Letter Lizard

Project Status Presentation

March 10, 2014

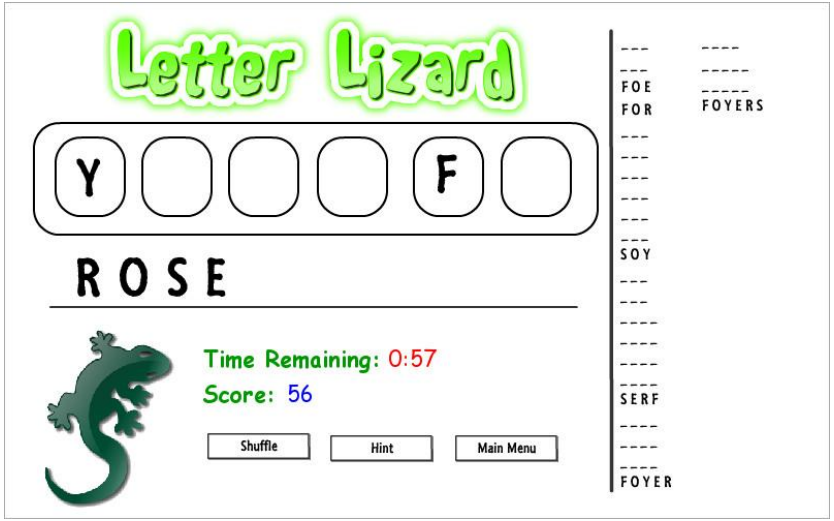
Afiya Nusrat, Alexander Pokluda and Michael Wexler

Outline

- Recap of Letter Lizard gameplay, features and configuration
- Project Status
  - Game generator
  - Python implementation
- Demo

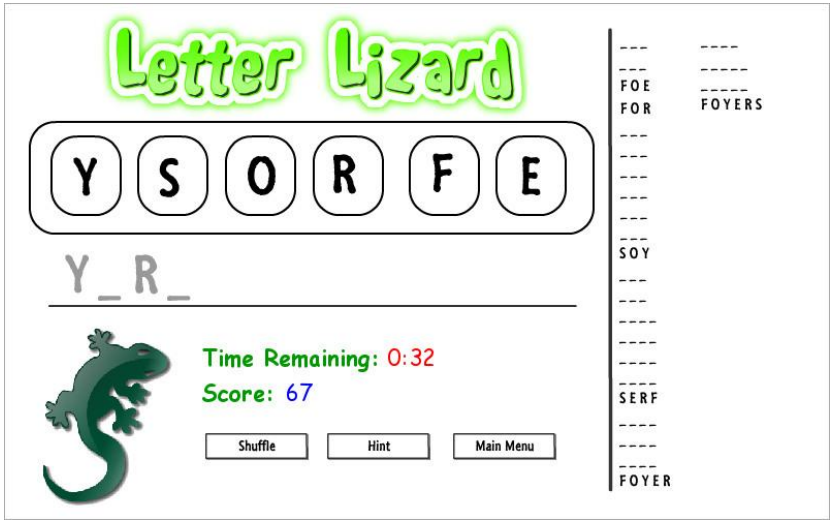
Recap: Letter Lizard Gameplay

- An interactive *word game*
- A sequence of scrambled letters is shown to the user and they must construct as many words as possible from the letters
- The number and length of the words left to be found are indicated by placeholders



Game Features

- Timer
  - A countdown timer shows the time remaining in each round
- Scoring
  - The user is given a score based on points for each letter in words found (vowels are worth less)
- Hints
  - The user can request a hint which shows some letters in place for a word yet to be found



Configuration

- A number of parameters control the degree of difficulty:
  - Number of Rounds
  - Time per round
  - Difficulty of words to find (based on usage frequency)

Project Status

- Completed
  - Game generator written in Python
- Under Development
  - Ruby version
  - JavaScript version
  - Python version
    - Live Demo

Game Generator

- Python module that generates a set of scrambled letters and a list of words that can be made from those letters
- Uses Spell-Checker Oriented Word Lists (SCOWL)
  - A collection of word lists organized by popularity and other factors
- Can be run as a standalone script or imported as a Python module

Game Generator Parameters

```
$ ./game_generator.py -h
usage: game_generator.py [-h] [-s SIZE] [-n NUM] [{easy,medium,hard,
insane}]

positional arguments:
  {easy,medium,hard,insane}
                        level of difficulty

optional arguments:
  -h, --help            show this help message and exit
  -s SIZE, --size SIZE  the number of scrambled letters in the game
  -n NUM, --num NUM     the number of games to generate
```

Game Generator Algorithm

```
def generate_game(difficulty, size=6, num=1):
    words = []
    for line in fileinput.input(files=dicts[difficulty],openhook=fileinput.hook_encoded("iso-8859-1")):
        word = line.strip().upper()
        if len(word) > 2:
            words.append(word)
    while True:
        scramble = generate_scramble(size)
        letter_count = make_letter_count(scramble)
        soln = []
        for word in words:
            if can_make_word(letter_count, word):
                soln.append(word)
        if len(soln) > 3:
            return (scramble, soln)
```

RubyLetterLizard

- Primary Developer: Afiya Nusrat
- Rubygame vs Gosu
- Under development using Gosu

LetterLizardJS

- Primary Developer: Alexander Pokluda
- Status:
  - Currently working on an implementation using CreateJS

PyLetterLizard

- Primary Developer: Michael Wexler
- Status:
  - Basic game features implemented
    - Main menu
    - Uses games generated by `game_generator`
    - Core game features with timer
      - Scrambled letters are displayed
      - Words to be found are shown with placeholders
      - User can type words they find
      - Correctly guessed words are shown in list

PyLetterLizard Implementation

- Function `main()` processes user input and redraws screen
- Class `Game` manages scrambled letters and guessed words

```
class Game:
    def __init__(self):
        ...
    def __guess_word(self, word):
        ...
    def __partition_words_by_length(self, words):
        ...
    def __find_length_counts(self, words):
        ...
    def guess(self):
        ...
    def process_backspace(self):
        ...
    def shuffle(self):
        ...
    def process_letter(self, letter):
        ...
    def draw(self, screen):
        ...
```

Demo