Design Document for MP3
Alex Huddleston
CSCE410 - 900
Due: 6/26/17

PageTable
My implementation of the PageTable class used for this assignment was basically unaltered from what outline was given to us. In the constructor for the PageTable, I first initialized two unsigned long pointers to be the page directory and first entry page table respectively. These were both given 1 frame from the kernel_mem_pool structure. I calculated the address in memory that these pages were associated with by shifting the frame number by 12, since these will be allocated using direct-memory mapping.
The constructor proceeds to fill out the first page table entry with present read/write bit entires for each 4kb entry in the table, then adds the address of the page table as the first entry in the page directory and marks it as read/write and present. Finally, the constructor finished up by labeling the remaining page directory entries as read/write, but not present and assigns this page directory to the static page_directory member of the class. For safety, I make sure current_page_table is set to the PageTable object that calls the constructor, since most likely the PageTable being initialized is the one we will want to be working with.

load(), enable_paging()
Since the instructions do not say to mention these, I will not be going over them. They should be implemented exactly as we were instructed to implement them regardless.

handle_fault()
This function gave me the most trouble. First I have the function initialize some variables to reference the current page directory and relevant page table to the page that has caused the page fault. I also initialize a couple variables to hold the indexes of the page directory and tables given by cr2 that are associated with the address that caused the fault.
Next, I check to see if the page table that caused the fault is present. If not, the handle_fault() function creates a new page table allocated by a frame from the kernel_mem_pool. The process of adding a new page table is almost identical to the way it is done in the PageTable constructor, so I will not elaborate on it.
Finally, if the memory access is above the 4MB address limit, this is where handle_fault() would mark a needed page as present and allocate a frame from the process_mem_pool to associate with the game. However, as I have outlined in the program using some comments, this part is written out, but not activated. Due to some issue with process_mem_pool, I cannot make a call to get frames from the pool without the program crashing entirely. This happens regardless of whatever interrupts I try to put in the program. Since I cannot determine if this is because I did something wrong with implementing my cont_frame_pool program or whether I missed something in the documentation that informs me of such an error, I've decided to leave it commented out. This way the program will still run and succeed for direct memory-mapping tests.