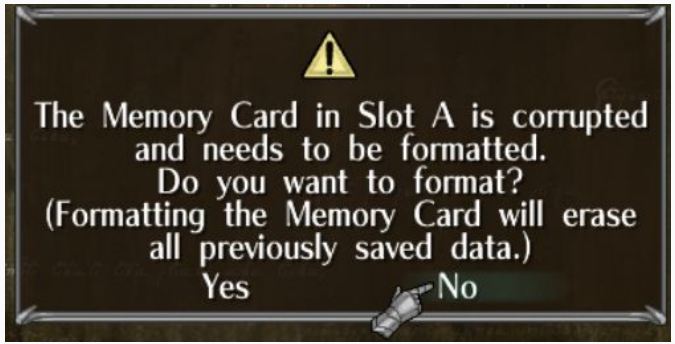
Game Development

Save & Load

Saving progression in video games

- Saving player progression is a core feature of nearly every video game
- During the last generation we still got external save devices
- Nowadays, modern consoles and game services save those on the cloud



The Request

We want to **serialize** our game:

1. The engine should be able to **write** to a file its **state**
2. The engine should be able to **read** a file saved previously
3. The system should be **easy to expand** as the engine grows
4. Should be simple to debug
5. Save files that are human readable

The Proposal

- Create a method for **load** and **save** for all modules.
- *App* will hold the core **load** and **save** methods.
- *App* should handle the creation of the file.
- *App* will create a section in the file for each module.
- *App* will make sure the save or load happens by the **end of the frame**.

The Test

In order to test the functionality:

- We will have a load happening when pressing “**L**”
- … and save when pressing “**S**” ***overwriting the previous file with a new one***
- The only information that we will serialize is camera position
- Check *solution.exe* in *Game/* directory

TODO 1

“Create methods on Application to request save / load”

- Very short methods, just to save the **intention** of saving/loading a file
- We will do the actual save / load later
- If we receive both in the same frame, in which **order** should we call them ?

TODO 2

“Create new virtual methods to Load and Save”

- Very similar to Awake method
- Think which argument each method should receive and *how*
- Call them at the end of the frame when it’s necessary
- Introducing mutable: use it with care!

TODO 3

“Call load / save methods when pressing l/s keys”

- Call Application Save / Load
- To read the keyboard, check how the camera is moved

TODO 4

“Create a new handmade xml that contains information about the camera”

- Very similar to config.xml
- Define how you will store the camera position
  - The renderer is the module that owns the camera
- Check “savegame.xml” file example from the solution

TODO 5

“Create a method to actually load an xml file, then call all modules to load their data”

- Start by opening the file as a xml_document (as with config file)
- Iterate all modules and call their **load** method
- As an argument send the xml section (as with config file)
- Make sure you print all possible errors using **LOG**

TODO 6

“Create a method to load the state. For now it will be camera's x and y”

- Read the data from the xml node you receive (as with config file)
- Then set the camera position

TODO 7

“Create a method to save the current state”

- First fill a pugi::xml_document calling all modules
- Use .append_* methods from pugi xml
- Finally save it to disk with xml_document::save_file()

TODO 8

“Create a method to save the state of the renderer”

- We just want to save the camera position
- Use *append_child* and *append_attribute*

```
pugi::xml_node cam = data.append_child("camera");

cam.append_attribute("x").set_value(55);
```

Homework

- Add a method in ModuleAudio to control the volume
- Change volume with +/- from the numeric keyboard
- Add default volume in *config.xml*
- Make the current volume to be saved and loaded