Brandon Aubrey

Morris Lagrand

Tyler Sammons

Final Project Report : Pokemon

Brief description of program:

We split our program into four main parts; Pokemon, move, ash, and moveTest. The first part Pokemon contains a class to manage the Pokemon. It is just a class that contains the basic structure for how each Pokemon is constructed and interacts with each other. Instead of using inheritance and having each Pokemon have its own class inherited from the Pokemon class, we put each Pokemon’s important attributes in a file called PokeList.txt and had a constructor that reads the information from a specified line given the Pokemon’s number. This made implementing the 150 pokemon much more time and memory efficient. Furthermore, the Pokemon class relies heavily on composition. It has a set (array) of four moves it uses to interact with the opposing Pokemon. The moves are loaded in a similar fashion, from a moveList.txt file. The core of the game play is coded in the useMove function and its numerous helper functions. It works by taking in an index and another pokemon then calling a number of other Pokemon functions to appropriately act.

The move file, like the Pokemon file, streams in a move from a text file named moveList.txt. By doing this, we were able to save time, memory, and also include all 163 Gen I moves. The constructor reads in the number, name, type, power, accuracy, powerpoints, status change, and several other attributes associated with the moves and stores them into variables. The move file handles all actions pertained only to the moves such as reducing the power points after a moves is used, displaying the move name and stats, and resetting the power points after a battle. An array of 4 moves is created inside of the Pokemon file so that the pokemon can load 4 moves and use them accordingly.

The ash file includes a class that represents ash ketchum, including his xoffSets and yoffSets. the function that we use in our main file, moveTest, is called go(). This function takes in and returns an int. When this function is called, the Veridian Forest background is applied to the screen, and then Ash is applied to the screen at a location dependent upon the integer taken in. Ash then moves around the screen in response to the arrow keys. There are boundaries throughout the map that won’t let ash enter, and there are four battle areas. When ash enters a battle area, the go() function returns an integer to the moveTest which determines his next spawning position, and the battleScreen opens.

The moveTest file contains the driving code behind the game. It includes the classes we have described above, and many SDL functions. The main function begins by applying a choice background to the screen. The user chooses his Pokemon by clicking one of the 150 choices on the screen. Next the program enters a while loop that runs while the player’s pokemon has not fainted. Next, the opponents pokemon is randomly chosen, and the go() function is called. The Veridian forest map opens, and ash moves around until he enters a battleArea, at which point the battleScreen is applied,along with the user’s and the opponents Pokemon sprites and statistics. A while loop runs while both the user’s and the opponent’s pokemon have not fainted, and the battle takes place. Once someone faints, the next opponents pokemon is chosen, and the Veridian forest map opens again. This happens over and over until the user’s pokemon faints.

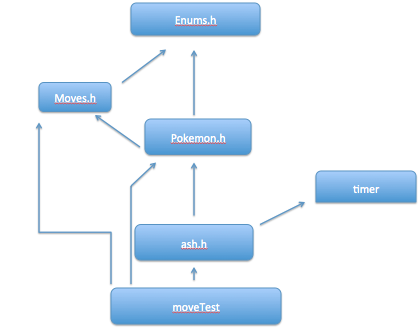
How to play:

The game begins by displaying a sheet of the 150 pokemon options for the user to pick. To choose, the user simply clicks on the Pokemon they wish to use on their adventure. After selecting the Pokemon, they are put into a forest to wander around until they are attacked by a wild Pokemon. This opens a battle screen which is a turn based battle. The user is shown a lists of their Pokemon’s moves on his turn. He must simply press the number that corresponds with the move they would like to use and their Pokemon uses that move. The user can also hit the number 4 to exit the whole program when it is their turn. On the wild Pokemon’s turn, nothing is displayed and the Pokemon will simply wait for the user to press any button to use a move. This process continues until either Pokemon faints or the user presses 4 on their turn. If the wild Pokemon faints, the user gains some experience and is put back into the world. If the user’s Pokemon faints however, the game ends and the user loses. The user can battle endlessly as long as his Pokemon keeps from fainting.

Bugs:

* Boundaries in the world are not aligned with the objects
* In battle scene, if the user hits multiple buttons before a turn ends, the game will play out as if those buttons were hit in successive turns, pushing the battle scene forward without the user’s control
* In Pokemon selection if user clicks on a space that is not a Pokemon it crashes with a friendly error message
* Cannot exit game from forest scene
* Recoils that reduce HP to 0 do not end battle

UML Diagram



Log of Hours Worked

|  |  |  |
| --- | --- | --- |
| **Brandon Aubrey** | **Tyler Sammons** | **Morris Lagrand** |
| 3/19/2015  3 - SDL tutorial 1-5 | 3/21  4-SDL tutorial 1-12 | 3/20/2015  2 - SDL tutorial 1-3 |
| 3/23/2015  3 - SDL Tutorial/Basic Animation made | 3/22  2-SDL tutorial 13-22   * compile something | 3/21/2015  2 - SDL tutorial 4-5 |
| 4/3/2015  5 - Made Basic Pokemon Class and tester | 4/3/15  1.5 - familiarizing self with Pokemon game and graphics | 3/23/2015  3 - Animation and Game play code |
| 4/4/2015  5 - Made Pokemon Database and added types to Pokemon Class | 4/17/15  4 hours, background and movement graphics, and experimenting more with SDL | 6 hours,  Game play code |
| 4/10/2015  5 - Added a constructor to Pokemon class that loads from file PokeList and fixed bugs with typing | 4/18/15  3 hours, making an “ash” class move around veridian forest map | 4 hours,  Game play code |
| 4/21/2015  3 Group meeting | 4/21/2015  3-Group meeting  Map out rest of project and fix bugs | 4/21/2015  3 Group meeting |
| 4/24/2015  3 - miscellaneous | 4/25  3 - Work on getting text to show up in right order, experimenting with ttf | 4/24/2015  3 - miscellaneous  adding finishing touches |
| 4/25-4/27  4 - added special case moves | 4/27  7-Create battlescreen, get level and HP to display,  function to cut 150 sprites from 1 int, edit Pokemon.cpp functions to return string, get moves and results to show up and change | 4/27/2015  2 - fixing the give and receive hp function |
| 4/28  4 - save and load | 4/28  6-Register keystrokes to choose move SDL, make disp\_moves() return a vector, get sprite sheet to choose pokemon, register one click to correspond to correct sprite, integrate ash class and veridian forest with main moveTest file. | 4/29/2015  2 - miscellaneous |
| 1.5 Final bug fixes/ changes | 4/29  2-Made choice screen appear first, put battle code in continuous loop, made sure ash returns to map at right location |  |
| Total: 36.5 | Total: 35 | Total: 26 |

Meetings:

Our group has been in contact via group text message, and short meetings after Fund Comp classes. We have split roles in the project. Tyler will be focusing on graphics/animation, Morris and Brandon will be focusing on movement and gameplay. We have made progress creating Pokemon classes and code for the gameplay.