Saving/Loading Player’s Progress – Research

## *Using PlayerPrefs (Unity’s built-in option)*

* PlayerPrefs is a built-in Unity option to save game data. It is easy to use and can be implemented easily without additional APIs.
* It is great for saving simple data such as graphics settings, audio settings, etc.
* Compatible across different device types, no code tweaking required
* Data saved by PlayerPrefs is saved in:
  + On Windows – stored in registry
  + On Linux – stored in ~/.config/unity3d/
  + Windows Store App - \AppData\Local\Packages\[ProductID]\LocalState\playerprefs.dat
  + Windows Phone – Application’s local folder
  + Android – stored in SharedPreferences.
  + iOS - /Library/Preferences/[BundleID].plist
* In some instances, the file can be found easily and can be adjusted by the players; this makes the PlayerPrefs also not very secure if the developers are looking for a secure way of saving data (binary formatting might be a better way if security is key).

## *Using json.NET (saving to JSON files)*

* json.NET allows to save the data to .json (JavaScript Object Narration) files
* Unlike PlayerPrefs, json.NET allows developers to choose where the files are saved; this means that the location of the saves would have to be adjusted based on the devices (mainly difference between iOS/Android)
* JSON files, just like PlayerPrefs, are easy to open once found by users and the data can easily be adjusted by the players; this means that once again, if the developers are going for security, this is not the best option.
* Because of the way JSON files are formatted, it easy to check if the data is being saved on the go; the ability to modify the save data also allows the developers to quickly check if the data is being loaded properly.
* JSON files can be moved across multiple devices and can be easily read by the software. They are lightweight files, this means they can also be transported across the internet and used on other devices (for example, sharing same save file across multiple devices).

## *Using custom binary files (Binary Formatter)*

* Unlike the previous two options, binary formatters allow developers to create secure files. The files are stored in binary format, making them more difficult to read and modify (unlike previous two, which are easy to read and edit). Developers can also choose custom file endings.
* Custom binary files do not allow the same ability to check if data is saved properly (due to being saved as binary) and will take more time to debug and get working.
* As the systems get more complex, the code can quickly become messier; using this might be and overkill for a small project/game and using JSON files might a better option.
* If developers are looking for security, this is the best option to go, as it also allows them to customize and adapt this to their needs.
* Just like with JSON files, developers need to define where the files are saved and must make sure that that directory is compatible across different devices.

## Sources:

1. <https://www.newtonsoft.com/json>
2. <https://www.raywenderlich.com/418-how-to-save-and-load-a-game-in-unity>
3. <https://docs.unity3d.com/ScriptReference/PlayerPrefs.html>
4. <https://docs.microsoft.com/en-us/dotnet/api/system.runtime.serialization.formatters.binary.binaryformatter?view=netframework-4.7.2>
5. <http://wiki.unity3d.com/index.php/Saving_and_Loading_Data:_XmlSerializer>