The Canvas Painter game engine is built upon the principles of modularity, reusability, and simplicity, ensuring a flexible and efficient foundation for game development.

Abstractness:

Modularity: Emphasizing a highly modular structure, the game engine segregates different components to handle specific aspects of the game. For instance, the NetworkManager class takes charge of network-related activities, while input manager handles the input related work. This segregation streamlines code management, debugging, and future expansions.

Reusability: The game engine thrives on reusability, enabling various components to be utilized beyond the current game. the NetworkManager and input manager can be easily integrated into other games requiring network communication or handling input related tasks.

Simplicity: the game engine adheres to simplicity in design. The code structure is well-organized and accompanied by helpful comments, making it developer-friendly for easy comprehension and modifications and explanations to what each function does.

My matrix/table for Object/Entity components :



**Networking:**

The Canvas Painter game engine employs the WinSock2 API for network communication, facilitating a simple server to client based network in the game environment. The NetworkManager class handles socket and connection management, orchestrating network-related tasks efficiently.

The NetworkManager establishes a listening socket to accept incoming connections from client . Additionally, it provides a message-sending method, enabling communication among players when actions like sending a hellow message.

Networking is using a dedicated thread that runs the listen for messages fucntion on loop, so that even after sometimes we wanted to send a message it is always listening.

By segregating networking listening task into separate thread, the game engine avoids blocking or slowing down the main game loop, guaranteeing smooth gameplay. This networking approach efficiently manages client/server connections. Moreover, the networking code incorporates robust error handling, recovering from common issues like socket creation or binding failures. As a result, it attempts alternative port binding if the default port is unavailable, ensuring seamless networking functionality.

**Threading:**

I have used a single thread that runs in loop for network related fucntion which is listening for messages on the network