Level File Format

* Big-endian file
* Begins with 6 bytes
  + First four bytes are the integer 0, stored in hex. This was supposed to indicate the level format version, but this seems to never have been taken advantage of
  + Next byte is a Boolean. According to the decompiled editor, this tells the game whether to use “special graphics” for this level
  + Next two bytes are a short int. This tells the game how many objects are in the level
* Object data begins at 00000007 with a single hex byte
  + 00 = Block
  + 01 = Spike
  + 02 = Pit
* Followed by two int variables
  + Each object’s X and Y position begins at its lower-left corner
    - Since the y position starts at the bottom of the object, setting an object’s y position to 30\*3 will result in it being in the 4th grid space from the bottom of the level
  + First integer is X-position, second integer is Y-position
    - Note: the pit object works a little differently. It still uses the same parameters, but the first is the beginning X-position and the second is the ending X-position. The game will treat this as a single object, and automatically generate a pit that spans the distance between the given x positions
  + X starts at position 810 (03 2A in hex), Y starts at 0
  + One grid space is 30 units
  + Some information on minimum distances:
    - The minimum distance between the very beginning of the level and first object is 27 grid spaces (810 units)
    - The actual level start (shown as a red line in the editor) is on grid space 6
* After all object data is finished, there is one more int
  + This is the level end’s X position
  + Unsure if this is a uint or a normal int
  + Defaults to 3015 (0B C7 in hex)
* According to the decompiled level editor, data after this is the level’s background color information
  + Next 4 bytes is an int, which defines how many times the background changes in the level
  + Next 4 bytes is an int, which is the X position of the background change
  + Next 4 bytes is the color ID of the change
    - 00 = blue
    - 01 = yellow
    - 02 = green
    - 03 = violet
    - 04 = pink
    - 05 = black
  + The editor seems to have support for loading a custom background (possibly an image file?) if the byte immediately after the bg x position is set to 01. If the byte is 01, the game attempts to read a UTF string from the level data, then load it. This functionality seems to be unused ingame
* The data after the background change information is for gravity changes
  + First 4 bytes is an integer set to how many gravity changes are in the level (in the decompiled editor, this is stored in gravityChangesCount)
  + Next 4 \* gravityChangesCount bytes are the X positions of all gravity changes in the level
* The data after the gravity change information is for the falling blocks fade effect
  + First 4 bytes is an integer set to how many falling block sections are in the level (in the editor, this is stored in blocksFallingCount)
  + Next 8 \* blocksFallingCount bytes are the starting and ending X positions of all falling block patterns in the level
* The data after the falling block information is for the rising blocks fade effect
  + This is formatted and handled identically to the falling blocks
* Music data is not stored within the .dat, instead it is placed alongside it in a file called music.mp3. Both the game and editor are hardcoded to load and play this file if it exists along the .dat, it is not referenced in the .dat itself
* Marks are also not stored in the level file, and are lost when the level editor is closed
* Placing rising blocks, falling blocks, and gravity changes seems to be impossible in the level editor

Example level file with 1 block at the bottom-left corner of the level:

* Offset 00000000-00000003: The level version, in our case 0
* Offset 00000004: Whether or not to use special graphics (set to 00 in our case)
* Offset 00000005-00000006: Number of objects in the level (in this case of one object, this is 00 01)
* Offset 00000007: first object type byte, since our object is a block this will be 00
* Offset 00000008-0000000B: first object X position, since our block is at the leftmost position, this will be 03 2A
* Offset 0000000C-0000000F: first object Y position, since our block is at the bottom of the level, this will be 0
* Offset 00000010-00000013: End wall X position, defaults to 0B C7
* Offset 00000014-00000024: Background change count, gravity change count, falling block count, and rising block count. Since there are none in this level, all these values are set to 0, and there is no data in between them, making for a total of 16 zeroes at the end of the file

TODO

* Calculate the exact data size of one object