What is our system?

Our program will be a Pacman game with a UF theme. The player has to try and survive as long as possible on the University of Florida campus in Gainesville while being chased around by enemies, such as people handing out flyers in Turlington and FSU defensive ends. The longer a person stays alive on the game, the more points they get and the more enemies appear and try to tackle them.

How will users interact with our program?

User inputs will include:

- Choosing options in the initial menu system.
  - switching between the choices using the arrow keys
  - viewing instructions for how to play
  - choosing the difficulty
  - starting the game
- Giving movement directions during the game
  - UP
  - LEFT
  - DOWN
  - RIGHT

The program will output:

- The initial menu system GUI, with choices between...
  - viewing instructions
  - switching difficulties
  - starting the game
  - also includes an indicator for which menu choice the user has selected
- A list of instructions, should the user choose to view the instructions
- A GUI during the game's running, which will be the game map showing
  - the player avatar moving around
  - the enemies (FSU footballers, people handing out flyers) chasing the player
  - the time the player survived in game
  - any powerups (such as bottles of Gatorade!)
  - the pathways the player and enemies can use
- The player's final score:
  - perhaps also include a table of high scores including the best scores from previous runs
  - choice to exit the game or try the game again at the same or varying difficulty

One special use case for the program is the coder desiring to debug the game, to analyze whether the enemy movement algorithms work, whether the movement shown on screen is the same as that displayed under the hood, etc. This will involve the debugger/tester to use something other than the UP/LEFT/RIGHT/DOWN arrow keys to control enemy avatars, in order to make sure that the game runs as intended, that the player avatar can defeat enemies when powered up, and that the game terminates when an enemy catches a player and allowing the debugger/tester to use a key to force enemies to move in small increments.

- using a command line argument (e.g. "./[PROGRAM NAME] debug" to run debug mode) when running the program should suffice for this

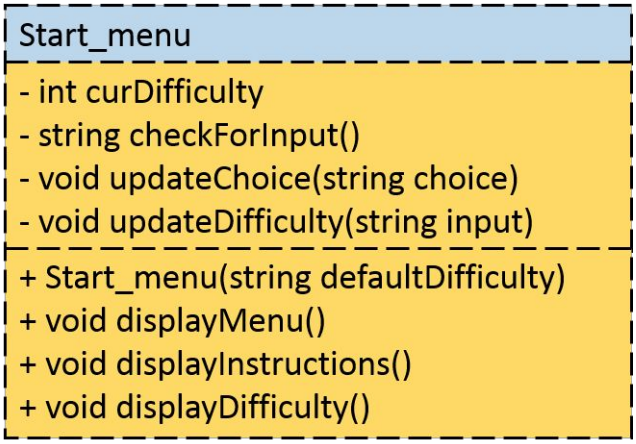
The program's overall flow will be:

- Output Start Menu to player, allow them to choose between options
  - display instructions if wanted
  - allow switching difficulty (will make rate of enemies appearing faster, or make enemies quicker)
  - allow starting game
- Start game, initiate game GUI
  - create window, create Pacman maze, overlay UF campus map section onto maze
  - allow player to move around
  - enemies start appearing and chasing the player, score (survival time) begins counting
  - player/enemy movement is constrained to moving within the maze walls/campus roads and walkways.
  - periodically, extra enemy appears
  - periodically, power-up (Gatorade) appears
  - while enemies move around, ensure they don't collide with player
- End game when an enemy has collided with the player
  - output player score
  - output previous high scores
  - allow player to choose to exit game or start again from the Start Menu

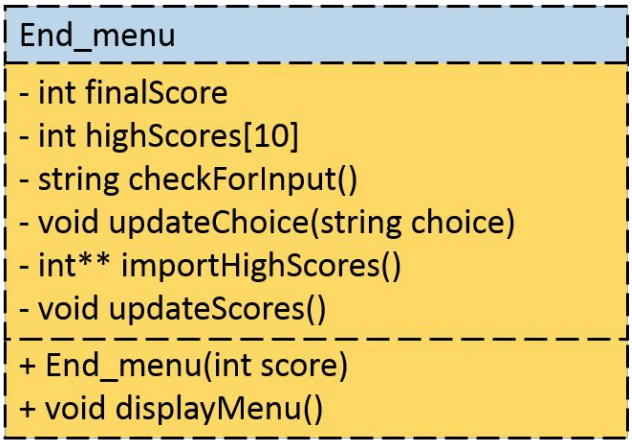
How will the program operate under-the-hood - modules, classes, methods?

The Menus module (menus.h) will contain the Start_menu and End_menu classes: these correspond to the menus the user interacts with before starting the game and after the game has ended.

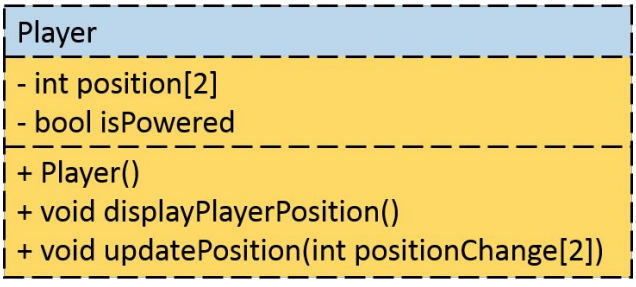
The start menu mainly has to present the choices the user has, allow the user to switch between instructions, the main menu, and the difficulty menu, and allow the user to start the game (only choice is made in the menu - actually starting the game is done in the manager).



The end menu has to display the user's final score, as well as previously achieved high scores - if the user's score is higher than one of the high scores, it has to update the list of scores in the interface and in the score data file. Finally, it has to present the choices of starting again or exiting the game to the user.

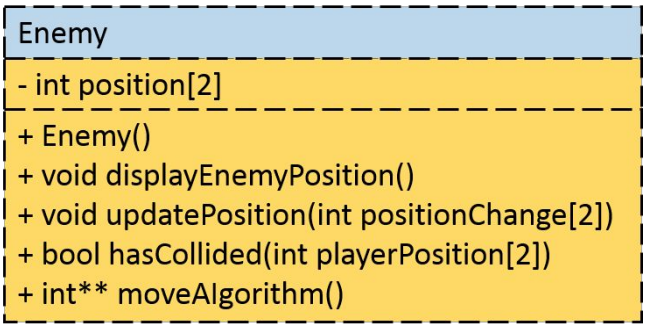


The characters module (characters.h) will contain the Player and Enemy classes - representing the player's avatar and the enemies that chase the player around the game map.



The Player object has to store where on the map the player is located and whether the player has powered up. It also has to provide a position updating function and a position displaying function.

The Enemy class has to store an enemy's position and be able to change it, and has to allow the class storing enemy objects to update positions based on how an object moves according to its specific algorithm. Enemy objects also have to be able to check if the player object has collided with them.



The Game_interface (interface.h) class will be the majority of the code running while the user plays the code. It stores the difficulty chosen in the start menu, the user's score, the size of the interface, locations of power-ups and functions to create them and deal with the player's gaining and losing power-ups, a map of places the player and enemies can and can't move, a container of all the enemies, and the Player object. It also checks for input from the user (arrow keys to move the avatar in different directions), as well as moving the player and enemies automatically, and displaying updated positions of the characters. It can check if the game has ended yet (player has collided with an enemy) and manage the score of the game.

The main program itself will manage switching between the menus and the game interface depending on the states in the program.

What will be our execution plan for this project?

The tasks for this project are to create the two menu systems (Start_menu and End_menu), the character (Player and Enemy) classes, the overarching game interface (which should take up the bulk of the coding/time) and figure out how to switch between the menu systems and the interface while maintaining the data (difficulty, score) required across multiple interfaces.

As such, the plan for this project is:

- **Week 1**
  - Create the menus
    - Start Menu
    - End Menu
  - Figure out which libraries we should use to create our GUI
  - Attempt to create menu GUI and see if inputs/data can be managed
- **Week 2**
  - Create character classes
    - Enemy
    - Player
  - Create movement algorithms (AI) for the Enemy class
- **Week 3**
  - Create a basic game interface to allow testing switching between the menus and the interface
  - Use basic game interface to test how Player and Enemies interact
- **Week 4**
  - Create a more robust game interface
- **Week 5**
  - Final debugging and testing