Camp X-Ray: A Game about Interrogation

December 8, 2014

Abstract

A nonlinear text game (also called "interactive fiction") about interrogation. Programmed in C++ for CSC17A, Dr. Lehr, FAL14.

Bradley Evans
December 9th, 2014

Introduction

This game is about interrogation. The user is able to ask the suspect questions based on previous dialogue. The entire game is based on interaction with this solitary NPC and takes place exclusively in the interrogation room. It is inspired by a work of interactive fiction called Galatea.

Concepts Demonstrated

- **Template Class:** Demonstrated in class Dice, which is located in Dice.h and Dice.cpp. Utilizing an enumerated value DiceType (a nonprimitive data type) with the template is an option.
- **Abstract Class:** Demonstrated in class Suspect, which contains a virtual function virtual void showStats();
- **Derived Class:** Demonstrated in class Game, which is derived from class Suspect.
- **Base Class:** Demonstrated in class Suspect, which has Game listed as a friend class.
- **Regular Expressions:** Demonstrated in class ParseCmd located in ParseCmd.h and ParseCmd.cpp. Named capture groups are also used in conjunction with the Boost library to simplify the parsing of user-entered commands which are stored as strings. Regular expressions are also used to parse out dice types from a given string in class Dice. A program called Expresso was used to aid in the writing of regular expressions used in this program.
- **Binary Input/Output**: Demonstrated in cmdtree.cpp, in the cmd_save_game and cmd_load_game functions. These functions collect game conditions from throughout the game, primarily from the Game class and the gamestate struct, and save them to savegame.bin. Also utilizes a **one dimensional dynamic array** with a **pointer.**

- **Test Input/Output:** Demonstrated in printLine located in cmdtree.cpp. Outputs a particular line of a file depending on game conditions. Also utlizes a **two dimensional array** with pointer.
- Other concepts used:
  - **Maps:** A map is used in classes Noun and Verb to connect strings parsed from ParseCmd's RegEx named capture groups to enumerated values listed in EnumNoun and EnumVerb, which are themselves **enum class** types. This usage of maps was originally inspired by a writeup found at http://www.cplusplus.com/forum/general//119658 but has been modified significantly to function within the framework of this game.
  - **Structures:** A structure is used in gamestate.h called struct gamestate that stores conditions throughout the game, such as if a particular action has already been taken or how much time remains in the game.

Program Flow

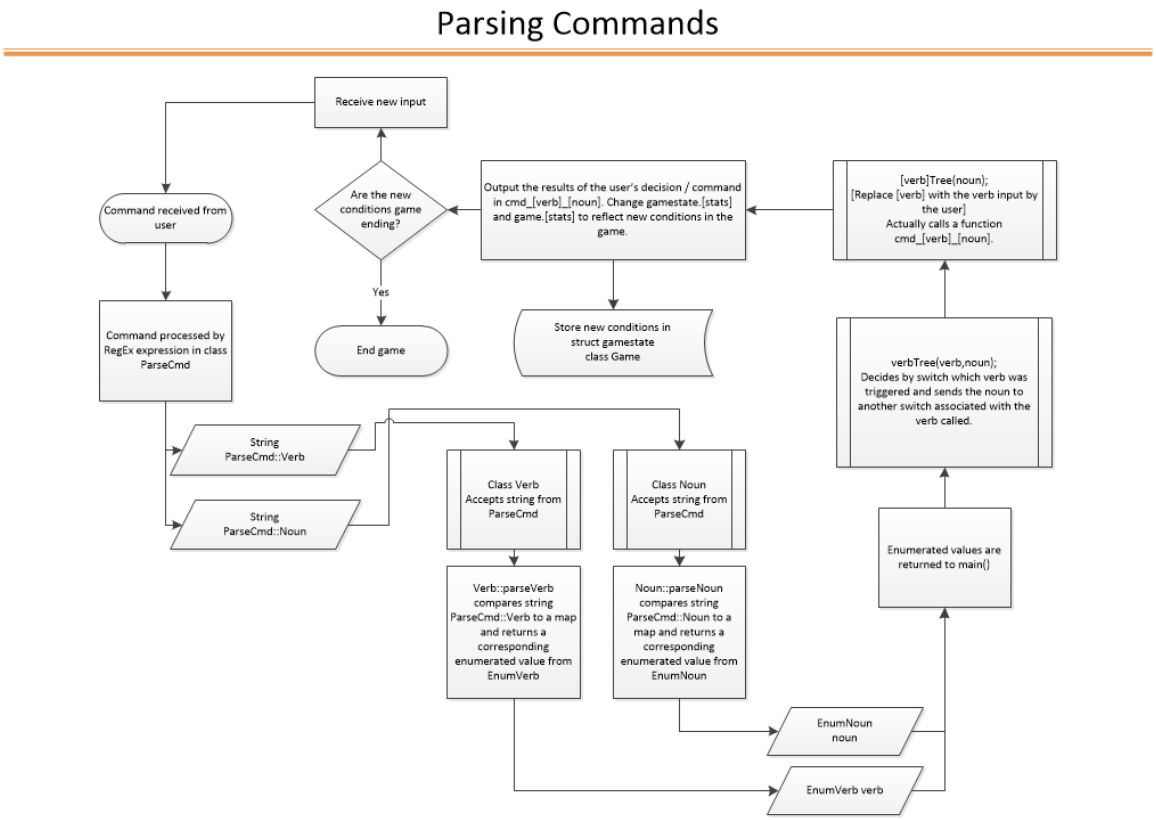
Initial run of the program begins with the displaying of an introduction and the setting of initial states in the game. This is done via class constructors in Game and via commands stored in the intro() function located in cmdtree.cpp. After the introduction is played, the user is invited to enter a command by the main function parse(). Valid commands are two words – a verb followed by a noun – such as "ask name."

After the user enters the command it is stored as a single string, at which point it is sent to ParseCmd's constructor. A regular expression with named capture groups divides the command into two strings – a noun and a verb – and ParseCmd::getVerb() and ParseCmd::getNoun() are used to return the verb and noun, respectfully. These are then passed to the Verb and Noun classes by constructor, and the class functions Verb::parseVerb() and Noun::parseNoun() utilize maps to convert these strings into enumerated values, EnumVerb verb and EnumNoun noun respectfully. If the string for either does not exist as a valid value in either tree, these functions will return EnumVerb::invalid or EnumNoun::invalid.

From there, EnumVerb and EnumNoun are passed by value to verbTree(), which first detects if the verb was valid in the first place. If it was valid, a switch statement passes EnumNoun to another switch statement appropriate to the verb. For example, if the command "ask name" was input, then the switch statement would see EnumVerb::ask passed to it, and would then pass EnumNoun::name to a function called askTree(EnumNoun).

The new tree detects if the noun is a valid pairing with the verb used. For example, if the command "look name" is entered, verbTree would pass EnumNoun::name to lookTree, which woud discover that looking at a person's name is unproductive and return a sort of error (cout << "There isn't much to see there.";). Assuming the noun is valid, it will run a function based on that verb and noun pairing. If the user entered look suspect, verbTree would send the noun suspect to lookTree, which would then run the cmd_look_suspect function. This function is what I call the function for the actual command entered.

As this function runs, game conditions are altered to suit the player's decision. The clock is decremented, certain statistics are changed (such as "trust" and "exhaustion"), and the game loops back to recieve a new command for as long as the clock still has time remaining.



Game Mechanics

The interrogation suspect is governed by a few simple primary statistics.

- **Intelligence** helps govern the suspect's ability to reason, recognize lies, respond to "deals," and other things.
- **Deception** is the suspect's ability to form lies or otherwise impede the investigation. Higher deception scores make the suspect more likely to fail to respond to questions or outright lie.
- **Loyalty** is the suspect's ideological desire to not betray his compatriots. Higher scores enhance the suspect's likelihood to decieve the interrogator.
- **Honesty** is the suspect's propensity to lie. High scores make unsuccessful outcomes more likey to be stalling actions or other actions that aren't necessarily lying.
- **Exhaustion** is self explanitory. As the interrogation proceeds, this statistic grows, eroding his ability to impede the investigation.

  - Derived Statistic: **Pain**. If physical, coercive force is used, a stat stored in gamestate.pain is iterated. Pain beyond a certain level implies the suspect is being tortured. Too much pain negates trust and honesty bonuses and produces a much more hostile suspect. The suspect is less likely to use stalling tactics and more likely to outright lie to drive the investigation forward.

- **Trust** reflects the rapport between interrogator and suspect. High trust levels make the suspect more cooperative and less likely to impede the investigation in any way.

As the user enters commands, dice are rolled, usually twenty sided dice, against a difficulty level defined by some combination of these statistics depending on the question asked. Outcomes are generally either cooperation, silent noncooperation, and lying.

Gameplay

The user may ask any question he can concieve of in the form of a verb followed by a noun, such as "ask name." If the command is invalid, the game will say so. To assist the user, valid nouns and verbs that appear in dialogue appear [bracketed]. For example, the dialogue "The [suspect] appears to have a faint [accent] that sounds British in origin" indicates that [suspect] and [accent] are valid nouns. Hitting upon a valid command – usually a valid question to ask – will decrement a clock by one "minute." When the clock reaches zero, the suspect is said to be "transferred" to the jurisdiction of a different authority and the interrogation ends.

Besides asking simple questions, the user can also use "enhanced interrogation" methods, but these come with their own cost. Even modest use of enhanced interrogation will get answers more quickly out of the suspect, but the likelihood of the suspect's answers being lies or otherwise hostile increases as well.

As a work of "interactive fiction," the object of the game – as it is in normal interrogation – is to hear as much of the suspect's story and pan as much truth as possible from the limited converstation. The user is left to infer on his own what might be truth and what might be lies.

Sample Output

```
================================================
CAMP XRAY: A Game about Interrogation
================================================
You observe your [suspect] through a grainy webcam image in a small window on your laptop screen. Th
You have been assigned as this subject's interrogator. He's being processed for transfer, at which t
================================
Suspect Statistics (Debug Mode):
Intelligence: 3
Deception: 2
Loyalty: 4
Honesty: 0
Exhaustion: 4
Trust: 0
================================
```

```
Time Remaining: 15
> look suspect
He sits upright, watching you intently.
> ask name
"Let's try something simple," you ask. "What is your name?"
"My name is Hamad Anwar," the man says softly. His voice has a trace British -- maybe South African
> ask name
"Let's try something simple," you ask. "What is your name?"
"I have already told you my name," he says, in his out-of-place [accent]. Hamad watches you intently
"Why don't you just humor me. Tell me again," you insist.
"Hamad Anwar, as I just said," he repeats. "Aren't you in a [hurry]?" he says mockingly, "Maybe you
> hit suspect
You decide that a little 'enhanced interrogation' is in order and move to hit the suspect in the che
> ask hurry
You're not really sure how to ask about that.
> hit table
You're not sure that hitting that would be terribly productive.
> look chair
Not much to see there.
> show debug
==============================
Suspect Statistics (Debug Mode):
Intelligence: 3
Deception: 2
Loyalty: 4
Honesty: 0
Exhaustion: 5
Trust: -1
==============================
Time Remaining: 15
> ask name
"Let's try something simple," you ask. "What is your name?"
"I have already told you my name," he says, in his out-of-place [accent]. Hamad watches you intently
"Why don't you just humor me. Tell me again," you insist.
"Hamad Anwar, as I just said," he repeats. "Aren't you in a [hurry]?" he says mockingly, "Maybe you
>
```

As time runs out

```
> ask name
"Let's try something simple," you ask. "What is your name?" "I have already told you my name," he sa
"Why don't you just humor me. Tell me again," you insist.
"Hamad Anwar, as I just said," he repeats. "Aren't you in a [hurry]?" he says mockingly, "Maybe you
The civilian reenters the room, glancing around with a look of contempt. "You're done. He's our pris
Two other men enter, cutting the man's restraints and bringing him to his feet roughly. "It's been f
```

Other Notes

Known Issues

The game's command set is fairly limited.

Valid Commands

A list of recognized verbs and nouns can be seen in the source code in Verbs.h and Nouns.h. The current commands with valid functions associated with them are:

- look suspect – Gives a dynamically generated description of the suspect.
- ask name – Simple command to find the suspect's name.
- [hit/punch/kick/strike] suspect – "Enhanced" interrogation
- show debug – Outputs key variables that drive the game.
- save game – Saves the game
- load game – Loads the game

Code Metrics

- Lines of Code: 1015
- McCabe VG Complexity: 57
- Comment Lines: 165
- Source Files: 17