# 

# TextLogo.png

# 

# 

# Team 18 - Design Document

Brendan Raftery, Kate Lorenzen, Vinson Luo, Jonathan Grider

# 

# 

# 

# Table of Contents

1. **Purpose**
   1. Functional Requirements 2
   2. Non-functional Requirements 4
2. **Design Outline**
   1. High-Level Structure Overview 5
3. **Design Issues**
   1. Functional Issues 7
   2. Non-Functional Issues 9
4. **Design Details**
   1. Class Level Design 10
   2. Database Schema Mockup 11
   3. Sequence Diagram 13
   4. Action Bar Detailed Sequence Diagram 14
   5. UI Mockup 15

# 

# 

# 

# 

# Purpose

Science is a crucial subject that exists in every aspect of our lives, but it is difficult to get that across to children. Therefore, having the appropriate teaching method for a young audience can help maintain engagement and boost learning from a fun, positive experience. Our project as a solution is an educational game app that is designed to teach users basics about chemistry and matter through a fun and interactive gameplay experience. Players will be able to collect atoms and create new compounds while learning trivia and factoid about all of the different elements on the periodic table and the chemical compounds they can synthesize.

# Functional Requirements

1. **User**
   1. Introduction
      1. As a user, I would like a tutorial for most features of the game
      2. As a user, I would like to be able to zoom in and out of my workspace to see larger and smaller areas
   2. Individual Atoms
      1. As a user, I would like to be able to collect atoms representing the naturally occurring elements
      2. As a user, I would like to have my collection of atoms visualized inside of my workspace
      3. As a user, I would like to be able to select an atom from the workspace to look at its stats and information
      4. As a user, I would like to be able to refer to a journal for facts and trivia on an atom of interest
      5. As a user, I would like to be able to increase my atoms’ level stats
      6. As a user, I would like to be able to refer to a glossary for all of the possible atoms discoverable in the application, and the ones discovered thus far
      7. As a user, I would like to be able to view a visual representation of the elements I have unlocked on the periodic table
      8. As a user, I would like to be able to discard an atom into a wastebin after receiving a notification to make sure I am certain of my decision
   3. Compounds
      1. As a user, I would like to be able to perform chemical reactions on chemical structures within my workspace to formulate new ones
      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
      3. 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
      4. As a user, I would like to have my collection of chemical compounds visualized inside of my workspace
      5. As a user, I would like to be able to select a compound from the workspace to look at its stats and information
      6. As a user, I would like to be able to refer to a journal for facts and trivia on a compound of interest
      7. As a user, I would like to be able to increase my compounds’ level stats
      8. As a user, I would like to be able to refer to a glossary for all of the possible compounds discoverable in the application, and the ones discovered thus far
      9. As a user, I would like to be able to view a visual representation of the compounds I have unlocked
      10. As a user, I would like to be able to discard a compound into a wastebin after receiving a notification to make sure I am certain of my decision
   4. Settings
      1. 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
      2. As a user, I would like to be able to turn sound effects on and off in the settings menu
      3. As a user, I would like to be able to turn any music that plays on and off in the settings menu
      4. As a user, I would like to be able to turn off and on the faces that appear on the chemical structures
   5. Education
      1. As a user, I would like to be quizzed to see if I am retaining the information taught to me on the app
      2. As a user, I would like for facts that I unlock to be narrated and accompanied with a picture if applicable (if time allows)
   6. Social
      1. As a user, I would like to share newly discovered elements, molecules and compounds with others (if time allows)
2. **Developer**
   1. As a developer, I would like to implement smooth graphics that will increase aesthetic appeal to the users
   2. As a developer, I would like to use a database to store factual information about elements, molecules, compounds, and reactions
   3. As a developer, I would like to allow users to sync their progress to a database
   4. As a developer, I would like to restrict the speed at which a user can progress
   5. As a developer, I would like to be able to update the application when more information gets added or bugs get fixed

**Non-Functional**

1. Must be able to play this game on Android products
2. Must be able to play this game on Apple products (if time allows)
3. Must have an interface that is intuitive and easily navigable
4. Must have a database that is easy to maintain and extend for new content
5. Must not be too complicated that detracts from gameplay experience

**Architecture and Performance**

We will be using the Unity Game Development engine, which utilizes the Mono .NET Framework and the programming language C#. We will be storing our data into a custom database that will contain all of the chemical structures, reactions, formulas, facts, and anything else that a structure would need in order to be used fully within the game.

We want to deliver to the user a smooth gameplay experience with a relatively low app file size. For a project with a large number of game elements such as this, we need to minimize the overall file size of the imported visual assets while still maintaining professional quality. Routes to achieve this goal will include using image manipulation code to modify images rather than having them stored as another file.

**Security**

It is critical to secure the user’s progress information in the game. We will store the data locally, which allows easy access and better control over the data. If time permits, we can encrypt the data to secure the information and prevent users from manipulating the data to gain uneven advantage. Additionally if time permits, we may go further to store user data on an online server.

**Usability**

The application will have a main screen that involves an area where all of the elements can be seen moving around. On that screen, there will also be noticeable, clearly labeled buttons to access other functions like fusing elements, viewing trivia, etc. The user interface itself will be intuitive, detailed, and it will perform smoothly to avoid difficulties interacting with elements of the game. Users of all ages will be able to use the app without frustration.

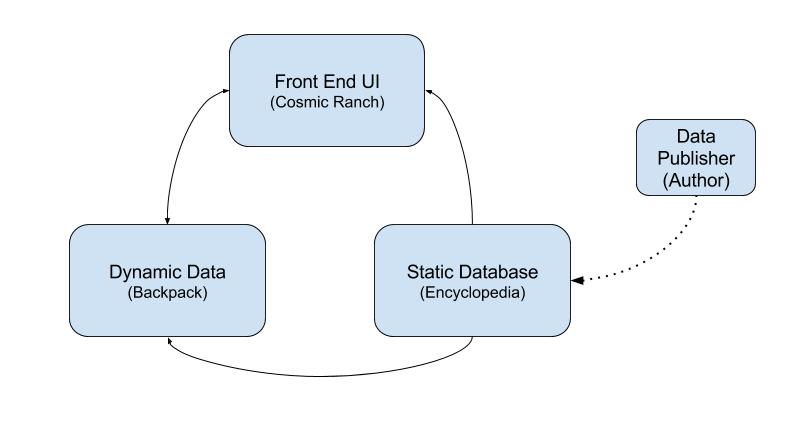
**Deployment**

The application will be deployed onto the iOS App Store and Android Google Play Store. Any updates to our app can be done through those app stores through normal app updates.

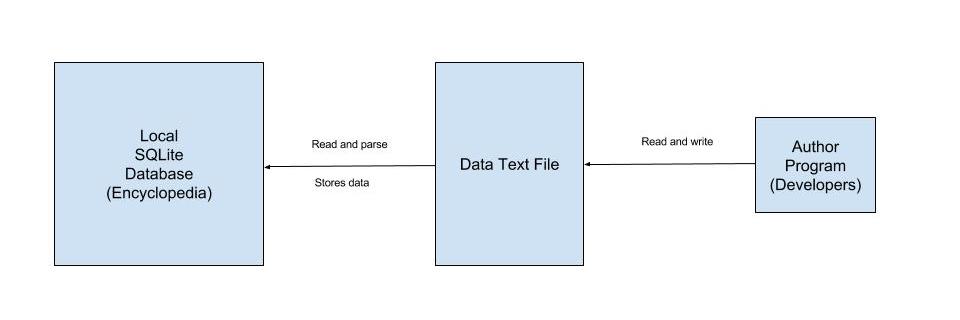
# Design Outline

### **High-Level Structure Overview**

Our project will be an educational game where users will learn introductory chemistry through gameplay. The users will collect elements and formulate molecules and compounds, unlocking information and facts on the respective object. This application will not adopt a Client-Server model, as it will have a fat client. The fatclient will have most of the data stored locally in a dynamic dataset and static database. It will perform most of the actions that a server would. The dynamic dataset stores data on the object instances the user “owns,” for example, the atoms and compounds the user owns in their space. The static database stores data on everything the client and the dynamic dataset will refer to and remain unchanged, such as the information on elements and compounds in the application, as well as the status, locked or unlocked, for each.



1. Client (Front End UI)
   1. Client will have the UI for interaction with our application
   2. Client will request information from the static database and create instances in itself
   3. The information of the instances created in the client will be stored in the dynamic data, and updates on the data will be stored in the dynamic dataset
   4. The client will request information from the dynamic dataset on the instances in the front end UI
2. Dynamic Dataset
   1. Dynamic dataset will refer to static database for default data
   2. Dynamic dataset modifies and stores the dynamic data from the Client front end UI
3. Static Database
   1. Static Database stores default data on all objects in the application
   2. Data mostly consist of text and strings, such as atomic name, symbol, compound formula, etc.
4. Data Publisher



The Author will be a development tool used by our team in order to easily add in and manage new Triums into our database. The developer will run author on his/her computer and will be prompted on what actions they would like to perform. Whether it being adding, removing, or editing an entry. Once the file has the data we want in it then it can be placed within our Unity files and be parsed into the SQLite database where it can then be saved into a compressed game data file. We do not want to include the actual raw text file within our released product since it would make it easier for people to change information of our database if the file were to be parsed into the database after being edited by a user.

# 

# 

# Design Issues

### **Functional Issues**

**Issue:** How should we moderate the user’s progress speed in the game?

**Option 1: Implement a level system**

Option 2: Implement a currency system

We decided to implement a level system. Although a currency system could allow the user to manipulate the portal to get the element they want, the main topic of discussion was ‘how do we stop a player from progressing too quickly?’. With a level system, the player will gain experience every time they create an element, compound, or molecule (aka Trium). Their level will also constrict the highest atomic number of element they can reach at that moment in time. This will encourage the user to explore different Triums using the ones they already have. By only allowing certain elements to be obtained by each level, we are supporting the educational aspect of the game. On the other hand, if we used the currency system that was discussed, they could quickly reach the most complex Triums without having discovered all of the ones in between since they could simply ‘pay’ their way through the periodic table.

**Issue:** How does the user receive new atoms from the Wormhole?

Option 1: Constant supply of atoms

Option 2: Tapping for one randomly generated

**Option 3: Tapping with holding down option to specify which elements would have a higher chance of spawning**

We chose tapping the Wormhole with a holding down option to specify which elements would have a higher chance of spawning. We decided that the constant supply of atoms would overflow the screen and give the player too many atoms to deal with. With option two, the simple tapping option would not benefit the player. For example, if they require an oxygen atom and have the toss up between 8 or so atoms that could be generated, they may get frustrated not being able to get the one they need. From those two discussions, we decided that tapping is the more interactive way of having the user gain atoms. But to solve the issue at hand, we decided that the Wormhole will have a wheel of the elements they have gained access to that would show up on the screen if the Wormhole is held down. They can then scroll through all of the possible generated elements and can choose what four elements will be the only possible elements. Once they tap the Wormhole again, one of those four chosen elements will appear, giving them a greater chance of progressing the way they want to.

**Issue:** How many buttons should there be to allow a reaction?

**Option 1: One big react button that would induce any reaction**

Option 2: Multiple little buttons that would show options of how the user could induce a reaction (add heat, etc.)

Option 3: Both option 1 and option 2 depending on what Trium are a part of the reaction

After contemplating the accuracy of the information in our application, it was decided that having a simple react button would be best. Although reactions often involve energy, it would take us more time than necessary to implement a system that specifies the reaction. Instead, we will have a button that simply says “React.” Once clicked, it will say that they added energy to create the new Trium. This decision could also be seen as more user friendly for the intended audience.

**Issue:** What should the compounds look like if they are unlocked, locked, or can be unlocked?

Option 1: Colored - unlocked, Grey - can be unlocked, Black - locked

**Option 2: Colored - unlocked, Grey - locked or can be unlocked**

With having only two possible colors, we can simply check to see if the compound has all of the reactants or not. If it does, keep it colored, and if it does not, grey it out by turning off color saturation using code. With option one, there would have to be a system constantly parsing through every single reaction and check if any part of it can be used. With only having to check the entire reaction if it is tapped, it will be easier to classify the Triums as unlocked or locked since we would only have to check the selected Trium’s reactants.

**Issue:** What should the background of the ‘Cosmic Ranch’ space look like?

Option 1: An image of the universe/galaxy

**Option 2: A solid color**

Although the theme of our game does involve outer space, solid color is the better option. Since we have a zoom in function when the user taps on a Trium, the image would require a high resolution. The size of the image would force our game file size to be much bigger than we intend, which can impact our performance. Having a solid color would not alter our file size, as it will most likely be code generated. This can ensure that our user interface does not distract from the fun and educational elements of the game.

**Issue:** How do we test the player's knowledge?

Option 1: Give an optional quiz question for the user to take after an amount of time

**Option 2: Quiz them when making a new reaction**

We decided to test the user by giving them a trivia question when they are about to unlock a new compound. This will force the user to test their knowledge and use all of the educational value that our game possesses. The quiz question will be based on an element they already have access to so they have the ability to learn the facts in the glossary and test what they learned. If we had a timed quiz question that popped up in the corner but did not force the user to do it, most users would ignore it and dismiss the reason we are creating this application.

### **Non-Functional Issues**

**Issue:** What framework should we use to implement this application?

**Option 1: Unity**

Option 2: Create our own framework

Our group chose to use the Unity game development engine. Creating our own engine or framework for the game would create an unrealistic amount of programming to be implemented in the given time span. Using the Unity engine will provide us with dozens of powerful libraries and allow us to render our game with relative ease. In addition, members of our group have pre-existing knowledge of Unity and other engines like it, making it easier for us to start using it and saving more development time in the process.

**Issue:** What language do we want to implement the application in?

Option 1: JavaScript

**Option 2: C#**

Unity allows for much of its programming to be written in either JavaScript or C#. We decided that using C# would be the best route since we have members with past-experience using C#. In addition, C# has numerous powerful libraries at our disposal, and its Object-Oriented functionality makes it a powerful tool that can make use of O-O organization and constructs.

**Issue:** What database system is ideal for what we need to store?

**Option 1: Local database**

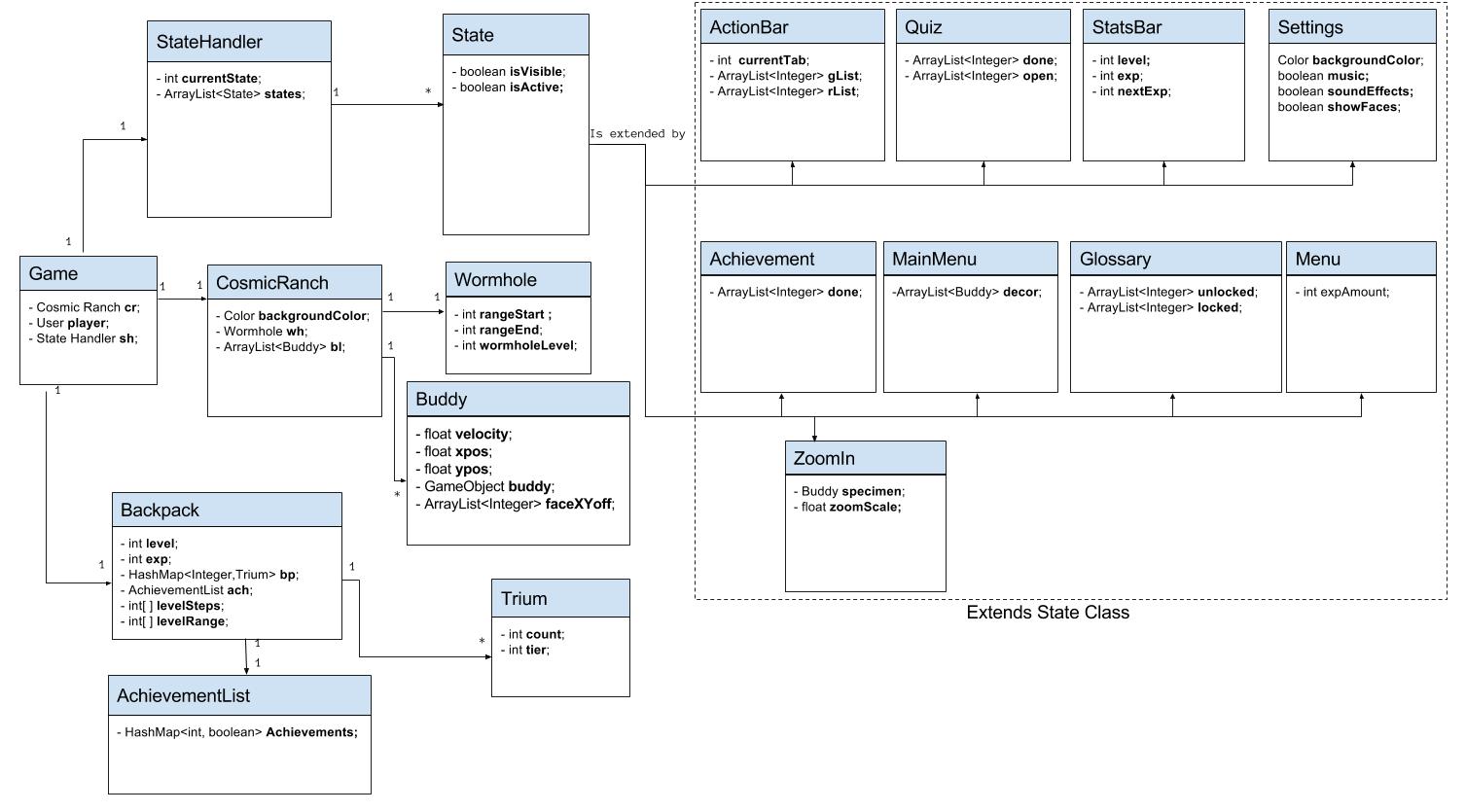
Option 2: Server-hosted database

We opted to use a local database within the application. Using a local database allows us to avoid the cost of hosting a server and the application to operate without requiring a network connection. In addition, because of the way we would like to handle factual information about elements, molecules and compounds, a static local database can more than suffice these needs. It would act simply as a group of static lookup tables from which we can store information about Triums such as their name, mass, chemical formula and facts about them.

# 

# Design Details

### **Class Level Design**

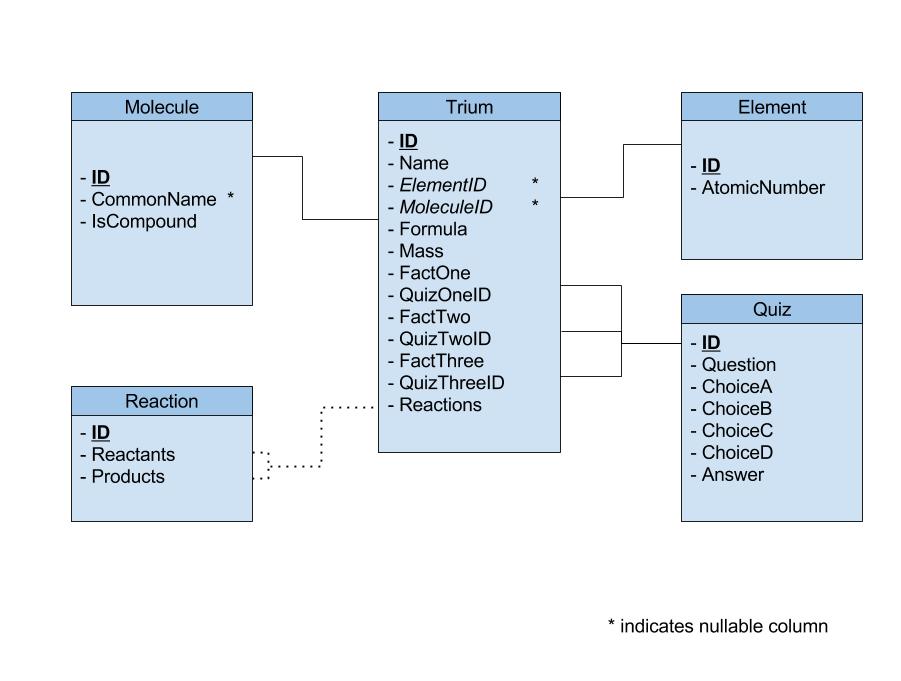


The overall goal of our program structure is to have a relatively light front end that is very efficient in grabbing necessary information from our locally stored database (the “Encyclopedia”). This structure can be divided into 3 major components. The Cosmic Ranch, the Backpack, and the State Handler, all of which are controlled by the overarching Game main class.

The Backpack is the main area for storing dynamic data about the player such as level, experience, achievement information, and statistics about the Triums that a player has. The backpack holds a HashMap of Triums, all of which hold the count and the tier of that trium. These two pieces of information is what we will need in order to control several other components of our game in coordination with the Encyclopedia. We will take this information to not only show the player what they have collected and accomplished but also to determine what the next steps for the player should be if they want to progress efficiently and have the ability to create more new Triums within a shorter period of time.

The Cosmic Ranch is the the workspace area of the game where two structures will reside. The first being the wormhole, which is where new atoms are generated for the player. What atoms the wormhole can generate will depend on the player level, but preferences can be given to the wormhole on what atoms you wish to generate. The other structure on the ranch will be the Triums that you have created or your “Buddies”. Buddies will be programmed to float within your ranch and the player will be able to interact with them by either moving their finger near the buddy, or by selecting the buddy and entering “Zoom Mode”, which shows more information about the buddy and allows for the player to remove it from the area if they so desire.

The last component in this trio is the State Handler. This class will handle all of the UI elements in the game and control how the user can navigate through these different screens and menus. All of the classes that extend the State class will control their individual UI components and also be coordinated to work well together using the StateHandler.

**Database Schema Mockup**

Our project’s database is designed to be static, as it acts for the sole purpose of looking up information. This means that nothing the player does in the game will change the contents of the database. The tables are connected explicitly and implicitly, with some direct links and some that require parsing of information. This will be explained in greater detail below.

1. Trium
   1. Contains the basic information about Elements/Molecules/Compounds such as the Name, Formula (or Element symbol) and Mass of the Trium, among others.
   2. Can be connected to either the Element or Molecule table, but not both.
   3. Is connected to three rows in the Quiz table, corresponding to a quiz about the Trium.
   4. A Reactions column includes a varchar representation of a list of Reaction IDs separated by a delimiter. This is used to parse and determine Reactions related to the given Trium.
2. Element
   1. Contains information specific to an element, such as its Atomic number.
3. Molecule
   1. A distinction of the Trium, also functions as the Compound table.
   2. Contains a “Common Name” column if there exists a common name for the Molecule / Compound (E.g., Water).
4. Reaction
   1. This table represents a general reaction, with two columns to represent the two sides of a chemical reaction equation.
   2. Each entry in the Reactants and Products column is a varchar representation of that side of the equation. The varchar string separates Elements and Molecules with a + or - delimiter so that they can be parsed and sorted later.
5. Quiz
   1. Contains information about a quiz, of which there are three per trium.
   2. Has a question, answer choices and a correct choice column.

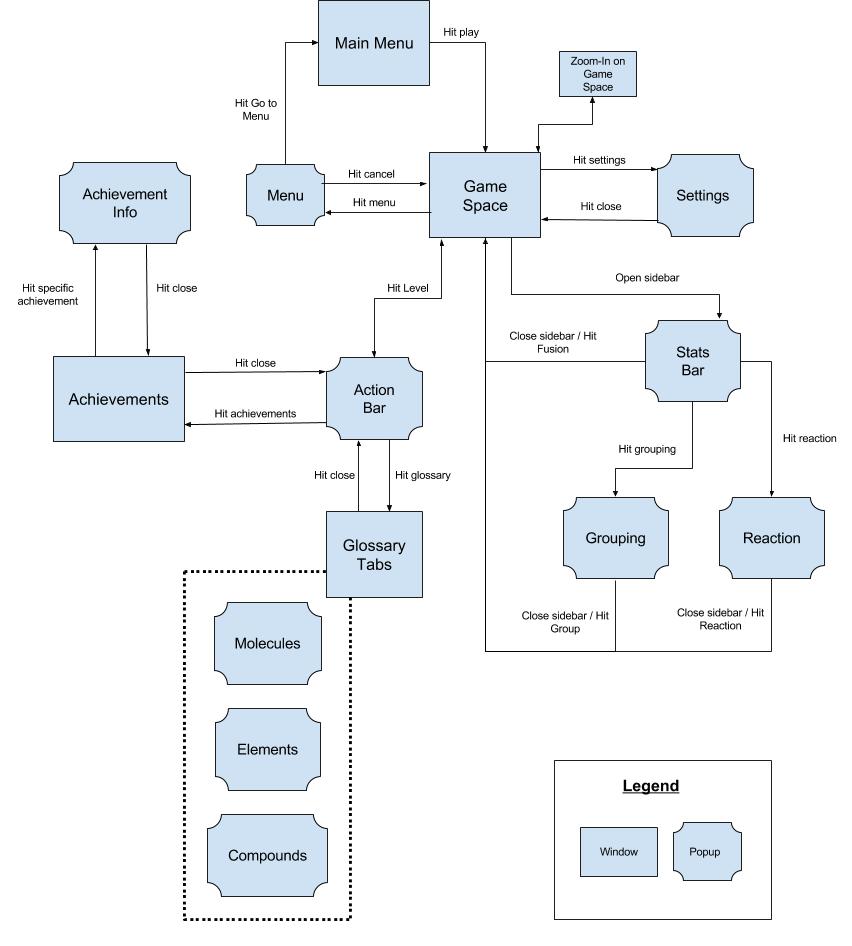
### 

### 

### 

### 

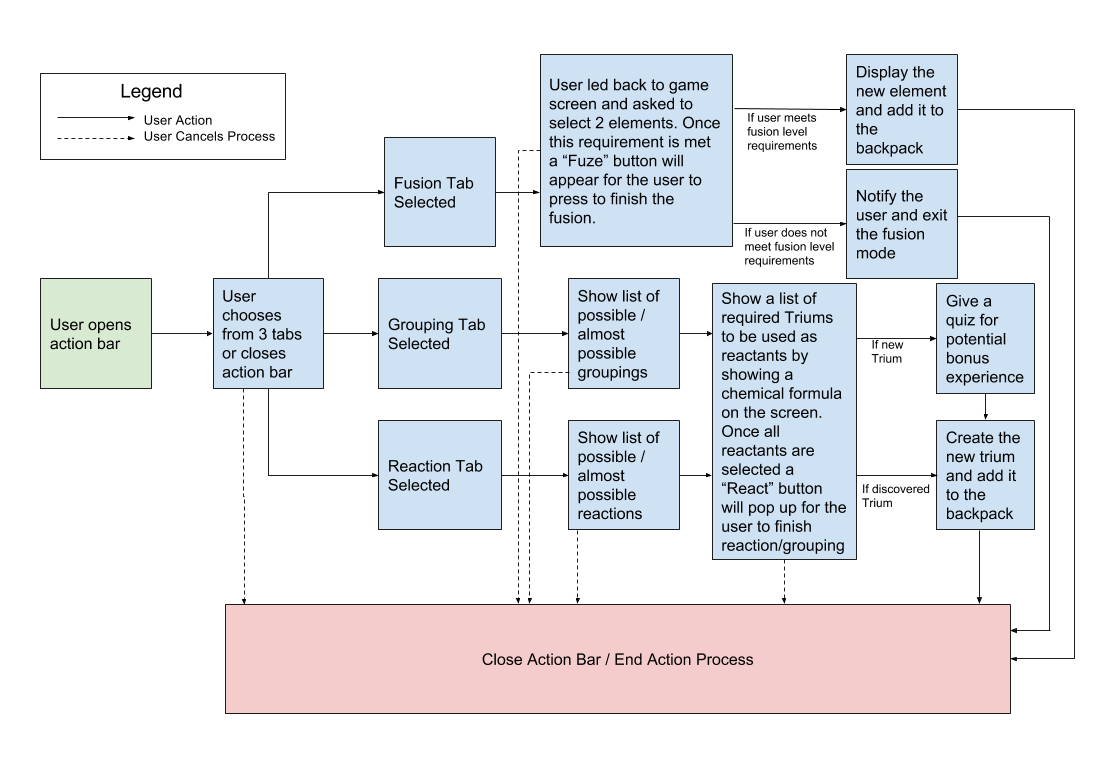
**Sequence Diagram**



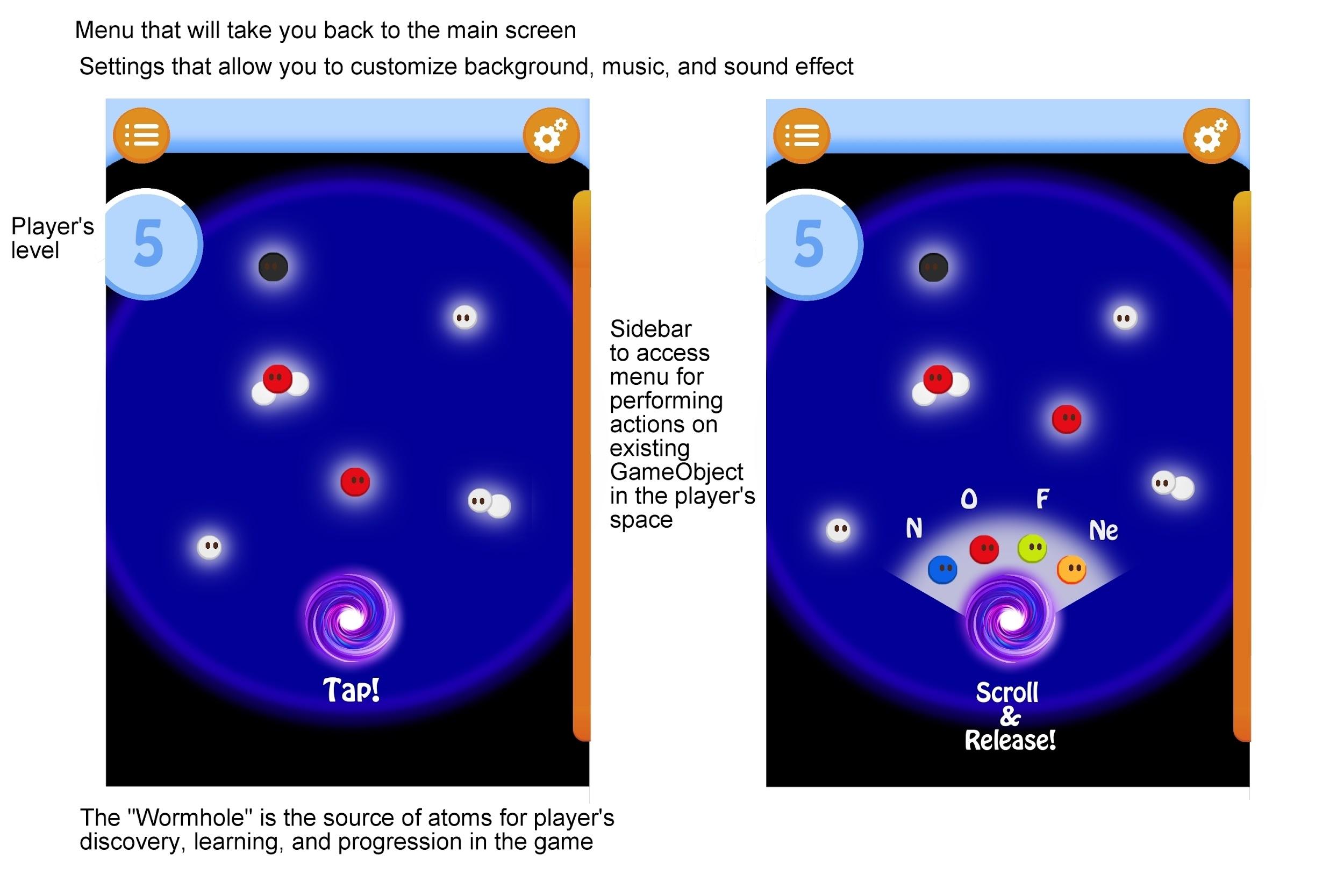
The sequence begins with the user starting in the main menu, where they have the option to quit out of the application or continue into the main game space. The user can do several things in the state, all lead to pop ups above the current state. These states, such as settings, glossary, achievements, actions from the side bar, will refer to the dynamic dataset or static database for data and information.

All of these states and pop up states will be handled by the StateHandler class. All states will have two important components, being visible, and being active. The visibility of a state will be simply whether or not the state is drawn onto the application. When the state is not visible then it will also not be active. Being active means that the user is allowed to interact with the components of that state. We can deactivate states that we want to have appear in the background such as the GameSpace whenever another state should be taking the user’s attention, such as the ActionBar.

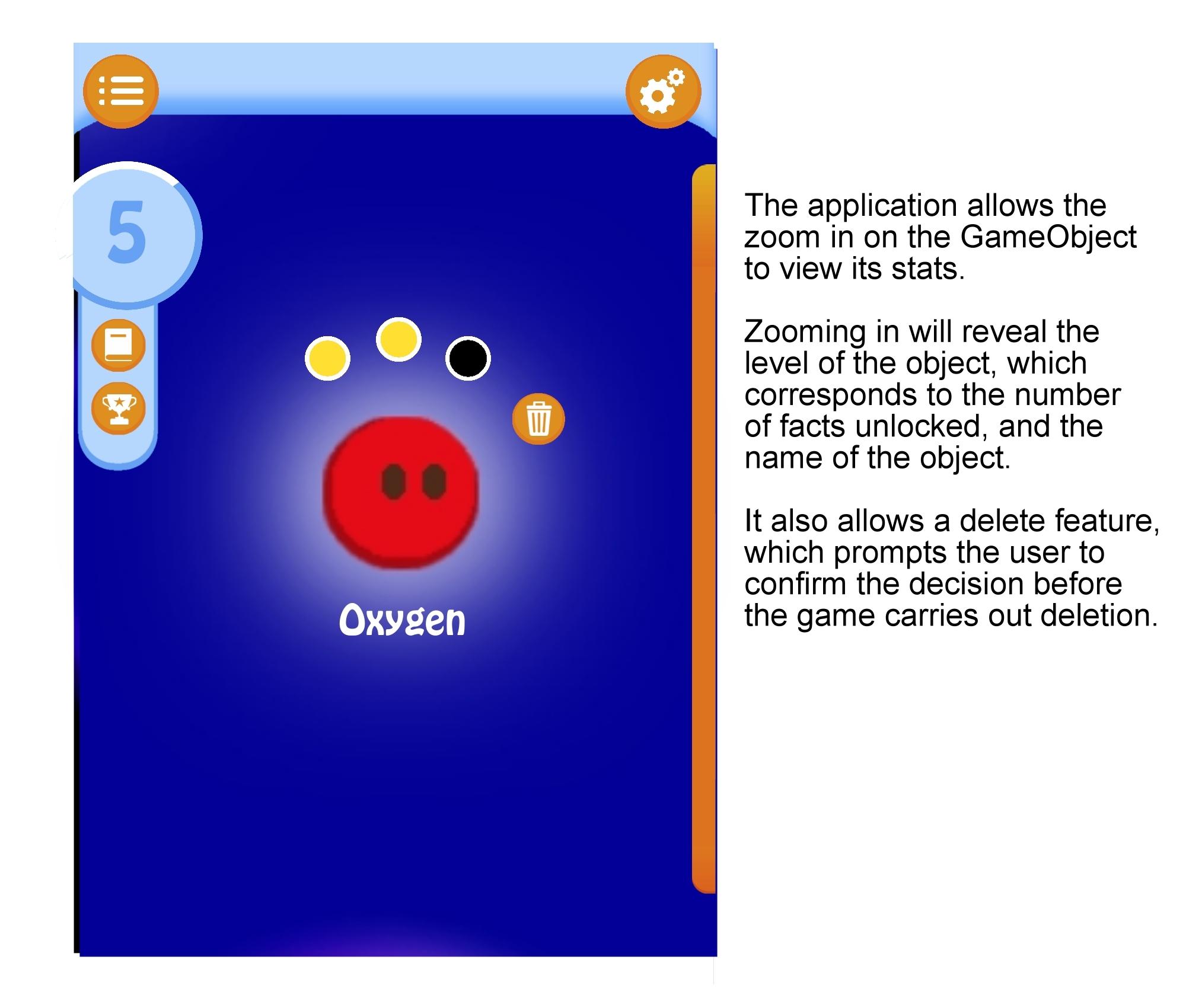
**Action Bar Detailed Sequence Diagram**

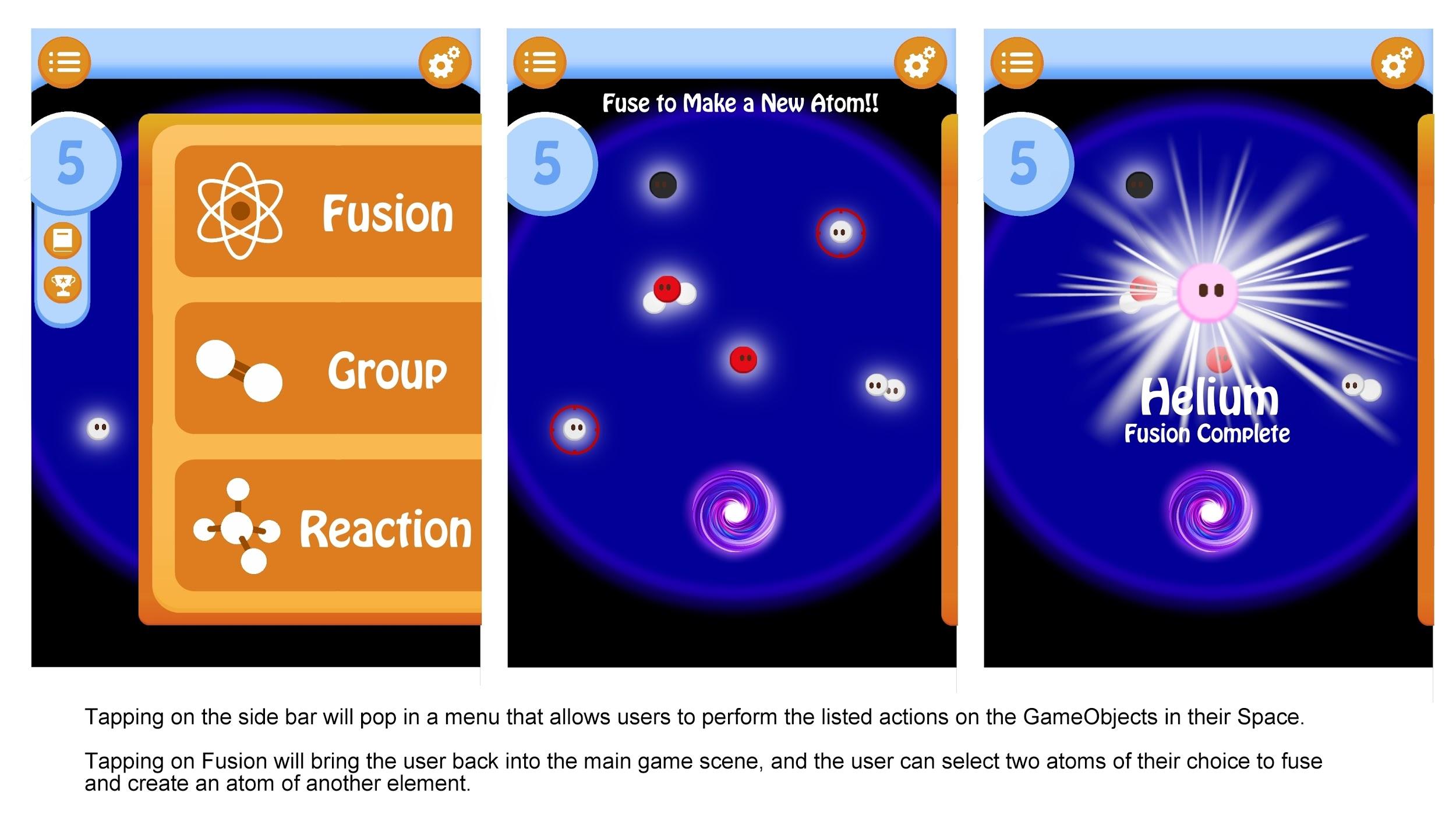


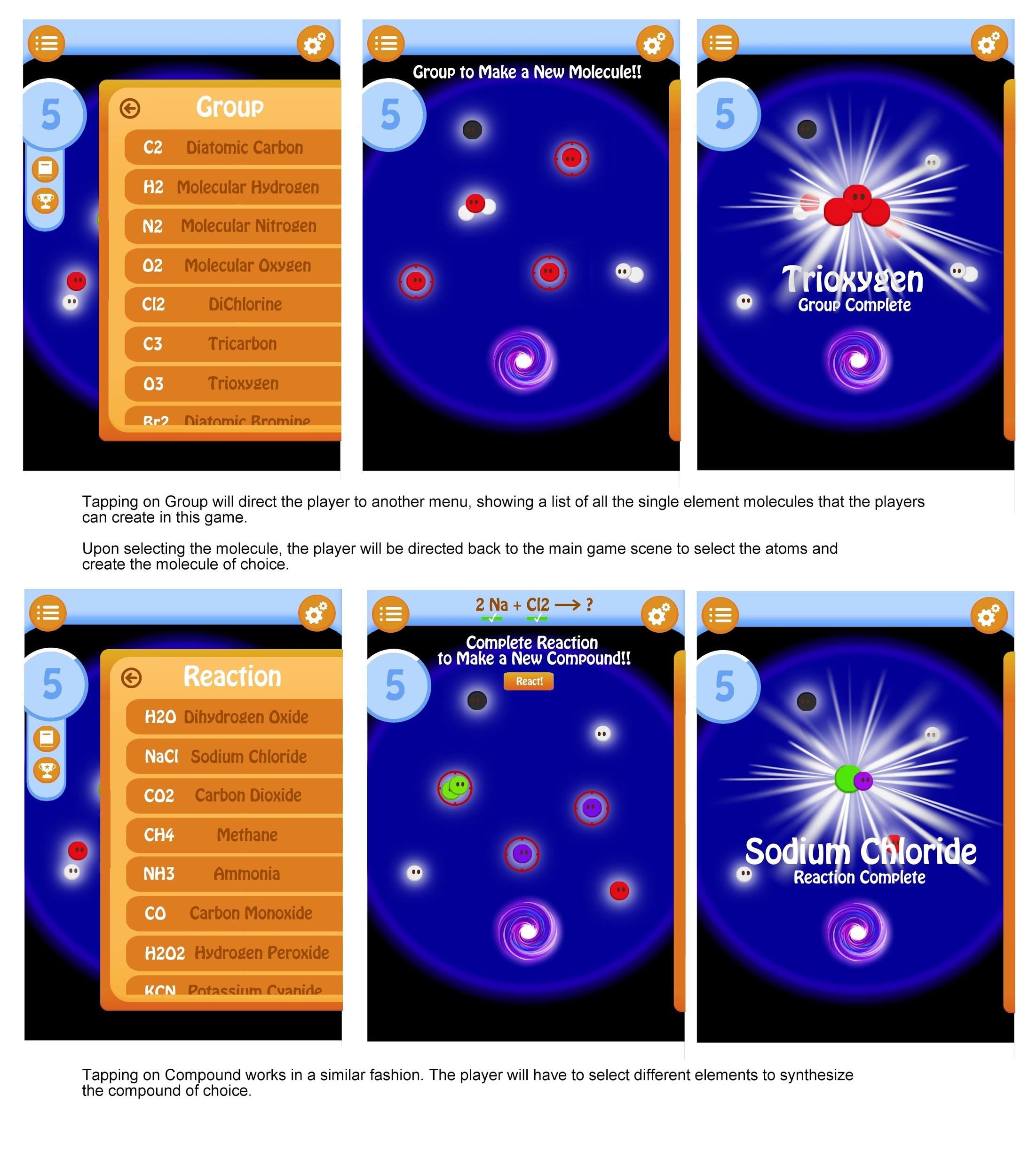
**UI Mockup**



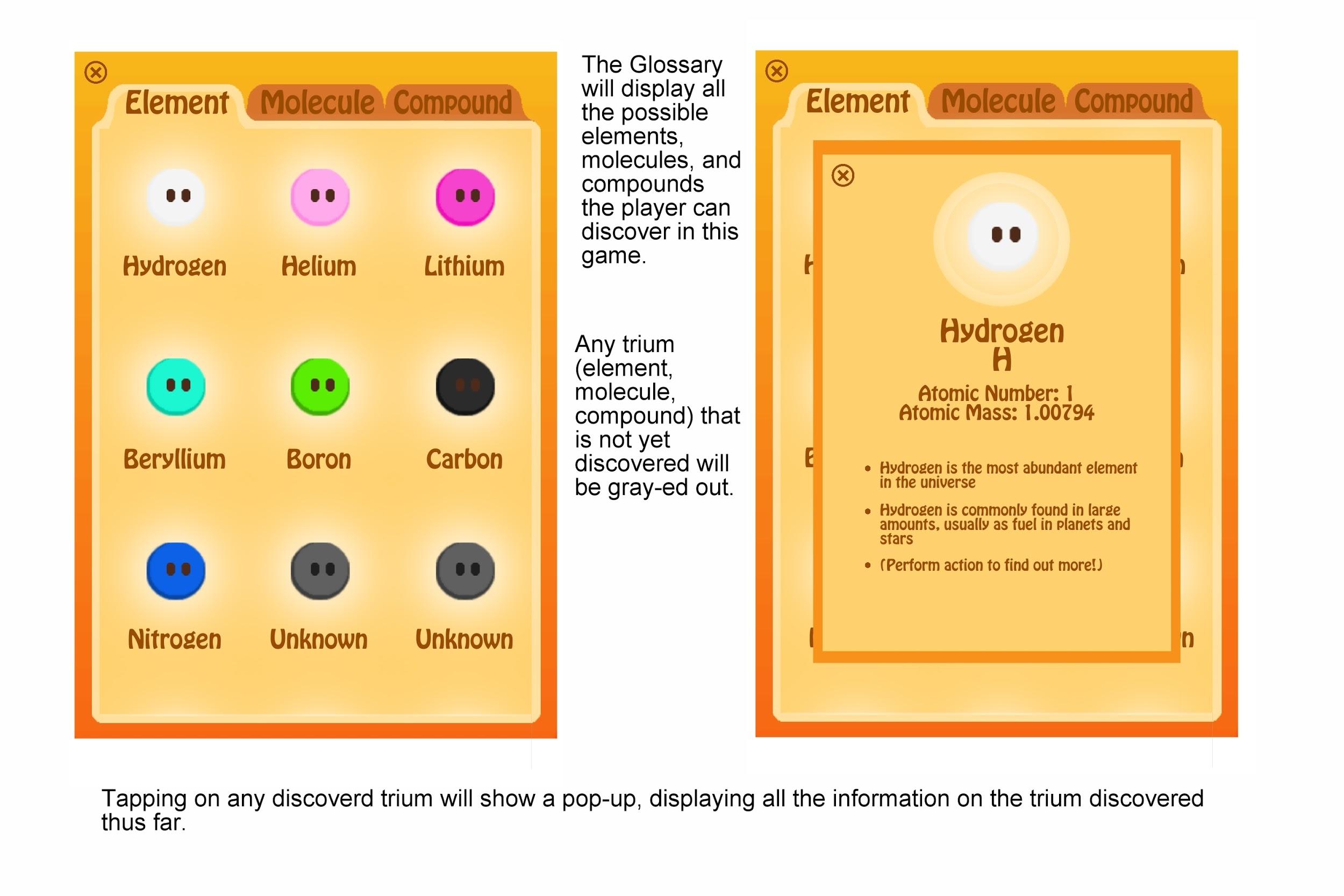
The image on the left is the main game screen. Tapping on the spiral or the Wormhole will spawn an atom of a random element. Tapping and holding on the Wormhole will allow the user to select a small array of elements, and the Wormhole will randomly generate one from the array.







Tapping on the Level will show a drop-down menu, allowing the players to view the “Glossary,” which contains all the information on introductory chemistry the player has unlocked thus far, or the “Achievements,” which displays the achievements the players have accomplished.





The Achievements menu will display a list of all the achievements. For the achievements that are complete, tapping on it will display a pop-up showing a bonus chemistry fact.

