Software Architecture

Since we were creating a game, for Project 3, we had no need for databases or storage. Therefore the most logical software architecture choice was the Client-Server Architecture, although that does not fit properly either. We used a software architecture loosely based on Client-Server Architecture. I say loosely based, as we did not use servers for our game. It is based on CSA on the basis of us using keys to move the bird and navigate the screen. On pressing a key, a request is sent and the bird moves in response.

The game in itself does not require any persistent storage with respect to Project 3. It could be expanded to support leaderboards and similar features (in the future), which would result in the implementation of databases. The key press gets scheduled on to the pygame event queue and is executed one-by-one. Since our program does not use actual servers, the game engine becomes the server, while the user becomes the client. Upon offering an input, the game engine processes it, and sends out a response, which in our case, is the flying motion of the bird in the game.

The application in itself, has an external dependency (pygame - a python module) that needs to be installed pre-usage. All assets are localized, making the architecture platform independent (works on Windows/Mac/Linux) The main reason for using CSA here, is the absence of persistent data. The program works only based on the logic layer, which acts as a server. All mathematical and graphical calculations are done upon client request.