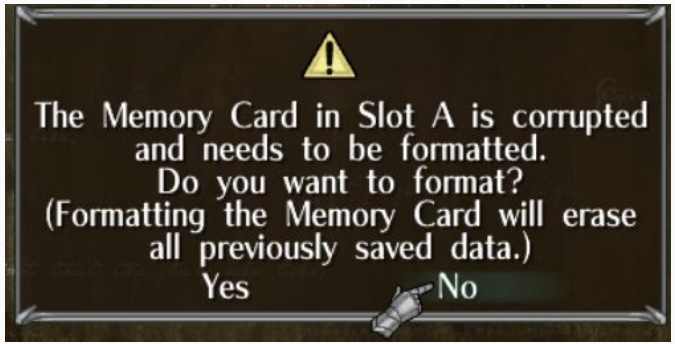
# Game Development

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

*“**Request** Load / Save on Application when pressing L / S”*

- Save the player’s **intention** of loading / saving the game state
- This is **not** the actual Load / Save, we will call them at the end of the frame

# TODO 2

*“Create methods on Application to save / load the game state”*

- Leave those functions empty for now, we will fill them later
- If the user has requested a save / load, call those functions at the end of the frame

# TODO 3

*“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

# TODO 4

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

- The xml is already given to you in the solution (“savegame.xml”)
- You are free to create your own file and store the data as you wish
- Define how you will store the camera position
    - The renderer is the module that owns the camera

# TODO 5

*“Fill the application load function”*

- Start by opening the file as a **new** 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

*“Implement the load method on the renderer. For now load 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

*“Fill the application save function”*

- Generate a new pugi::xml_document
- Create a node for each module and send it to their save function
- Use .append_* methods from pugi xml
- Finally save it to disk with xml_document::save_file()

# TODO 8

*“Create the save method on 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 the audio module 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