Assignment 1 Design

By Derfel Terciano

Background:

- The objective of this first project is to create a simple game that involves an *asymmetrical* die or an unfair die. This die is called the **pig**.
- There are 5 different sides to this pig: **Side, Razorback, Trotter, Snouter, Jowler**.
- Since this is an unfair die, there is a higher probability that one side would land more often than the other.
    - Landing a **Side** has a probability of occurring **2/7** times.
    - Landing a **Razorback** has a probability of occurring **1/7** times.
    - Landing a **Trotter** has a probability of occurring **1/7** times.
    - Landing a **Snouter** has a probability of occurring **1/7** times.
    - Landing a **Jowler** has probability of occurring **2/7** times.
- Now let's talk about the game aspect.
    - There are *k* amount players in the game (technically it's k-1 since we are starting our indexing at 0).
    - Since the game takes turns in a cyclic fashion, we return to player 0 after player k-1 has landed a side.
    - The point system:
        - Landing **Side** gives the player **0** points and ends the player's turn.
        - **Razorback** or **Trotter** results in **10** points for the player.
        - **Snouter** results in **15** points for the player
        - **Jowler** gives the player **5** points.
    - Dice Rolling Info (VERY IMPORTANT):
        - Player keeps rolling the die until they land a **Side** OR **someone wins**.
            - If a **Side** is rolled go to the next player.

- If we find a player that has 100+ points, then they win and the game ends.

Deliverables/Files in assignment 1

- pig.c
    - This file is the source file that holds the main() function. In addition, it will also hold any game mechanics and scoring mechanisms that are implemented.
- names.h
    - This is a header file that contains a constant array of 10 string elements. Each element is the name of each player in the game. The array is based on 0 indexing so that means that player 0 is always Wilbur.
- Makefile
    - This file formats and compiles the program into the "pig executable." The formatting aspect of the Makefile uses the clang-format command.
    - Here are some additional Makefile commands as well:
        - make clean: removes and executable and compiler files
        - make format: formats all source code
- DESIGN.pdf
    - This file thoroughly describes the how the program works

PSEUDOCODE/STRUCTURE OF PROGRAM

- General approach:
    - Ask user for the number of players playing
    - Ask user for a valid seed number
    - Start on the first player, record each of their rolls and scores, move onto the next player when they land a **side**.
    - End the game when we find the first person who scored 100+ points.
- **Main Game Loop Pseudocode**

- Ask for valid # of players
- Ask for valid seed #
- While everyone's scores are 0
    - Start on player 0 and **RANDOMLY** roll the pig
        - Check to see where the pig landed
            - If the pig lands upright OR back, then add 10 points to their score
            - If the pig lands on its snout, then add 15 points to their score
            - If the pig lands on its ears, then add 5 points to their score
            - If the pig lands on its side, then end the current players turn and move on to the next player
                - *NOTE:* keep randomly rolling the pig until the player wins OR lands on the side.
    - Increase the player number, so that we are now onto player 1
        - Repeat the steps that were done with player 0
    - ...
    - ...
    - Increase the player to player k-1 (k is the number of players the user inputted)
        - Repeat the steps that were done with player 0
    - Go back to player 0
        - Repeat process until we find someone with a score of 100+
    - If we find someone with a score of 100+, then print a message and make sure that the main game loop either ends or breaks.
        - (since there would be a function that continuously checks if someone has a score of 100+, I don't think I would need a break statement)

- **<u>Main Game Loop Pseudocode Notes</u>**

    - Most of the pseudocode provided above will be separated into separate functions that deal with the different mechanics for the game. For instance,
        - A game_master will be a function that deals with the player rotation and calling the points_updater function for each player
        - A points_updater will handle the random pig rolling and updating a player's specific scores based on what they rolled.
        - A score_checker will check people's scores when called.
- **Error Handling**
    - **Invalid player number input:**
        - Users must input between the integers of [2,10]. If the input is not between 2 or 10, then use fprintf(stderr, "") to display an error message and set the player count to 2 by default.
    - **Invalid seed number**:
        - Based on the manual for srandom(), the input for srandom() must be an unsigned integer. To handle this, the limits.h library gives us the max number for unsigned integers. Therefore, if the seed input is less than 0 OR greater than the max unsigned integer, then an error message is displayed. The message is printed with the fprintf(stderr) command as well. Lastly, if the seed is invalid, then the default seed would be set to 2021.

Credits

- Brian (TA) and Miles on discord suggested to the limits.h library to deal with the invalid seed error handling.
- Auds_o on the class discord helped put up an example doc of what is expected in the DESIGN.pdf
- Miles on discord guided me as to how to deal with invalid characters being inputted. He suggested checking the return value of scanf().

Below is an image of what the game looks like visually