

Sprint 2 Retrospective

Team 18 : Brendan Raftery, Kate Lorenzen, Vinson Luo, Jonathan Grider

# What went well?

### **General**

* More frequent meetings
  + The team came together to work on the project more frequently, which boosted productivity and coordination, and helped us fulfill our user stories
  + Programming-focused meetings greatly improved our progression and helped us gauge how much extra time we needed to allot to our user stories.
* All of the user stories were implemented during this sprint
  + During this sprint, much of the algorithm for performing user stories were completed and fully integrated into the user interface of the application to create a functional build. Though we were not able to demonstrate completion of all the user stories in the sprint review, we completed all the listed user stories before the sprint ended
* Improved communication
  + All of our communication channels were used more effectively, which allowed for better meeting coordination and a unified understanding of our progress.
  + Comments and concerns were openly shared and discussed, allowing for a more open and constructive environment.

# 

# 

# Tasks

The following are User Stories that we have successfully implemented in Sprint 2:

* User Story # 1: As a user, I would like to be able to refer to a journal for facts and trivia on an atom of interest.
  + Task 1: Create UI panel to display atom information
    - Visual sprite was created on a separate illustrator program and imported, with text objects awaiting scripted input read from the database.
  + Task 2: Create mock program to show how to interact with the database
    - A C# script was written with an example implementation of a database call, including query-specific code and in-depth comment explanations.
  + Task 3: Create an algorithm to obtain information about the atom from the database
    - A C# script was written to connect to the SQLite database and to find all of the atoms. The records are filtered by what they have unlocked then shows the respective atoms. Their individual popups are given extra information from the database for each respective element.
  + Task 4: Debug and test algorithm using unit tests
    - A C# script was written that contains a series of debug logs from Unity’s coroutines executing to compare the expected output of the Glossary’s atom tab to its actual output.
  + Task 5: Connect algorithm to UI that displays information about the atom
    - A C# script was written to access text objects from the Trium information display panel, and edit it to display correct information of the Trium in runtime.
* User Story # 3: As a user, I would like to be able to refer to a journal for facts and trivia on a compound of interest.
  + Task 1: Create UI panel to display compound information
    - Visual sprite was created on a separate illustrator program and imported, with text objects awaiting scripted input read from the database.
  + Task 2: Create an algorithm to obtain information about the compound from the database
    - A C# script was written to connect to the SQLite database and to find all of the molecules. The records are filtered by what they have unlocked then shows the respective molecules. Their individual popups are given extra information from the database for each respective compound.
  + Task 3: Debug and test algorithm using unit tests
    - A C# script was written that contains a series of debug logs from Unity’s coroutines executing to compare the expected output of the Glossary’s molecule tab to its actual output.
  + Task 4: Connect algorithm to UI that displays information about the molecule
    - A C# script was successfully written to access text objects from the Trium information display panel, and edit it to display correct information of the Trium in runtime.
* User Story # 4: As a user, I would like to be able to have different background color options available for me to chose from in the settings menu.
  + Task 1: Find best colors for background
    - The colors chosen all were somewhat of a neon color. Colors took some time to find because the objects in the scene that needed to change color were a little difficult to figure out how the colors added together.
  + Task 2: Implement buttons and color changing of background
    - An existing implementation was revised to space out buttons more so the user wouldn’t accidently press the wrong button.
  + Task 3: Implement new background
    - Most of background was added in the Cosmic Ranch class. It should be relatively easy to serialize these settings for later loading.
* User Story # 5: As a user, I would like to be able to turn any music that plays on and off in the settings menu.
  + Task 1: Develop ambient soundtrack
    - Ambient soundtrack was developed on a third party application.
  + Task 2: Build UI switch to toggle music
    - Visual sprites were created on a separate illustrator program and imported, which were applied to a UI button in Unity Game Engine to simulate toggle switches, with attached script to toggle music on and off.
  + Task 3: Create unit tests to assess functionality of the toggle switch on the background ambient soundtrack
    - A C# script was written that contains a series of debug logs from Unity’s coroutines executing in sequence to compare the expected output of the music control switch to its actual output.
* User Story # 6: As a user, I would like to be able to turn sound effects on and off in the settings menu.
  + Task 1: Develop sounds
    - Sound effects were gathered from open sources and attached to the application’s user interface OnClick functions.
  + Task 2: Build UI switch to toggle sounds
    - Visual sprites were created on a separate illustrator program and imported, which were applied to a UI button in Unity Game Engine to simulate toggle switches, with attached script to toggle sound effects on and off.
  + Task 3: Create unit tests to assess functionality of the toggle switch on the sound effects
    - A C# script was written that contains a series of debug logs from Unity’s coroutines executing in sequence to compare the expected output of the sound effects control switch to its actual output.
* User Story # 7: As a user, I would like to be able to level up to unlock new atoms and compounds to create.
  + Task 1: Set up experience and player system
    - Most of experience system handled within backpack class since it is a universal object.
  + Task 2: Set up player progression path for Sprint 2 demo
    - The demo path varied a little bit from the actual path planned for the final release of the game, but it did a good job of demonstrating the functionalities that we did implement in this sprint.
  + Task 3: Program in experience on initial task completion vs. repeat task completion
    - This is handled within the backpack, when assigning experience, if the element/tirum being created does not exist yet in the backpack’s data then assign 10x exp points to the user.
  + Task 4: Program experience bar and Stats Bar to show correct experience and level
    - Stats bar works well with the level / left menu circles. It was set up so that the bar would fill itself up over time rather than just complete, it would also finish the rotation when advancing to another level. We might add some communication between the exp bar and the level text so that the level text changes once the bar actually circles around, rather than right away.
  + Task 5: Create unit tests to ensure proper functionality of the UI and ensure proper leveling
    - All tests use the addExp() function within the backpack and rely on us seeing if adding exp works properly. Issues do occur if we go over the programmed level (level 3), but this issue was easy to fix and will just need to be capped for builds made for users.
  + Task 6: Make it so that when the user levels up, there will be a notification of the new atoms that the user can fuse into
    - This text will be displayed below the “Fuse 2 Elements Together” text at the top of the screen, or below the Fusion bar tab. This will be easy to implement and we will decide together where it should be displayed
* User Story # 8: As a user, I would like for there to be a chemical formula listed at the top of the user interface that will help guide me in selecting the chemical structures needed for the reaction or bonding I am in the process of completing.
  + Task 1: Create UI to display a chemical formula of choice and buttons to confirm or cancel an action
    - Visual sprite was created on a separate illustrator program and imported, with text objects awaiting scripted input read from the database.
  + Task 2: Connect parsing algorithm with UI algorithm
    - The use of rich text in unity made this task easier than it could have been if we would have needed to add in text boxes with font colors using a script. The only time consuming part was coming up with a proper algorithm that would not be time intensive.
* User Story # 9: As a user, I would like to be able to perform Fusion on two atoms to create a new element.
  + Task 1: Create Algorithm to perform Fusion
    - A C# script was written to perform “fusion” of two selected atoms within the workspace of the application. The selected atoms would be used to determine the next appropriate fusion product, obtain its information via SQLite query, and store the newly acquired information in a new Buddy object as well as in our dynamic data structure.
  + Task 2: Create unit tests to ensure proper functionality of algorithm
    - A C# script was written that contains a series of debug logs from Unity’s coroutines executing in sequence to compare the expected output of the action Fusion to its actual output.
* User Story # 10: As a user, I would like to be able to perform Group on two or more atoms to create a single-element molecule.
  + Task 1: Create Algorithm to perform Group
    - An existing C# script was enhanced to support both Grouping and Reactions, as these two processes follow the same logic. This allowed our process to be more robust and enabled the reuse of existing code functionality.
  + Task 2: Create unit tests to ensure proper functionality of algorithm
    - A C# script was written that contains a series of debug logs from Unity’s coroutines executing in sequence to compare the expected output of the action Group to its actual output.
* User Story # 11: As a user, I would like to be able to perform a Reaction on two or more Triums to create a compound.
  + Task 1: Create Algorithm to perform Reaction
    - Reaction to perform algorithm was created using 4 lists of information along with the ‘selected’ list which consists of all triums that the user has selected. This is the function that is also now used with grouping as grouping requires the same input information and the code can be reused.
  + Task 2: Create unit tests to ensure proper functionality of algorithm
    - A C# script was written that contains a series of debug logs from Unity’s coroutines executing in sequence to compare the expected output of the action Reaction to its actual output.

The following are tasks that we have successfully implemented in Sprint 2 whose encompassing User Story was not fully completed:

* User Story # 2: As a user, I would like to be able to open a list that shows all of the possible compounds that are made up of at least one of the structures that is within my workspace.
  + Task 1: Create a scrolling list that will auto generate its contents at runtime
    - Visual sprites were created on a separate illustrator program and imported, and the scrolling list user interface was implemented in the Unity Game Engine, with buttons populating the list and triggering action on click.
  + Task 2: Create an algorithm to read compound data from the database
    - A C# script was written to obtain information about what the user has unlocked and use this to gather possible reactants and product from the database.
    - The database was enhanced to allow for an easier yet structured parsing of the relevant reactants and products a Trium was a part of.
  + Task 3: Create an algorithm to parse compound product and reactant data
    - The C# script created in Task 2 was enhanced to parse the obtained data through the use of multiple string delimiters, and present the it in a way that outlined the Triums a user was required to select and use.
    - The database was populated with reactant and product data within existing and newly created tables.
  + Task 5: Connect parsing algorithm to scrolling list
    - The algorithm was attached to the scroll list script, which will parse the information read from the database and successfully display the parsed text on the scroll list buttons

# What did not go well?

### **General**

* Outside activities prevented progress in some cases
  + Commitments to outside activities and courses (not limited to other projects, meetings and exams) inhibited our abilities to meet as a group or stalled progress.
* Testing and debugging occurred towards the end of the sprint
  + Our efforts to use unit tests and other forms of testing occurred later in the sprint, creating some stress and required last minute fixes and/or changes to our implementations.
* Some of the previous setup that we organized has shown to be hindering progress
  + The architecture of our project in Unity is causing some added work that likely could have been prevented if we had previous knowledge of the software. While these functionalities weren’t implemented within this sprint, we did see the complications arise from those decisions. It is important that we research all possible upcoming design decisions so we can avoid obstacles down the road.

### **Tasks**

The following are User Stories that were neither successful nor fully implemented, as well as the tasks that were not completed for the aforementioned story. *Tasks successful to each user story are listed in the above section titled “What went well?”*

* User Story # 2: As a user, I would like to be able to open a list that shows all of the possible compounds that are made up of at least one of the structures that is within my workspace.
  + Task 4: Debug and test the read & parsing algorithms with unit tests
    - Although the user story was completed and implemented before the end of the sprint, it did not make it into the build we used for demonstration in the sprint review. A unit test was not written to assess its functionality.

# How should we improve for Sprint 3?

* Continue to have more meetings, especially those involving group programming
  + While more meetings greatly improved our performance over sprint two, additional programming sessions would help ensure that all of our user stories are completed on time.
* Ensure full completion of User Stories
  + Although many of the user stories were successfully completed, we ran out of time to include the last working implementation of a user story in the final build for the sprint review. We have made great progress in this sprint compared to the last one. Therefore, we should continue on our current path, and further improve on planning accordingly and keeping track of everyone’s progress to ensure that tasks are completed. We will work on leaving the required time to gather and collaborate on merging each member’s parts to create the final product.
* Work on spreading out progression throughout the sprint
  + Although we have improved during this sprint, our progress as a whole still happened towards the end of the sprint, like the previous. We can try to do a better job of either spreading our workload throughout the sprint or by trying to get more done towards the beginning of the sprint. This will not only help us complete our tasks earlier on, but will allow for more debugging and give us the time to make our project more robust.
  + With additional time, we may also be able to work on completing a higher number of user stories, giving users more functionality, and a more complete experience.