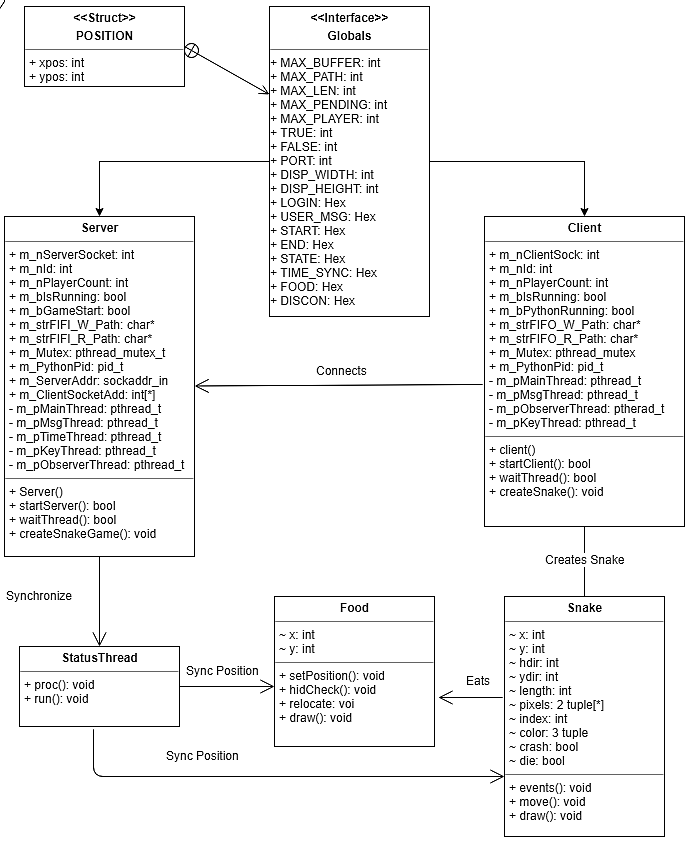
Snake Game

Diagrams:  
1. Class Diagram:



1. Globals: This a header file that simply describes all the static attributes for our game, for example, the maximum number of players, the size of the game window, certain Boolean constants, etc. These values are stored in the back-end to reduce/ avoid using magic constants and updating certain game-wide global variables becomes easier.
2. Position: this is a struct, that is defined in the “globals” header file. This struct is used to define the position of an item, in the game, be it, a snake or food for the snake.
3. Server: This class encapsulates the server for the game. Any player can choose to be a server by using the “-s” option while running the game. The function of this class is to start a TCP server bound to the host’s IP address, listening on the port that is specified in the globals header file. The server then waits for all the players to connect. (specified by the MAX\_PLAYER in globals) Once all the players have connected to the server, the server can initialize the game by sending a “-c start” command and sends the required game data to all the connected clients.
4. Client: Each client represents a (non-server) player in the game. The client is responsible for instantiating a snake object via the python front-end based on the parameters provided to it by the server, for example, the player index. The client can establish itself as a client by the option of “-c [IP\_ADDRESS of Server]”.
5. Snake: The snake class encapsulates an in-game snake, that is controlled by one player. Each snake object is constantly listening for key-events. On the appropriate key event (left, right, up or down) the events function will update the parameters of the snake respectively.
6. Food: The food class represents the food, once consumed, will increase the length of the snake.
7. StatusThread: This class is responsible for thread synchronization amongst all the players, and ensures that all the players have consistent copy of the game.