Game Save Manager

Progress Report

CMPS 4910, Fall 2017-Spring 2018

Nick Polach

gamedb.xml

- Contains many games
- Currently uses <name>, <notes>, and <linux>
- Extra tags are for additional features that will be added if time allows

```
<game>
    <name>Bastion</name>
    <notes> </notes>
    <cross> </cross>
    <locations>
        <windows> </windows>
        <linux>/home/{user}/.SupergiantGames/Bastion</linux>
        <osx> </osx>
    </locations>
</game>
```

dbtools.py

- Contains class “Database”
  - Loads XML database into Python dictionary
  - Able to create a cache file for the database
    - This cache is the serialized dictionary
    - Cache is recreated when XML database is updated
    - Loading cached database is slightly faster than loading from the XML database
  - Simple functions to query the database dictionary (not needed but easier to read/understand in code
    - Ex.
      - is_in_database(self, gameName) // returns True if game in database
      - get_saves(self, gameName) // returns list of saves for given game
      - get_notes(self, gameName) // returns notes for given game

formatools.py

- Contains functions for handling save file paths

- format_saves(saveString)
  - Takes file path(s) with placeholders and returns the formatted path
  - Ex.   /home/stu/{user}   ->   /home/stu/npolach

- process_saves(gameName, db)
  - Returns list of formatted saves for given game from given db

maingui.py

- Starts the main graphical interface
- Connects widgets to handlers
- Inserts games being managed into the list widget
- Other complicated things...

Finished so far

- Reading in XML file
  - Reads games from xml database into Python dictionary
- Process saves containing placeholders into something useable
- Some graphical interface functionality

Working on next (for next progress report)

- Backing up save files
- Restoring Backups?
- SQL Database?