Web Development – Mr. Turner

Project – Text Adventure Game Engine

Project Overview

Create a web page for building Text Adventure Games. Text Adventures are games that are essentially interactive books. They contain multiple story blocks and feed new blocks to the player based on the commands the player gives to the game.

Display

The Engine should have two modes of interface. The first would allow a user to build an adventure. The second would allow for the user to play through and test an adventure.

Building An Adventure

The Build Screen will have the following sections available to the user:

- Adventure Name - The user must name the adventure.
- Story Block Name - The user should name each story block.
- Story Block - The user may type in a story block.
- Targets - Targets are items in the text (see the functionality below).
    - The user should be able to add targets
    - The user should be able to remove targets
    - The user should be able to assign what happens when each command (see functionality) is used on a target. The user should be able to type a response, access another story block, or select a default response.
- Save a story block - there can be a button that does this.
- Load a story block for editing - There should be a list of all of the story blocks available.

The Play Screen will load the first story block from the adventure. It will have the following sections:

- The story block text.
- A text box where the user can enter commands.
- A log of the user's commands and the game's responses.
    - New entries should be at the bottom of the log and the log should continue to scroll down as the game continues so that the user can see the last entries at all times.

Functionality

What is a Text Adventure?

A true Text Adventure is more than a simple Choose Your Own Adventure story book. The player, while limited in the number and types of moves they can make, is not given choices. Instead, the player must puzzle out the proper commands for each area in the game.

This interface also presents a problem that does not affect writers of Choose Your Own Adventure books. That is, an area can change based upon the actions.

For example, read the following excerpt from a played Text Adventure…

> *You are standing in the Administrator's Office. The office is a keen reflection of the man, lavishly furnished and decorated with pictures of the Administrator and awards given to the Administrator. The room is specifically designed as a tool of intimidation so that anyone who may sit across from the Administrator in his own office will be inundated by his successes. Though there are many items placed about the room, your eye falls upon his datebook laying open on the desk.*
>
> *What would you like to do?* ***take datebook***
>
> *You take the datebook.*
>
> *What would you like to do?* ***look office***
>
> *You are standing in the Administrator's Office. The office is a keen reflection of the man, lavishly furnished and decorated with pictures of the Administrator and awards given to the Administrator. The room is specifically designed as a tool of intimidation so that anyone who may sit across from the Administrator in his own office will be inundated by his successes.*

Note that once the player has taken the datebook, the description of the room is modified to reflect that action. Should the player leave the room and then return, it would be very important that that modification remain in effect.

In addition, if the player ever wanted to look at or use the datebook, the game would also need to be aware that the player has the datebook.

These two circumstances, more than any other, separate the Text Adventure from the simple Choose Your Own Adventure story and need to be carefully considered when designing the engine.

The Build Interface

Each adventure will be built as a series of text files. As soon as the player saves the first story block, a folder with the name of the adventure should be created. Create a text file for that adventure that lists the name of the first story block.

Each story block will also have its own text file. That file is going to contain:

- The story block text
- A list of targets
  - Each target will have a series of responses that corresponds to a command.
  - A list of commands is included in the player interface section of this document.

There are 2 types of responses to a command/target pair.

- A text response - the command and the response are added to the log
  - The text response should also contain keywords that indicate whether or not an item is lost after use (it can be something as simple as “the item is lost”.
- Loading a new storyblock - the command is added to the log and the new story block is loaded.

Your engine should include a default response. For example, if the user types in **climb apple**, that probably won’t make sense within the context of the game. A good default response is something like, “I don’t understand you” or “You’re just not making any sense”.

There should always be a list of the targets associated with the story block and a text box that allows the user to add new targets. The user will be editing one target at a time and it should be very clear which target is currently being edited.

When the user creates the target, they should see a list of the commands and be able to type in the responses for each command. The target and its responses should be stored as an object in memory. Only when the user saves the story block is the whole thing saved to the text file.

A user can load a story block and edit it. If the user loads the story block, populate the necessary fields with the information about the story block. Once the story block is loaded, the interface is the same as if it was a new story block. The user will have to save the story block in order to complete the edits.

The Player Interface

The basic interface of a Text Adventure will include listing out a storyblock on screen and prompting the user for an action. The user's actions will be based on a string including what will be called a command and a target (there are exceptions to this rule).

The command is the actual action that the user is taking. These must be pre-defined and are listed below. In the above example, take and look are considered commands.

The target indicates some object within the storyblock upon which a command can be executed. The office and the datebook are examples of targets. Targets are defined by and within the individual game so there's no need to worry about them when creating the engine.

Listed below are the commands that should be programmed into the engine. Some commands will not require targets (although may take them if necessary). Any command that does not require a target is listed in italics.

● attack	Attack another creature or, even, an object
● buy	Buy something from someone.
● climb	Climb an object.
● close	Close something that is closeable (door, chest, etc…)
● drop	Drop something from inventory.
● get	Get something from the area (same as take).
● go	Go in a direction or to another area.
● help	Show instructions for the game.
● hint	Show a hint for the area.
● inventory	Show a list of the items in the player's inventory.
● look	Look at something - get a description (with no target, assumes area).
● open	Open something openable (door, chest, etc…)
● pull	Pull something.
● push	Push something.
● put	Put something from inventory down or into something else.
● search	Search for hidden objects (with no target, assumes area).
● sell	Sell something to someone.
● speak	Speak with another creature or object (same as talk).
● take	Take something from the area (same as get).
● talk	Talk with another creature or object (same as speak).
● throw	Throw some object from inventory.
● use	Use some object from inventory.
● wait	Wait for something to happen.

The Player is the most basic of the objects in the engine. Essentially, the Player can be as simple as a name and a list of items in inventory. The items in the inventory list need not have any functionality. Different story blocks will be written to respond to the use of certain items. As mentioned, they have no special abilities. The engine will simply want to check whether or not the player has the item (s)he is attempting to use/throw/look at/etc… The storyblock itself will take care of the rest.

On a side note, an item should be labeled as reusable or not so that the item can be removed from the player’s inventory if it is a 1 use only item.

It will be necessary to implement a way to add items to the inventory as well as a way to remove them.

The Storyblock is far more complex. The base piece of the Storyblock is the text that describes what's happening in that part of the story. More importantly, though, a storyblock needs to know how to react to the player's commands.

Every Storyblock object must define a series of targets. The Storyblock doesn't list these targets, but they must be visible (or semi-visible) in the text.

Note the following example:

> *As you step into the lobby, the front door swings closed behind you. The lobby is a large room, surprisingly empty. All that you can see is the reception desk with the receptionist sitting behind it. To your left is a short hallway that leads to the elevators.*

What are some of the targets in the Storyblock?

lobby, front door, door, room, reception desk, desk, receptionist, left, hallway, elevators

Each of these words in the Storyblock indicates something upon which the player can act. The player may choose to type *look elevators* in order to get a better view of that area. The computer's response could be anything from *You can't get a good look from here* to a detailed description of that area. That's up to the adventure's author.

Also note the similarities in some of the targets. The targets **front door** and **door** refer to the same thing. An author may want to list multiple variations on a target in anticipation of the player's commands. If every last target has only one reference, the game can become extremely frustrating for a player. It becomes more of a cat and mouse game with the player spending more time trying to figure out the author's references than getting through the game.

But again, that's up to the author. This is not actually a game, but a game engine. What the engine needs to do is have a convention for accepting unlimited targets and handling those targets based on each command. In order to do this, and account for the changing environment, the engine must implement a multilayered list of responses.

Each Response must have a list of targets which is created by the author of the adventure at the time the adventure is created. The engine just maintains the convention for using the list. When the user enters a command and a target, one of 2 things will happen.

1. The system will print out a simple text response.
2. The system will load an entirely new Storyblock.

Either way, the system will be responsible for handling changes to the environment. It will be up to the author of the adventure to *write* the test that describes those changes, but they must be organized by the program. It's up to the engine to make sure that the links can be set up properly so that the right story portions get loaded at the right times.

Enhancements

- Allow the user to create their own commands and add them to the command list. This will require an interface in the build section.
- Allow a player to save their progress in the adventure so that they can come back to it later on. This would require that all of the important information for the story be saved including the user's information and the story block's information.