The purpose of this project is to create a game that demonstrates competency and comprehension of standard programming concepts and create a functional program that meets the given requirements. The current build of *The Schism* utilizes an object-oriented approach and has a functional combat system, which goes beyond the given requirements. I took this approach to challenge myself and learn how to manage instances across modules and classes and as a personal challenge. If this deviation from the assigned requirements is unacceptable, the combat system can be made entirely optional. The current alpha build does not have an adequate win/lose check other than player life. One planned feature for the next alpha is to code a choice at the opening screen to play with or without combat. Choosing with combat ties victory conditions to defeating the final boss; choosing without combat ties victory conditions to item collection and room location.

When contemplating a game interface, the need for something robust yet flexible yields a desire for something beyond while loops. There are several built-in options for command prompts, and the first that came to mind is the simple cmd line interface used to build those little turtle games so long ago. The Cmd class of the cmd module in the common library provides an excellent framework on which to build. The [common docs](https://docs.python.org/3/library/cmd.html) have more details on this class and the methods therein, which pass some powerful functionality. Cmd has a method cmdloop, which handles a continuous prompt until a True value is returned to the onecmd method, which is the method that interprets the user input. Thus, the Schism’s command line interface was born.

Player input will be handled by Cmd, however the action methods need to be defined. Action methods are prefixed with **do\_**; if the name after the underscore matches user input (line from cmdloop), then that **do\_** method is executed. Thus, the input validation is handled automatically and the **do\_** method is akin to the IF in a standard while/ if loop. There are two primary areas in which action methods are necessary – the core game loop and the combat loop. Please see Full Game Logic and Room Movement Logic for details on output and flow. The action commands are as follows:

**Core Game Loop**

|  |  |  |  |
| --- | --- | --- | --- |
| do\_north | do\_east | do\_south | do\_west |
| do\_help | do\_status | do\_look | do\_bag |
| do\_search | do\_quit | do\_fight |  |

**Combat Loop**

|  |  |  |  |
| --- | --- | --- | --- |
| do\_attack | do\_item | do\_run | do\_help |

Items:   
 There are several items in the game. Fourteen can be found in various rooms, the remaining are only accessible through the combat system. As of this writing, there are nineteen total items in the stable alpha build. Items are as follows:  
**Findable** :

|  |  |  |  |
| --- | --- | --- | --- |
| Apple | Armor | Boots | Sword |
| Bucket of Water | Chitinous Gauntlets | Dusty Book | Tome of Power |
| Feather | Ornate Greaves | Palantir | Salted Pork |
| Potion | Orb of Con-Creet |  |  |
|  |  |  |  |

**Lootable:**

|  |  |  |  |
| --- | --- | --- | --- |
| Goblin Helm | Great Axe | Royal Pauldrons | Light of Elune |
| Mithril Chainmail |  |  |  |

**Rooms:**

There are several rooms contained within a modular system built on JSON dictionaries. This affords an easy way to add more rooms without modifying code. As of this writing, there are 12 rooms in the stable alpha build. Please see the supplemental Game Map.pdf. Rooms are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 The Culvert | 2 The Storeroom | 3 The Guardhouse | 4 The Armory |
| 5 The Rookery | 6 The Wizard’s Library | 7 The Secret Passage | 8 The Chapel |
| 9 The Shattered Tower | 10 The Schism | 11 The Great Hall | 12 The Dark Passage |

[The stable build can be found here.](https://github.com/Ivy0789/The_Schism-Alpha-)