THE 1563 WORD PLAYER MANUAL FOR BANGKOK MEATSPACE

INTRODUCTION

Bangkok Meatspace is a parser text adventure in the style of interactive fiction (IF) that have been around for a while. IF had its commercial heyday in the late 80s, but it still has an enthusiastic following now (see IF Comp 2016). It can take many forms apart from the text parser format that has been ubiquitous in the world of IF since the 90s. Recent development tools such as Twine have enabled enthusiasts to easily create hypertext interactive fiction, where the player still interfaces with the narrative through written prose, but instead of typing an action in a command line interface (CLI), they use the mouse to click on hypertext links in the form of words integrated in the prose. These hearken back to the original interactive fiction: physical choose-your-own-adventure books in which the player interacted with the narrative by physically turning to the respective page of their decision, and then continuing the story.

However, the sole focus of IF has not always been narrative. When the text parser was introduced, the world model and CLI afforded parser-based interactive fiction to be more puzzle focused. The player had to reason out the next course of action instead of selecting an option from a displayed list of hyperlinks, or guess the correct verb, or combination of words, to use. In fact, early parser IF like Zork were characterized more by the puzzles that made up the meat of the gameplay rather than the narrative present in the writing.

The most holistic description of interactive fiction is that it is a novel in which the reader has agency that is afforded to them through the technology of the computer. Even though the player has the power to decide the course of the story, the computer code ultimately has control. Digital storage allows multiple variables to be stored, multiplying the number of end game states beyond what is afforded by a printed choose-your-own-adventure.

Invariably, the main mode of interface with the narrative is through written language (although some IF has audio or images or graphical user interfaces to enhance the experience). Language is an organic creature of its own, with syntax and forms that can be illogical when compared to the way a computer functions. To add better flow to the user experience, the parser must make efforts to reconcile its own digital nature and the analog nature of language. Existing IF parser software make up for this somewhat, with some sophisticated enough to understand full sentences, such as ‘put letter in mailbox and then close it’.

Though it aims to feel organic in its storytelling, IF nevertheless is still a command line interface with computer code, and part of the “fun” for text adventure players is puzzling over sophistication of the parser; whether to use ‘use teabag on pot’, or ‘put teabag in pot’.

Over the course of its existence, parser IF has developed its own tradition and conventions that are expected in each parser game.

IF CONVENTIONS

```
>go south
>south
>s
```
The world model of the parser text adventure is comprised of separate rooms that are linked by directions. To navigate to the room to the player's south, the player would enter "go south". It is commonplace in the IF community to shorten this command to "south" (foregoing the 'go'), or just "s". IF traditionally uses the cardinal directions of north, south, east, and west, as well as any combination such as southwest or northeast, and sometimes up and down.

```
>look
```
In these rooms, objects may exist for the player to interact with. To have a general view of the room the player is in, the player would use 'look' or the shorthand 'l'.

```
>examine
>look [object]
```
If the player desires further examination of a room's item, they would enter an object after the 'look' verb. For example: 'look lamp' would return the author's prose describing the lamp. Note the shorthand 'l' would work in place of 'look' in every case, ie 'l lamp'.

Another verb conventionally found in parser IF that serves the same purpose is 'examine'. 'examine lamp' would return the same message as 'look lamp'. The two verbs are almost synonyms; however using the verb 'examine' on its own would not be the same as using 'look' on its own. Instead, it would return some variation of the parser asking the player, 'What do you want to examine?'

```
>take [object]
>get [object]
```
Another synonym is used when the player wants to take an object. They would enter: 'take lamp' or 'get lamp'. Both examples perform the same function, removing the lamp object from the room and adding it to the player's inventory.

```
>inventory
>inv
>i
```
To take list of the player's inventory of collected world items, they would use the command 'inventory', often shortened to 'inv', or just 'i'.

```
>tell [npc] about [subject]
>ask [npc] about [subject]
```

Apart from rooms and objects, players are typically able to interact with non-player characters, or NPCs. This poses more of a challenge for the parser, as conversations in the real world and in prose can veer into any given direction. The linear nature of computer code, and the fact that the IF writer must write every single line of possible dialogue, limits the player to ‘ask’ing or ‘tell’ing an NPC about a subject, usually a keyword, eg. ‘ask professor about physics’.

This should give you a rough idea what interaction is like between human and computer in a parser-based text adventure. Other common commands include ‘drop’, which enables a player to drop an item from their inventory into the player’s current room, and ‘wait’, which allows in-game time to pass.

NAVIGATING THE MEATSPACE

Bangkok Meatspace aims to pull some of the cover off the interaction between human and machine, and exposing the inorganic harshness that is normally glossed over in the conventional text adventure.

First the game requires the user, with a python interpreter preinstalled, to open a terminal with a command line interface (such as the Windows cmd), navigate to the directory where the game file is stored, and then open it. The player is then greeted with a short introduction and an ASCII title screen. It quickly becomes apparent that Meatspace is not quite your conventional parser IF. Instead of having rooms that the player navigates, the world model of Meatspace is set up like a directory tree of a computer’s file system. The player no longer navigates through ‘n’, ‘s’, ‘e’, and ‘w’, but “up” a file directory (‘goto exit’), or “down” into a subdirectory (‘goto [subdirectory]’). The player must always use the ‘go’ verb as shorthands do not exist. Note that the ‘go’ is replaced by ‘goto’ (without a space in between), and using ‘go to [subdirectory]’ produces an error in the game. This same design is echoed in its other commands, which must be typed in a very specific way reminiscent of actual computer line commands, or else be unrecognized by the parser.

Any time the player enters the incorrect syntax into the CLI, a jolting Windows error message pops up, stalling the flow of the game. This breaks the form of the text adventure, which is traditionally contained on the white “page” of the parser window. In typical IFs, misusing a command returns a friendly message from the parser in the same style as the prose.

Meatspace emulates the unforgiving feeling of using a computer’s command line interface and reveals that the player is not interfacing with the subtle nuance of language, but rather the harsh digitalism of computer code. This harshness is reflected in the core gameplay loop: Every player action costs the player’s in-game energy, even the ‘about’ actions which provide the player more information about the game world. As the player’s energy depletes, their hunger meter increases. If either of them are mismanaged, the player character will either pass out, or die of starvation. To combat this inevitability, the player’s only option is to spend in-game money to buy various items that the player can consume to add energy or remove hunger. On every

initialization of the game, the player may have a random amount of money ranging from 100 to 1000. Once again the computer imposes its control on what the player is able to do in the game.

If the silicon world of ones and zeros makes up “cyberspace”, the analog world of bones and flesh would be the “meatspace”. Although the player character is nominally navigating the meatspace of New Bangkok, the player is actually navigating cyberspace, the representation of the computer’s internal code, but transformed into a pseudo-organic structure. In breaking the flow by having annoying pop ups along with having an unforgiving and inorganic parser, Meatspace draws attention to the player’s own limited affordances and the control the computer has over every aspect of the interaction. The form in which the IF unfolds adds to this message; though the content is archetypal of other parser IF, it forgoes the clean lines and oftentimes graphics of an IF interpreter like Parchment, for the rawness of a cmd terminal.

FULL LIST OF COMMANDS

```
>goto [subdirectory]
>goto exit

>get [item]

>drop [item]

>about
>about [item]
>about [subdirectory]
>about [npc]

>list
>list [prices]
>list [inventory]
>list [subdirectories]
>list [directory]

>buy [item]
>eat [item]
>drink [item]
>smoke [item]

>talkto [npc]

>sleep

>wait
```

```
>die

>quit

>help
```