06

Serilização de objetos

Discutimos durante o curso que a classe `JsonUtility` só consegue salvar informações que possam ser serializadas pela Unity. Mas o que significa isso?

"Serialização" é quando a Unity pega propriedades, atributos e objetos do nosso jogo e converte essas estruturas para um formato que ela consiga salvar e reconstruir mais tarde. Esse processo acontece internamente quando a Unity salva as informações que colocamos dentro do inspetor de um componente, ou quando salvamos a cena inteira.

Como estamos utilizando uma classe da Unity para transformar um objeto em algo que possamos salvar, internamente, estamos pedindo para a Unity serializar o objeto que passarmos por parâmetro para o método `JsonUtility.toJson()` e ao invés dela mesma salvar esse objeto, ela apenas retorna um texto no formato *Json*, que nós mesmos vamos decidir onde salvar.

Mas o que pode ser serializado pela Unity?

Quando falamos de classes, apenas classes que não são:

- abstratas;
- estáticas;
- genéricas;

  Podem ser serializadas, desde que estejam marcadas com o atributo `[System.Serializable]`.

```
[System.Serializable]
public class MinhaClasseSerializavel
{
    //Atributos e métodos
}
```

Dentro de uma classe, as propriedades serializaveis devem:

- Ser públicas;
- Não podem ser estáticas;
- Não podem ser constantes;
- Não podem ser de somente leitura(`readonly`);
- Devem ter um **tipo** serializavel;

No caso de propriedades privadas, podemos utilizar o atributo `[SerializeField]` para que a Unity ignore o modificador de acesso delas durante o processo de serialização. Se falarmos de coleções, as únicas coleções que a Unity consegue serializar são listas(`List<T>`) e arrays.