Samurai Duel

David Yao
Alex Cole

Project Objectives

- Create a game that accurately measures a person's reaction time
- Store each score in a text file
- Read and display text file contents on pygame screen
- Have fun

What is Samurai Duel?

- Defeat five enemy samurai by drawing your sword faster than the enemy
- Game counts time between exclamation mark and the player drawing his sword
- Finds players reaction time by averaging out each time
    - Lower time is better

Challenges We Faced

- Finding the best way to measure time
  - Allowing timedelta comparisons, as well as converting timedelta to floats for operations
- Learning to use pygame
- Getting buttons to work properly
  - Initially would not display at all, then would not highlight properly
- Updating scoreboard correctly without redundancies and blank spaces
  - Initially, first slot on scoreboard would be taken by a blank space
  - Would also keep re-adding entire score list each time user accessed it
- Figuring out best way to get username
  - In the interest of time, decided to simply use command prompt input

Code Overview

Modules Used

- pygame
  - Used for almost every aspect of the program
  - Displays window, loads and displays images, allows for simple animation in conjunction with time module
- datetime
  - Used for recording player reaction time
- time
  - Used to incrementally change image to give illusion of movement
  - Regulates screen duration
- random
  - Used to randomize reaction time prompt

Buttons

- Non-gameplay elements rely heavily on buttons to call functions
  - Buttons created through a “Buttons” class
    - Lets them be highlighted when moused-over
    - Lets them be displayed on screen through pygame.draw.rect() function
    - Lets them display text
  - Each screen gets mouse position, gives to Buttons for highlighting and clicking
- Scoreboard, main menu, and instructions screen call Button class
  - Allows player to fluidly switch between screens

Main Menu Screen

- Displayed through screenInitialize() function
- Prior to initializing the game screen, module takes player name as input
  - Used to distinguish player score from others' scores
- screenInitialize() is called by main()
  - Essentially the main menu screen
  - screenInitialize() calls pygame.init(), uses it to display image and window
  - Uses while-loop to display screen until player exits game or changes screens
  - Contains three buttons, which initialize the scoreboard, gameplay, and instruction screen functions respectively

ScoreReader()

- Called each time the player enters the scoreboard menu
  - Opens text doc, “scoreDoc.txt”, reads lines, splits each line into string, writes string to “listoscores”
  - It then iterates through listoscores, checking if it has stuff in it and if it is not in SCORELIST
  - Writes them to global variable, SCORELIST
    - This prevents the first slot on scoreboard from being empty
    - Also prevents repeat entries from displaying each time scoreboard is opened

Scoreboard

- As name suggests, displays screen with player scores
- Called through button on menu screen, “Scoreboard”
  - Button calls scoreScreen() function, which in turn calls scoreReader()
  - scoreScreen() iterates through global variable SCORELIST
    - Displays each score string on screen via pygame
  - Has button to return to menu screen

How reflexes are calculated

- All gameplay contained in reflexGame()
- Plays animation, then waits a random amount of time between 3 and 8 sec
  - After wait, starts datetime.datetime.now() as initial time
  - Calls datetime function again to record end of player reaction
  - Displays difference, with decimals cut-off, on screen as reaction time
- Contains counters for level and total reaction time, averages time/level at end of gameplay
- At end of gameplay, win or lose, displays average time
  - Writes stats (username, average time, level beaten) to text doc
  - Sends player to scoreboard, where their score is displayed

Demo