Inputs are WASD to move, Z to Undo, Q to Quit.

Object Pooling is an Design Pattern to be used for optimization. It works by disabling objects in the object pool when they are no longer used, instead of deleting them. Also, instead of creating new objects each time, it just enables them from the pool when they are needed. Object Pooling is implemented by storing a queue of unused objects, methods to fetch objects for use and to return objects to the queue after they become unneeded. Object Pooling helps optimize scenes in which certain objects would otherwise be getting unnecessarily destroyed and initialized often. They would be unnecessary since they would just be getting initialized or destroyed soon enough again. This lessens the strain on the Garbage Collector. The objects are instead disabled and re-enabled, a much cheaper operation.

Command Design Pattern turns actions into data. The main use of the Command Pattern is for implementing Undo/Redo functionality. A Command is implemented as an interface, which has realizations which can perform, undo, and redo the specific action it desires. When an action that would potentially want to be undone would be performed, instead initialize an instance of the applicable realization of the Command interface.
A way the command pattern could be used in a pacman-style game, could be to let the player "go back in time" a bit, for if they get cornered/surrounded by the ghosts. This would have to cause the pellets eaten in that time to be recreated, which is what was required, but it would also have to involve moving the pacman and sprites backwards along whatever paths they took.

I would've implemented an Input Rebinding System. To do so I would've created a new Canvas which would get toggled when the player pauses/unpauses the game. This canvas would have labels for each action, with buttons next to them which display the current binding for that action, with buttons next to those for resetting them to their default binding. When the button displaying the current binding is pressed, I would blank its label, and then would capture any input and bind it to that action, and set the label of the button to the new binding. I would also add a button to the canvas which would reset all the bindings to their defaults.