# Molecular Mission - 3D Molecular Visualization Game

### The RIsoTopes

|  |  |  |
| --- | --- | --- |
| Andrew Popovich |  | Christopher Timmons |
| Forrest Shooster (GD&D) |  | Justin Cotner |

### Project Sponsor

### Joseph Lanzafame

### Faculty Coach

Al Arujunan

## Project Overview

This project involves creating a 3D molecule visualization game for chemistry students to link their in-class studies with a virtual representation. This is meant to help the students understand how the molecules are supposed to look and how they interact with other molecules and forces. The concepts taught in the students’ classes will be reinforced with this game.

This project continued the work of the last year’s senior project team and expanded upon the gameplay in the game. In addition, this project aims to expand upon the basic server structure in place so that chemistry instructors can gain a better understanding as to how students are progressing in the game and how that progress correlates to the students’ understanding of specific chemistry concepts. The top two priorities in this project were playability and realistic modeling. These priorities were properly balanced during the development of this project since the team showed off the game to the sponsor after each iteration, and showcased how the playability and realism changed from previous iterations.

## Basic Requirements

### Game Requirements

* Student Learning
  + The game has 3D molecule models for students to view.
    - This is the most basic and important requirement. The game will have the 3D molecules that will allow students to see the molecules and understand them better than if they viewed them in a 2D space.
  + The game has models of multiple molecules for students to be able to distinguish
    - The game will not only have the models for a couple select molecules, but it will contain the models for many different molecules that exist in nature. Students will be able to view many of the molecules they would be exposed to in their studies.
  + The game provides information on molecular bonding geometries
    - Since this game is an extension of the students’ studies, the game needs to have the same information that the class would have. The game will show the angles of the bonds, what they are called, and how/why they are formed.
* Tasks and Goals
  + The game provides an intriguing story for students to follow
    - Students want to be entertained while they learn to break up the monotony that school can provide. The story in the game will keep students’ interest and distinguish this game from a learning tool.
  + The game has goals and tasks to complete that act as incentives for students to want to play the game
    - The game has to have goals to keep students occupied. Students will keep playing the game if they have something to work toward.

### Server Side Requirements

* Game API
  + The server side implements an API for the game to add stats from the game to the server’s database
    - The game will generate statistics for the students as they progress. The stats will be sent to the API so that students can log onto their accounts and view their statistics.
  + The API is simplistic so that the game can easily call it
    - To ease the transfer of statistics, the API is simple enough to receive, send, and store data without creating unnecessary complications.
  + The API is also easily extensible so that new stats can easily be added
    - As the game expands, there will be opportunities to add new statistics into the game. To help with this process, the API is made simple enough with room to grow whenever a new statistic is created.
* Student Accounts
  + The server enables students to view their own stats
    - Students can log into the server side of the project to view their game statistics. They can see how well they are performing and maybe compare themselves to other students for competitive purposes.
  + The server allows students to download the game executable
    - The game runs through a stand-alone executable that can be run on multiple machines. The server will have the different versions for students to download and run on their home computers.
* Instructor Accounts
  + The server allows instructors to create an account via a self-signup feature
    - Instructors will also have their own accounts on the server. They will be given a key that allows them to create their own accounts.
  + The server allows instructors to view students’ stats that generated from the game
    - Instructors would like to view how well the students are completing the game, so when they log on, they can see all of their students’ statistics and progress. They can then alter their lesson plans or learning goals based on this information.

## Constraints

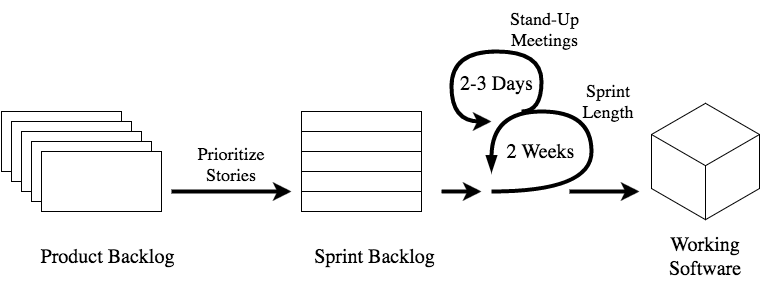
We were somewhat constrained by the technology choices of last years team. We could have changed technologies but that would have required more or less starting from scratch. Since we wanted to build on the progress already made we stuck with the technologies chosen by last years team. For the game that meant using the Unity Game Engine (specifically Unity 5). On the server side that meant using Node.js and MongoDB.

We were also ran into some resource constraints, specifically on time and personnel. Second semester was especially bad for finding a time when everyone was available to meet. The whole team also had other classes that took up quite a bit of time which slowed the pace of development. We were also unable to add a designer to the team which resulted in the new game objects being very basic visually. Also, our game design student did not actually enroll in the class second semester. As a result, he still helped us out but it was not his biggest priority.

## Development Process

Our decision on a process first required a bit of time to do some research, in which we also spent that time on formulating and finalizing our requirements. One of our main considerations for a process methodology was that the sponsor had given us almost complete control over the direction and implementation of the project. As a result, we actually didn’t have a formal set of requirements from the sponsor. Therefore, during our research we leaned towards an agile process methodology mainly because our requirements were not strict and could potentially change over time depending on our sponsor’s reaction to the work that we did.

### Scrum Process Diagram



Therefore, at the end of this phase we decided to use Scrum so that we could show off our work to our sponsor at the end of every sprint and receive feedback for the next sprint. This enabled us to meet with our sponsor relatively often and allowed us to address any concerns that our sponsor brought up in a timely fashion. Finally, the main process-based role that we used was a communications coordinator and this person was in charge of planning meetings and organizing the communication between the sponsor and the team. Beyond this, the team implementation roles were distinguished by who would be working on the game and who would be working on the server. These roles were mainly determined by who had the most experience working with the Unity game engine in the team.

## Project Schedule: Planned and Actual

### Planned Schedule

Our planned schedule was based around the Scrum process methodology. Therefore, we planned on having 2 week sprints, in which we would plan milestones to accomplish before the end of each sprint. In addition, we also planned to meet as a team 2-3 times a week to have project update meetings. In terms of the number of sprints that were planned, we planned on having 4 sprints during the fall semester, and either 4 or 5 sprints during the spring semester depending on the progress that was made. This plan gave us time to prepare for our interim presentation in the fall as well as gave us some time to prepare for the poster and final presentations in the spring.

### Actual Schedule

Our actual schedule slightly varied from our plan due to user stories that were planned during a sprint were not completed and therefore had to be carried over to the next sprint. Therefore, we accounted for these issues by scaling back our next sprint to focus on finishing up the user stories that were carried over. In addition, we also needed to have 5 sprints for our spring semester, on top having a period of time for general project integration and debugging. This additional period of time was not accounted for in our project plan, but was determined to be necessary since our user stories did not account for integrating the communication between the game and the server side of the project. Therefore, as a result we were not able to complete another level, which was a feature that our sponsor wanted but was a feature that we viewed as a stretch goal. In addition, this additional period of time also went into our time to prepare for the poster and final presentation, but this was not detrimental to our performance for those presentations.

## System Design

### Architectural Design



Our high-level system architecture is a client-server architecture that has been modified to accommodate two different types of the clients, the Website Client and the Game Client. Prior to our work on the project, this high-level architecture was slightly different since the Game Client was actually embedded within the Website Client. As a part of some research we had performed early in the fall semester of our project, we determined that the original high-level architecture was not ideal for the sponsor or for future students using the system because it forced those future students to use specific web browsers that supported the Unity game engine plugin. Therefore, we decided that it would be best to abstract the game away from the Website Client and make it its own executable that students can download.

Also, since we abstracted the game so that it is now a standalone client, we also needed to determine the mechanism for communication between the Game Client and the Server in order to have a working Stats API. We decided to use standard HTTP for the communication between the Game Client and the Server because it enabled us to use the same Node.js process and corresponding configuration on the server to handle requests from the Game Client. Unfortunately however, we could not think of an appropriate alternative mechanism of communication over the internet. One of the main downsides to this however is that there is no encryption in the communication that is sent from the game to the server and vice-versa. This issue will need to be addressed by future teams going forward with the project.

Another high-level decision that was considered was to change the Server’s database from MongoDB to a SQL database. The main reasoning we had behind this switch was that no one on the team had any experience working with a SQL-less database such as MongoDB. In the end though, we decided to stick with MongoDB since the schema-less structure of MongoDB went perfectly with the Stats API since game stats are largely undecided at this point in the project. Therefore, by sticking with MongoDB we allow ourselves and future teams to be more flexible with regards to adding data to the database, which helps future work with the Stats API.

### Game Design

The architecture and design of the game was somewhat determined by Unity. Things like file structure and where exactly assets were located were left up to us but more technical decisions were enforced by Unity. It does not run on a traditional object-oriented design, it is more of a Game Object oriented design. Unity Game Objects are created then assigned scripts to control their behavior. We have a single file Reference.cs which contains static methods for retrieving data from and writing data to the database. We have a separate class called ConnectionUtilty.cs that handles all of the interactions with the server. We decided to add a SQLite database for containing information about molecules and for tracking the player’s progress on tasks. At the moment this does not sync with the server but the tasks at least would sync with the players online profile. We decided to use a local database mostly so we could enable an offline mode. This way students are not required to have an internet connection to play the game. It was also a better way to store information about the molecules than in static data files.

### Server Design

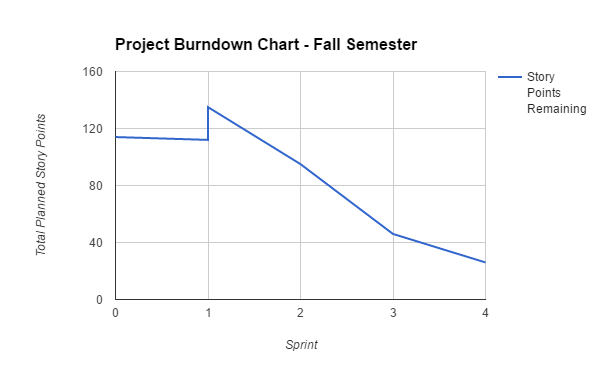
The architecture of the server side of the project largely remained the same this semester. As shown on the architectural diagram, the once a web request is received by the server the Router will send the request to the appropriate Controller, which will then interact with the corresponding Models from the database as necessary. However, at the implementation level this flow works slightly differently for the Stats API, which handles the game’s HTTP requests to store stats in the Server’s database. This difference is that the Stats API has its own Router and Controllers, and this decision was made so that there is a clear distinction between the Stats API and the rest of the Server that handles the Website Client.

## Process and Product Metrics

For our process and product metrics we decided to track our project velocity and defect density based on feature. Since our chosen process methodology was Scrum, tracking our velocity through the use of burndown charts lent itself very well to tracking our progress throughout the project. Also, we chose to track our defect density by feature so that we could determine what features are prone to issues during development. Therefore, with this information we strived to ensure that those issue-prone issues were implemented correctly so that no further issues arise during future maintenance.

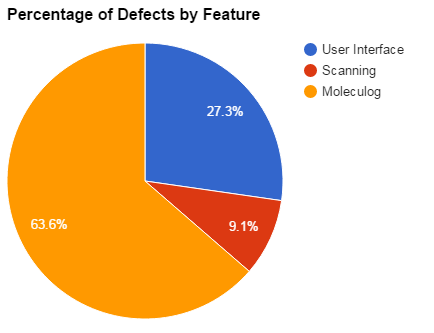
### Fall Semester Metrics

#### Burndown Chart



As shown by our fall semester burndown chart, we generally made good progress with our planned user stories for the fall. However, during the first sprint we misjudged the effort some of our user stories really required in order to implement them, so we increased the number of story points those user stories were worth. This miscalculation is mainly attributed to the fact that most of the team was new to game development, and that we specifically focused on the game during the fall semester so that we all could learn more about game development together. Also, another important note to point out is that we weren’t able to complete some of our user stories that were planned for sprint 4 since we the team generally had other priorities during that time. Therefore, to account for this we tackled those user stories in the spring.

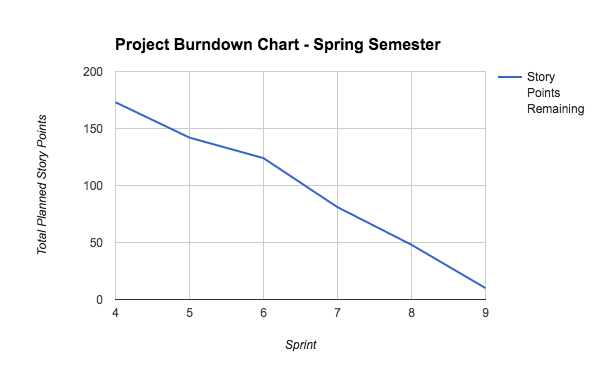
#### Pie Chart showing Defect Density



During the fall semester we had three main features that we were working on within the game. These features were the game’s user interface, the ability to scan a molecule, and the ability to view information about a molecule in the moleculog. As our chart shows, the moleculog feature was the most prone to issues, and this was mainly because of how coupled the moleculog was to the general user interface, the internal game database, and the scanning feature. We used this information during the spring semester to help guide the development of the logbook, which ties the moleculog feature and the task feature together.

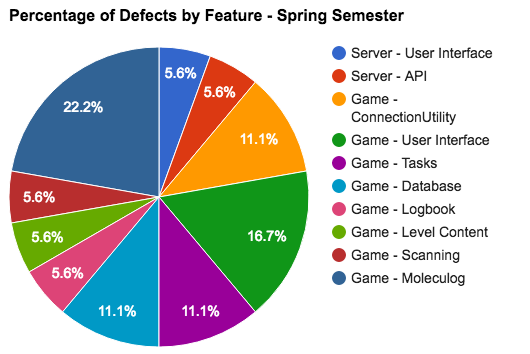
### Spring Semester Metrics

#### Burndown Chart



For our spring semester we decided to shift our focus away from the game and more on the server side of the project. This also included focusing more on the communication between the game and the server. Therefore, since we focused on new aspects of the project we ended up with about 170 story points planned out at the beginning of the spring semester, as opposed to the 115 that we started the fall semester off with. Despite tackling new user stories that required more effort, we made very good progress during the spring semester. This is partly accounted for by the team’s general familiarity of working with the server side rather than learning how to work on the game. In addition, we also had detailed discussions with our sponsor after every end of sprint meeting about what we could possibly accomplish during the next sprint. For instance, after sprint 7 we discussed if the team could add a new level to the game, and we determined that it would be very difficult to do so since we lacked an artist to work on environment assets for a new level. Then as a result, we did not add a user story to account for a new level into our sprint backlogs. Therefore, because we had discussions such as this one we were able to manage the scope for our project and make progress based on our priority of user stories. Also, it is important to note that we were not able to complete 10 story points worth of user stories since these stories focused on some server database rework to account for letting instructors view game stats for their own students. This slippage was simply due to the fact that we didn’t have enough time to implement this feature in sprint 9 since we also needed to work on the project poster during that time.

#### Pie Chart showing Defect Density

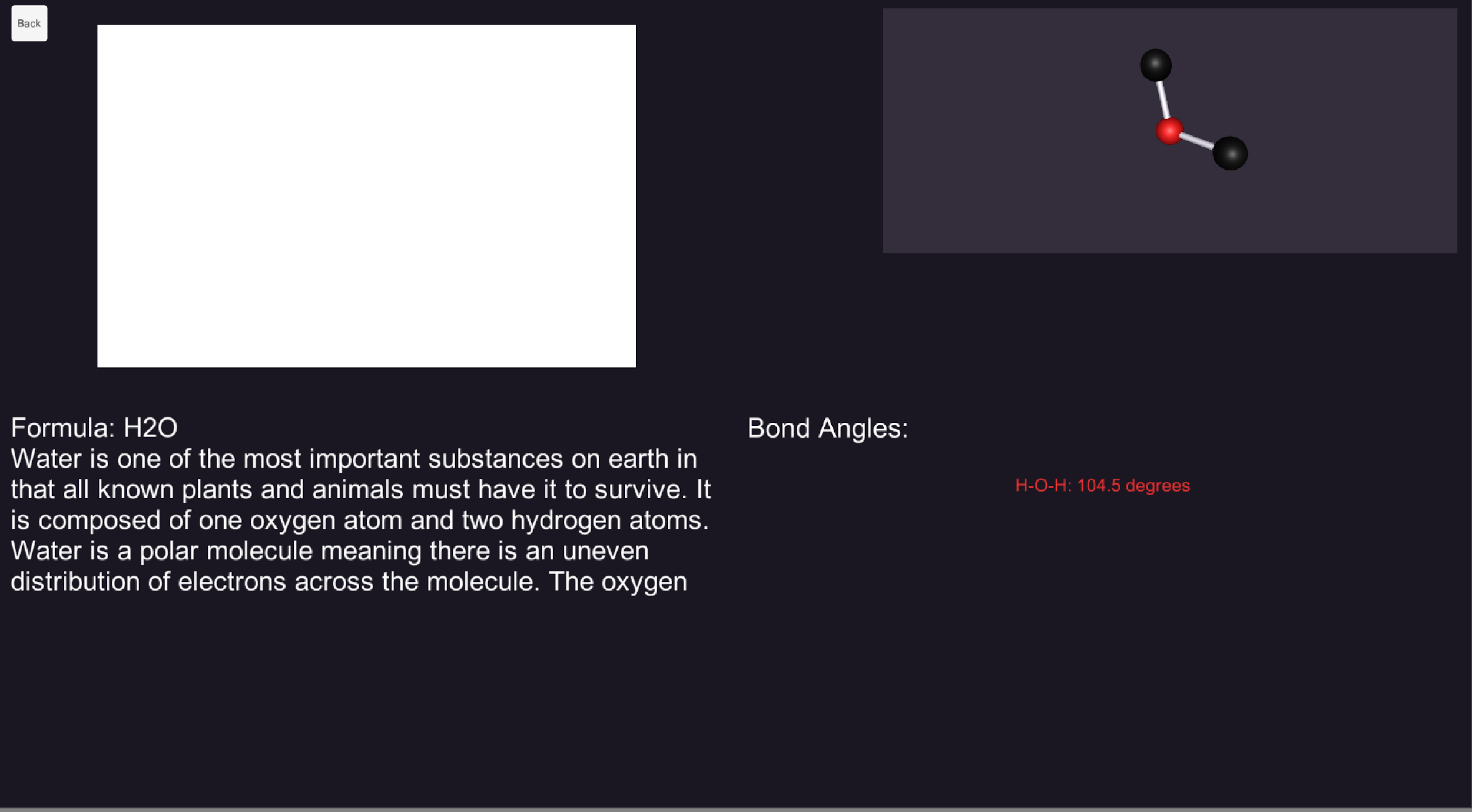


As shown by the defect density chart above, we worked on quite a few additional features as compared to the fall semester. In terms of issues however, the game side of the project still face the majority of issues since 89% of all defects found were associated with the game. Therefore, this shows that we still had more to learn about game development by the end of our final semester. Also, it is important to note that the feature that had the majority of the issues was still the moleculog. This is due to the fact that we integrated the 3D molecule view to the moleculog this semester, and that there were some issues with that integration. In addition, there were also issues with the general user interface while playing the game since we encountered new issues related to not scaling the user interfaces based on the resolution that the game was being run at. Therefore, this information should be used by future teams in the sense that the game requires more quality assurance activities to be performed with it as compared to the server since there are other general game development factors to consider that may be overlooked by software engineers that don’t have game development experience.

## Product State at Time of Delivery

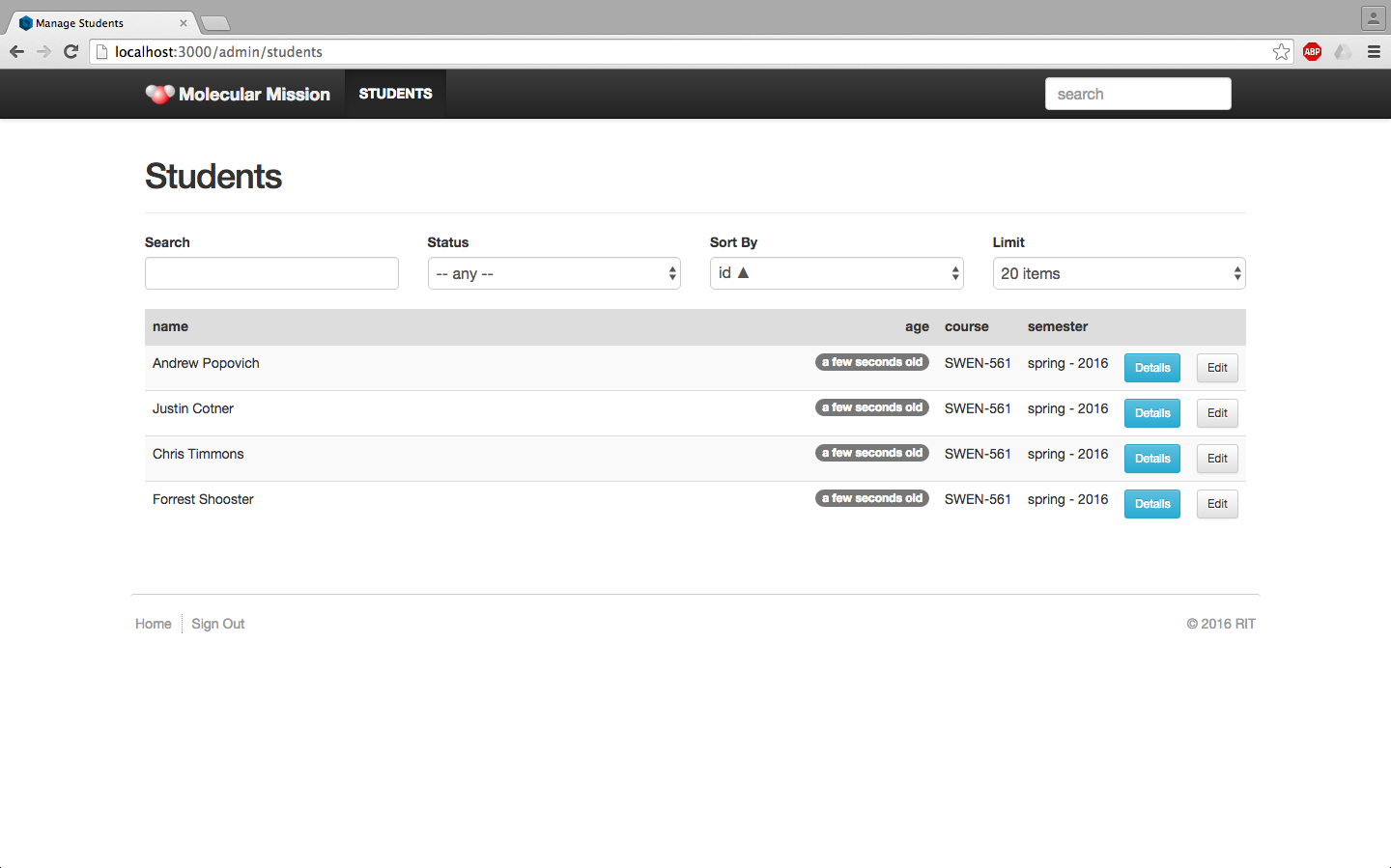
### Game Content

This year we focused on building the game systems and mechanics required before any additional level work could be done. We decided not to add more levels since it would be very difficult to go back and change them once game systems and mechanics were decided on. We decided on five things to focus on as deliverables. These were the Moleculog, Task System, dynamic (programmatic) molecule generation, server connectivity, and an additional level. Of these only the additional level was not delivered. The other four tasks have basic implementations completed. These systems will need to be expanded upon by future teams. The Moleculog currently shows all molecules in the database instead of only molecules the student has scanned. The details screen is also missing the 2D representation of the molecule. The task system only contains basic collection based tasks (collect X molecules). This will need expanded upon. The logbook through which the tasks manager and Moleculog are accessed needs to be worked on visually. Currently the screens are just solid color backgrounds. The dynamic molecule generator is complete. All that needs to be done is add the molecule information to the database and the game will generate the model. Adding molecules to the database is a complicated process so future teams might want to consider creating a tool to help with that.



### Server Content

In terms of the server, most of the implementation this year was focused on establishing the communication mechanism between the game and the server side through the Stats API. Specifically, this API is designed so that it can authenticate a student with their username and password as well as store any particular type of stat to the server’s database. However, this API is a little too simplistic and may need to be expanded upon in order to keep track of more complicated stats, such as the amount of time that the student has played the game. The other features on the server side that we added was the ability for instructors to sign up using a self-sign up service and a UI redesign on the instructor’s main page to account for the variable number of statistics that a student may have. This redesign is primarily in the form of a new ‘Details’ button in which an instructor can press the button for any particular student to view the that student’s particular stats. Also, for the server side all of the promised features were implemented, but can and should be fleshed out a little bit more by future teams.



## Project Reflection

As with any project this one had its ups and downs. The creative freedom that the sponsor gave us was good and bad. It was good since we basically got to define our own requirements. We still validated these with the sponsor to make sure we were on the right track. This did create some problems at the beginning though when we were trying to elicit requirements. We tried a couple times to get a list of requirements from the sponsor and it did not go anywhere. Once we started making our own requirements and bringing them to him it went much smoother. We also felt that we accomplished enough this year to give future teams a good starting point. Some features need expanding upon but they will be in a much better position than if we had simply just added more levels. While we did meet most of our milestones development could have gone better second semester. The team did not have as much time to dedicate to senior project so development slowed quite a bit. This led to another thing that did not go so well, adding another level. This was one of our original project milestones to complete but we did not get to it. Even if we had though it probably would have been a grey box level since there were no designers on the team.

There were a couple things that we would have done differently. One was choosing a different process. Scrum worked well enough but might not be quite right for developing a game. At the very least getting down most of your requirements before starting development would be a good idea. We also could have created a more in depth game design document or storyboard to build off of. Another thing we would have done differently was researching other game engines. We decided to stick with Unity so we could jump right into development but there might have been a better choice.

### References

1. Trello Scrum Board - <https://trello.com/b/uu054LoX/molecular-mission>
2. Project Plan Document - <https://docs.google.com/document/d/1Ws9SV0kxryjIu6QsvqkVjUKAoxxqIfYHDFcfqBLVXMc/edit?usp=sharing>
3. Projecy Metrics Document - <https://docs.google.com/spreadsheets/d/1BX9RL7r6OKvBeEGmFTgoAZ9EYv4KFJWjJXkTJTc4GOI/edit>