Daniel Schiefer

Mr. Charles Thompson

CPT-185-A80H-2024FA

5 Dec. 2024

Final Project Written Proposal

1. Project Title and Overview
   1. Title: Fantasy Fighters: Mystic Dice Battle
   2. Overview: To create a dice rolling battle game that tells a short story while providing some challenge and being enjoyable to play. I will implement some randomness to the battles which will make every play through different to increase replay value. The target audience would be casual gamers.
2. Functionality and Features
   1. The game will be developed using a c# form, opening on a main menu. The main menu will include buttons to view instructions and high scores. There will also be a start new game and exit button.
   2. The user will have three heroes in which to battle with, an elf, a dwarf, and a wizard. They will battle across a total of nine stages. The first eight stages will be broken into four different areas with a secret chest at the end of each area. Each of the first eight stages will consist of three different enemies, for a total of twenty-four different enemies plus a boss battle at stage nine. As each battle will be a three vs three battle, each enemy will be strong and weak against one different hero, which will add a rock, paper, scissors element on top of rolling the dice. The user beats the game and saves the world if he can defeat all enemies. The game ends and the world falls into chaos if all three heroes fall in battle.
   3. To battle, the user must select which of their heroes they wish to attack with, and which enemy they wish to attack. Then they will click the button to roll the dice, two dice are rolled for each the hero and the enemy. Damage will be deducted from each the hero and the enemy based on the rolls of the dice and any buffs that go along with it. The computer will then take a turn, selecting an enemy to attack the user with and a hero to be attached. This will continue until either all heroes or all enemies are dead for that stage.
3. Design and Architecture
   1. Forms
      1. Main Menu – Will display at start of the game with links to start a new game, go to the instructions screen, the high scores screen, and exit the program.
      2. Instructions – Will display an overview of the instructions and how to play the game. Will also have links to start a new game, return to the main menu, go to the high scores screen, and exit the program.
      3. High Scores – Will display a list of the high scores achieved for the game. Will also have links to start a new game, return to the main menu, go to the instructions screen, and exit the program.
      4. Username Input – Will be displayed after the user selects to start a new game. Soley used for recording the high score.
      5. Story Board – There will be a brief story board before and after each stage, linking each stage together.
      6. Battle – The main battle form used for playing the game. Will display the background, the heroes along with their stats and a radio button for choosing them, the enemies along with their stats and a radio button for choosing them, as well as a secret on some stages. There will be a roll the dice button as well as a quit to main menu button.
      7. Bonus Stage – A bonus stage will be prompted after each area (after every two stages) that shows a treasure chest. If the user finds the way to unlock the treasure chest, they will be given the reward. If not, the chest remains locked.
   2. Classes
      1. Heroes – Create a hero class to define the three hero objects, includes name, image, health, and boost.
      2. Enemies – Create an enemy class to define the twenty-four enemy objects, includes name, image, health, strength, and weakness.
      3. Game Data – Create a global game data class to store the following information:
         1. Username – used to recognize high scores.
         2. Stage – used to load the appropriate story board and battle stage information.
         3. Enemies Defeated – used to calculate high score.
         4. Hero List – used to hero list does not get recreated with each stage and stores their health and boosts from stage to stage.
         5. Random Number Generator – used to randomize multiple aspects of the game.
         6. Keys – used to unlock secret treasure chests.
         7. Dragon Amulet – if found used to lower final bosses hit points.
      4. Exit – able to call from multiple forms inside the game to exit the program.
   3. Random Number Generator – used to randomize many aspects of the game to make each play through different increasing replay value.
      1. Dice Roll – will randomize the roll of the dice.
      2. Computer Turn – will randomly pick an enemy to attach a random hero.
      3. Secrets – will be random so they are not always in the same place.
      4. Enemy Health – each stage will consist of three enemies with a combined forty health points, which will be randomly dispersed between the enemies.
   4. Stage Builder – will use a switch and case based off the Game Data int stage, so it knows which background, enemies, and secrets if applicable to load.
   5. Adding Units to the Battle Form
      1. Heroes are only generated once, and their data is passed from stage to stage.
      2. Enemies are generated for each stage with their total health points totaling forty to start each stage.
      3. The final boss is a single unit, must hide other unit markers and assign hit points. Initial hit points for final boss is seventy-five.
   6. Hit Point Generator
      1. Randomly picks a number from four (10% of stage total) – twenty (50% of stage total) for each of the three enemy units per stage.
      2. Finds the scaling factor so that the total enemy hit points should be equal to forty and scales each random number.
      3. Adjusts the min and max for each enemy back to four and twenty in case anybody was scaled out of that range.
      4. Find the remaining health needed to get the stage total health to forty and add one health point to each unit until forty is hit.
   7. Loading Health – must be loaded to the form at the beginning of the stage as well as after each time damage is deducted.
   8. Get Dice Image – will use switch case based on the random number generated to retrieve the appropriate image.
   9. Adjusting the Attack – the base attack of a unit cannot exceed that of its remaining hit points.
   10. Attack Bonus
       1. If an enemy is strong against a hero, the hero will lose one from its total dice roll and the enemy will gain one to its total dice roll.
       2. If an enemy is weak against a hero, the hero will gain one to its total dice roll and the enemy will lose one from its total dice roll.
       3. The final boss is doubly strong against all heroes, all heroes will lose two from their dice roll and the final boss will gain two to its dice roll.
   11. Minimum Attack – each attack will count at damage at least one regardless of the buffs and debuffs.
   12. Deducting Damage – The total roll of the hero will be deducted from the enemy and the total roll of the enemy will be deducted from the hero regardless of who initiated the attack.
   13. Checking for Dead
       1. After each time that damage is deducted, must check to make sure all units are still alive and disable radio button for any killed units.
       2. If all heroes health has reached zero, the game is over, and the user loses.
       3. If all enemies health has reached zero, that stage is complete, and the user moves on to the next stage.
   14. Resources (Images) – I will use a combination of AI generated art and google searches to find all the images I need for the game.
4. Methodology
   1. I will design the user interface first, adding elements to the game as I go along. The battle stage will be last, and once the stage and unit population is working correctly, I will start the battle mechanics.
   2. I will test my code as I progress. I prefer an agile development process so I can make sure each task is working before moving on. This will make finding bugs easier if I handle them as they come up.
   3. One of the more challenging aspects was getting the enemy random health generator to work as each stage had to equal a total of forty hit points, no enemy had less than 4 hit points, and no enemy had more than twenty hit points. Another challenging part was getting the heroes data to pass from stage to stage while having the enemy data start new at the beginning of each stage. However, I was able to address this by having the enemy data generate each time the battle stage was loaded while saving the hero data to my Game Data Class.
5. Expected Outcome
   1. The completed project will be a dice rolling game that uses randomization for multiple aspects to make each play through different and challenging. The user will start by taking their turn followed by the computer taking their turn until either all heroes or all enemies are defeated for a given stage. The user will win if they can beat all twenty-four enemies plus the final boss, the user will lose if all three heroes fall in battle.
   2. Success is a functioning game that meets the above criteria. I will know success by playing the game and testing the code.
6. Timeline – I suspect the total time taken to complete the game will be one week.
   1. Research and Planning – one day.
   2. Form and Stage Building – one day.
   3. Battle Mechanics and Debugging – two days.
   4. Final Implementation – one day.
   5. Testing – two days.
7. Optional Goals
   1. Unit Buffs – Time permitted, I will add unit buffs to player if certain criteria is met.
   2. Secrets – shhh, they are secret. Yes they are in there!
   3. High Scores – To calculate, display, and store high score information.