Design Document

- The game that we are going to design is hangman

History of Code Developing

- We used a top down design to try and break down the problem.
- The goal was to write the game first then implement a GUI next.
- We coded all the functions and all the variables that each function required.
- After writing each function we tested each functionality to see if all of our functions worked.
- Thus we ended up debugging the code one function at a time.
- The biggest thing that changed was us not incorporating the GUI
- We also ended up adding the theme idea while coding the game. It wasn't originally in our plan.
- After completing all the functions and getting a functional game we cleaned up the outputs to have a clean and simple visual for the game.
- After this we added the Gallows graphics of the actual hangman.
- Then we went back and changed the inputs to where they don't allow any inputs except the correct ones.

Top Down Design

Create Game

1. Create Graphics:
    a. Create different themed backgrounds
        i. Add each stage of each background to a list;

1. Make each image for each theme
    a. Save images into list

ii. Create Output Window for letters left, and guesses made
1. Use tkinter to create output window for game

iii.

2. Create main game code:
    a. Game startup
        i. Take input open game menu
        ii. Take input to choose amount of players
        iii. Take input to choose the game theme
        iv. Take input to begin game
        v. Access correct list of words
            1. Choose random word
                a. Take as input the key of the theme dictionary, and output a random word from that list
                b. Use random number generator to generate a number for an index of the list of words
                c. Use that index to select the special word
                    i. Strip the word of outside whitespace
                    ii. Split the word into a list
                    iii. Split each word into a list of letters
                d. Strip and split the word/phrase
                e. Create nested list with an underscore for each letter and a space to separate each word
                    i. print/display an underscore for each letter in the word with a space to separate each word
            2. User(s) guess a letter:
                a. If letter in list of letters:
                    i. Remove letter from list
                    ii. If letter is in magic word
                        1. Replace each index of occurrence of the letter into the display list
                    iii. If not
                        1. Move image to next image in the hangman progression
                        2. Subtract one from the counter
3. Create AI code:
    a. List of different themed words and phrases:
        i. Christmas

ii. Pirates
iii. World Cup

- Important Variables
    - theme_files - dictionary containing the names of the Christmas Words and the Sports words file
    - available_letters - nested list with lists of the valid characters left for each player to guess.
    - AI_letter - holds a list of letters for the AI player to display so the user does not see the AI's guesses
    - turn_counter - an increasing number with each turn that, when modulus divided by 2, yields either a one or a zero, which will determine which player's turn it is
    - image_indexes - list of two numbers which indicate what hangman image should be displayed for each player
    - lives - list of how many lives each player has left
    - masked_phrase - list containing two strings - each is what the player sees when trying to guess a letter -> _ _ _ _ _ _ _ _
    - Counter of mistakes left
    - List of themed words
- All the Modules, Functions, and Variables
    - sautomated_guess(x=None): - (Function)
    - opening_game(x=None): - (Function)
    - game_menu(x=None): - (Function)
    - game_theme(x=None): - (Function)
    - begin_game(x=None): - (Function)
    - random_word(theme index: str): - (Function)
    - nested_list(phrase): - (Function)
    - underscore(phrase): - (Function)
    - phrase_string(phrase): - (Function)
    - one_turn(player: int, special phrase, display phase): - (Function)

  - AI_turn(player: int, special phrase, display phase): - (Function)
  - “random” - (Module we used to take random indexes of lists and numbers)
  - alphabet - (Variable)
  - lives - (Variable)
  - image_indexes - (Variable)
  - already_guessed - (Variable)
  - invalid_character - (Variable)
  - turn - (Variable)
  - guess_prompt - (Variable)
  - printletters - (Variable)
  - theme_index - (Variable)
  - secret_phrase - (Variable)
  - save_phrase - (Variable)
  - masked_phrase - (Variable)
  - masked_phrase_display - (Variable)
  - players - (Variable)
  - theme_files - (Variable)
  - letter_distribution - (Variable)
  - letter_guest_list - (Variable)
  - title - (Variable)
  - enter_to_cont - (Variable)
  - menu_text - (Variable)
  - num_players - (Variable)
  - theme_text - (Variable)
- Roles and Responsibilities
  - Tate – Put together the whole code for the project.
  - Matthew – Create the background themes for hangman and all the pics that go with it as well as create and do all the documents.
  - Gianni - Wasn’t in class for us to assign him anything.
- Timeline (Dates that things should be done)
  - 12/04 - Code and Background Themes
  - 12/05 - All the Documents
  - 12/07 - Project and Group Presentation

Statement of Work

- Accomplished from Original Plan
    - The functionality of the code was the same from the beginning to the end.
    - We were able to create an AI that guessed letters in an intelligent fashion.
    - Was able to take user input and control whether it allowed a user to input something or not.
- Not accomplished from Original Plan
    - From the beginning we wanted to create a graphical user interface, but we could not get that to work so we created our graphic in the console.
    - Also, with the graphical user interface the screen is not split with each player's game.
    - There are not diverse backgrounds or hangman for the two different theme options we have, Christmas and Sports themes.
- External Modules used
    - We did not use any external modules outside of the standard python library. The only module we used was the random module. We used the random module to

select random indexes in a list, which was used for selecting the secret word and creating the AI's guessing mechanism.

Code of Cooperation

- Rule 1: Every member must participate to a full and equal extent
- Rule 2: Each team member must respect one another's time
- Rule 3: Come to class every day with a better understanding of the material than the prior meeting
- Rule 4: Be willing to help one another
- Rule 5: Be open-minded to new ideas and admit fault
- Rule 6: Each team member must communicate: do not ghost the group!
- Rule 7: If a team member cannot make a group meeting, they must contribute in their own time to the project
- Rule 8: Each team member must HAVE FUN!

Time/Effort Task Each Team Member

Gianni:

- Gianni didn't participate in the project. Was out of class a lot and failed to take initiative to tell the professor and failed to see where he could contribute while he was away.

Matthew:

Technical Documentation	1 hour
Statement of Work	4 hours
User Manual Rules of Game	10 minutes
Background Themes	4 hours
Sports Words Text File	½ Hour

Tate:

Game Code - All functions in code	15 hours
User manual rules of Game	½ Hour
Game Code - All loops and making the game work	5 Hours
Top down Design	3 hours
Test Cases	1 Hour
AI Letters Text File	⅓ Hour
Christmas Words Text File	1 Hour

Summary of Teams Work

You can always overestimate what you can do in a day and what you can do in a short amount of time and this project was no different. We wanted to create a graphical user interface that had four windows. Two of which displayed the hangman Gallows and figures for the players and the other two displayed the available letters and took input from each user respectably. Although trying to learn the module Tkinter and create graphics that accomplished this goal, the code that

we already had written inhibited us from fully integrating graphics into the game. We ended up creating a game on the console that displayed graphics in a much less optimized way, however, we determined that it would be better to have bad visual output than to have no output.

We assumed that we would be able to build a functioning game in the console first and then easily add graphics to it after it was complete, which was not the case. Because of the way the graphic modules are integrated with regular code, the input and the output of the already coded console game did not translate easily. Specifically, the problem that posed the most challenge was taking input from the graphics module and then updating it on each turn as the code looped. Another unforeseen challenge that we had was the difficulty of trying to split a window four ways and have unique outputs in each of the four frames.

Contributions by each Team Member

Tate – He helped tremendously by coding the whole game. From using top-down design to incorporating the different coding methods we used in class like functions, loops, and file openers.

Matthew – Created the background designs and printed each photo for the pieces for the hangman that would be displayed as the game went on. However due to struggles of getting the Graphical User Interface to work we did not incorporate them. Every single document was created by him from the rules of the game, what coding methods we used, to typing this very code he typed and wrote out thoroughly the documents.

Gianni – Did not contribute to the team and would leave class early and had a couple of times where he did not show up to class without leaving a valid excuse.

Reflection Statement

From the beginning we could have kept our teammates more accountable as we had one teammate that did not contribute at all. Our communication was decent over text, however, but we should have spent more time in person putting a game plan together and working together. That would have made our ability to complete this project with more ease and to our level of satisfaction.

We would have approached this differently by starting the project earlier as we had to crunch this project in limited time causing high stress levels and the level of success falling in not just this class but others. We would have not coded the game first and then put it in the GUI, but instead we would have done it in tandem. The way we did it made it extremely difficult to translate from console to graphics which led to us abandoning the idea all together.