We used the MVC (model view control) architecture for our game. MVC is most similar to the 3-tier architecture, so I will compare it to that. In our architecture, the presentation tier is the view., he logic tier is the control, and the data tier is the model. We have everything in the data tier stored as objects and the numbers and vectors that those objects hold. The main game loop is the controller, it takes input from the user (or in this case the player) and it updates the objects we have and the data we have in those objects. Before we go about processing more input, we render our frame on the window. This is the view, and the view relies directly on the data. The view is called to render from the controller, but all the specifics about the view are held in the data tier. This model worked out best for us because it's exactly how games work. Games for the most part use the MVC architecture. Every game engine out there (who's source code we can see and edit) uses this architecture. The most popular of the free engines: Unity and Unreal Engine both use this architecture. This is because it's the most suited to the task of a video game. There's no reason to do anything more complex since it will only worsen the performance of the game. Additionally, since the presentation tier needs to be updated every frame, just like the data tier, the 3-tier architecture makes the most sense.