprint we managed to organize the timing of the meetings a lot more effectively. Team showed up on time and the scrum meetings were consistent and useful.
- **Project's dynamic:** Better distribution of the work. We made sure that the work load this sprint was very effectively distributed based on each member's background. Our group happens to have a very diverse set of skills for each group member and we used this in our advantage.
- **Quick adaptation and fast learning:** Some of the tools we used such as OpenGL were somewhat new to some of the group members. The group was very efficient in bringing the members that were not familiar to the tools up to speed.
- **Smoother integration:** Earlier integration of the code comparing to the previous sprint.

Things to improve:

- **Group Communication:** Better communication when working in pairs. We did have a few pair programming sessions. However, one thing that we forgot was that the pair might not have been as fluent as the pilot. In this case we need to improve this by allowing the less fluent programmer to be the pilot first.

Furthermore, we had an incident when two of the members implemented the same feature separately.

- **Should improve the sprint's planning and the setup time planning:** One of the things that perhaps was not as good as we wanted to be was the assignment of certain user stories to more than one person. We would like to work on our sprint planning a bit better in order to avoid these cases and also adjust the pair dynamics in our development procedure.

- **Testing:** More integration tests - we had some issues with the automatic tests and we had to do the tests manually. We would like to figure away to automate integration testing and not have to do it manually again.

10 Sprint Retrospective Evidence

Please note we did our Sprint retrospective immediately after our last scrum for this sprint.

11 Contribution Bus

Similarly to last sprint we tried our best to make sure non of the group members are overworked. This sprint all members managed to contribute equally to the project. Each member had new things to learn, tough challenges to face and complicated problems to solve. By the end each member contributed no more or no less than the time budget they had. In general all members of the team contributed similarly to the project. Overall we can say that the contribution of each member was about 20% to the overall project.

11.1 Margaux Masson

Margaux was responsible with OpenGL platform. OpenGL often has a very messy API and it is very difficult to setup the scene properly. Often hours of coding and debugging leads to the "OpenGL's Magic black screen". Margaux set up the OpenGL pipeline, graphic states and scene posting. The result of her work was OpenGL Dynamically generate 307200 different vertices in the correct order and position. Furthermore, she setup the GL virtual object list (VBLs) in order to significantly speed up the rendering procedure. She was also responsible for setting up the multi-texture coordinates for every single vertex.

11.2 Behnam Saeedi

Ben was in charge of handling and producing the shaders. Shaders are pieces of code that run on GPU in order to produce process and compile images. Ben had to set up the graphics pipeline is such a way that the OpenGL platform which Margaux wrote is compatible with GLSL (Graphic language's shares language). This was achieved by using two of the 6 shaders available in Kronos drivers: Vertex shader and Fragment Shader. Furthermore, Ben had to make sure that the video frames get properly posted to graphics memory in order to be used by different Shaders.

11.3 Nghi Duong

Nghi worked on application program interface. He created a set of API functions that are called in the 3D video application and in UI. Nghi was also in charge of integration testing. He wrote an integration test plan and performed integration testing using this test plan.

11.4 Kamilla Aslami

Kamilla worked on a user interface. As it was decided on iteration planning, UI was implemented as a command-line interface with two commands - create a depthmap video and show a 3d scene. Both features can be adjusted with different options available to a user. Kamilla was also responsible for maintaining old unit-tests, developing new ones and integrating all the components.

11.5 Dhruv Jawalkar

Dhruv worked on the Embedded platform task, developed a tool to monitor resource usage (GPU, CPU and Memory) when performing a task. In our project we wanted to monitor how much resources our code was using when performing tasks like, generating depth map from pair of images, creating 3D video from video, its depth map footage and also determining the direction of camera movement in a video. Tool helps with that.