# Data Team Recitation Questions

Davis Treybig
Sanmay Jain

1. What feature/design problems are you most excited to work on?

We are most excited to try to write the code for the storing and reading of data in as minimalistic a way as possible. It is really easy to write very verbose classes and code when dealing with data, but at the same time there is significant potential to create a structure of classes that makes reading from and writing to data both easy and extendable. This is an interesting challenge and something we are really going to focus on.

Related to this, we are excited to become more familiar with JSON. This is one of the primary ways data files are arranged in today's world and so it will be really valuable to understand how it works and how to implement it in a project.

2. What feature/design problem are you most worried about?

We are most worried about making our data storage and retrieval flexible enough to handle the fact that new features will be continuously added to the game and authoring environment. Ideally, we need to design a structure for storing data so that a new type of object or NPC or event can be stored with minimal added code. However, it will likely be difficult to do this well.

What makes this issue a particularly complex problem is that solving it requires communication and agreement between our team and the authoring team. Since the authoring team is handling the WorldData class, which is the overarching manager class from which data about a specific game will be stored, it is imperative that we agree on the structure of this class. With a defined structure, we will hopefully be able write code that can handle features being added or removed from that structure. Otherwise, we will have to constantly modify our code to accommodate changes the game and authoring teams are making.

3. What is your API designed to be flexible about

Our API is designed to be flexible about the specific details of the game. In other words, our data storage code and public methods should be able to be coded knowing *only* the basic structure of our RPG, regardless of the specific details of our RPG features that we change over time.

For instance, our RPG system is designed around the following ideas: a set of tiles - each with a specific image - which have no functionality of their own, and a set of objects - each with an image, a position, information about their movement, information about whether they begin any events, and other characteristics - with carry most of the functionality of the game. We want to be able to handle any arbitrary number of tiles and objects of distinct types, images, and attributes, without needing to change our code at all.

4. What are you encapsulating?

If we are flexible in the above regard, then we will be able to fully encapsulate all the details about data storage and retrieval. In other words, no matter what the authoring or

game engine teams implement or change, the details of writing information of a specific game to a JSON file and the details of reading information from a specific game on a JSON file will be known only to our sub team.

Related to this, we will not need to know very much about what the authoring and game engine teams are doing. By ensuring that the WorldData class is well structured and that the overall project team sticks with our concept of how the game will operate at a high level, all of the specific details of the RPG options we are implementing won't require any specific work from us. In other words, the other teams should be able to encapsulate much of their information and it should not be an issue for our data code.

5. How is your API linked to other parts of the project?

We are linked with two other parts of the project: the authoring environment, and the player. The authoring environment will call on us quite a bit in order to both load resources (such as languages and images) and to save the games have been created. As such, we will have a few public methods that the authoring environment calls when a user chooses an image to upload an image for a tile or for an object. We will also have methods through which the authoring environment can give us the current WorldData object, which manages all the current information about tiles, objects, and world information, so that we can store a specific game in a JSON file.

We will also have the converse of the above methods so that the authoring environment can load a previously stored game or get an image that has been stored in a resource folder.

Our main public interface with player will be giving them a saved or previously stored game so that it can be run. In addition, we would like to include a method or methods so that the current state of a given game can be stored. However, this would likely add a significant amount of complexity to our project and is something we are not sure we will be able to implement yet.

6. Briefly justify why you think your API is good

I think our API is good for three reasons: it is simple, it is flexible, and it is comprehensive. In regards to simplicity, in total we imagine we should have less than 8 or so public methods. Because of this, it should be very easy for the other sub teams to understand how to interface with us. This should require almost no knowledge on their part of the specific details of how we store and load information.

In regards to flexibility, as has been discussed, we believe that so long as enough communication happens so that the WorldData class is well structured, we will be able to implement code that functions regardless of the details that change in authoring environment or player. Our goal, which we believe will be accessible, is that we design our data classes such that we can handle any combination or collection of tiles and objects of arbitrary settings.

Finally, in regards to comprehensiveness, despite the fact that we will not have a very large number of public methods, we believe that our design will be able to handle everything that any other sub team will require of us.

7. What is the most important thing for us to implement.

The most important thing for us to implement given the goal of making the overall project as good as possible is designing a paradigm for taking a WorldData class and parsing its information in a way that allows it to be stored in a JSON file. Also important is designing a good system for handling the loading of images, the storing of images, and providing images to the authoring environment.