Pseudocode

Step 1: Decomposition

To recreate the classic arcade game, Pacman.

Pacman is a game that involves a character called Pacman that moves around the game board, through inputs from arrow keys by the player.
Pacmans goal is to eat all the pellets on the game screen. Pacman cannot move through walls or fall off the game board.
The game board is set at the start of the game and does not change. The ghosts will start in a holding pen and move around the screen to chase Pacman. Pacman will start outside of the holding pen.
There are four ghosts that chase Pacman around the game board, with the aim of "catching" Pacman and causing him to lose the game.
The game is won when the player consumes all the pellets on the game board
The game is lost when Pacman loses all his lives from running into the ghosts.

Pacman is controlled by inputs from the player via the arrow keys. When the left key is pressed, Pacman moves to the left, etc. Pacman will stop moving in any given direction when he encounters a wall. Pacman must open and close his mouth during movement to simulate him eating the pallets.

The ghosts are controlled via random inputs where they move around the game board to catch Pacman. The ghosts start in the holding pen in the middle of the game board and move around from that starting point. When the ghosts catch Pacman, the game is lost.

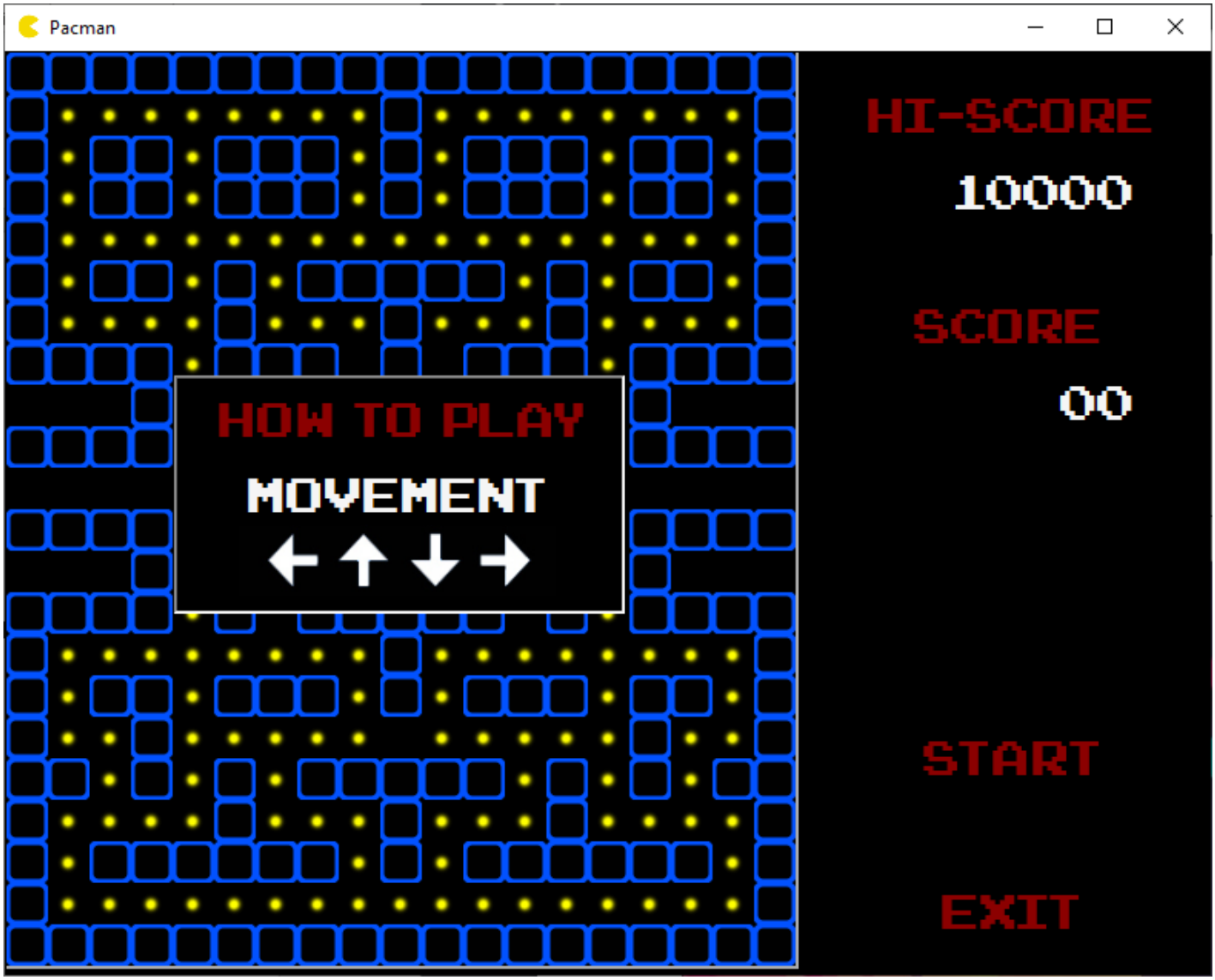
- What are the extra features or functionalities that are not essential to the basic game?

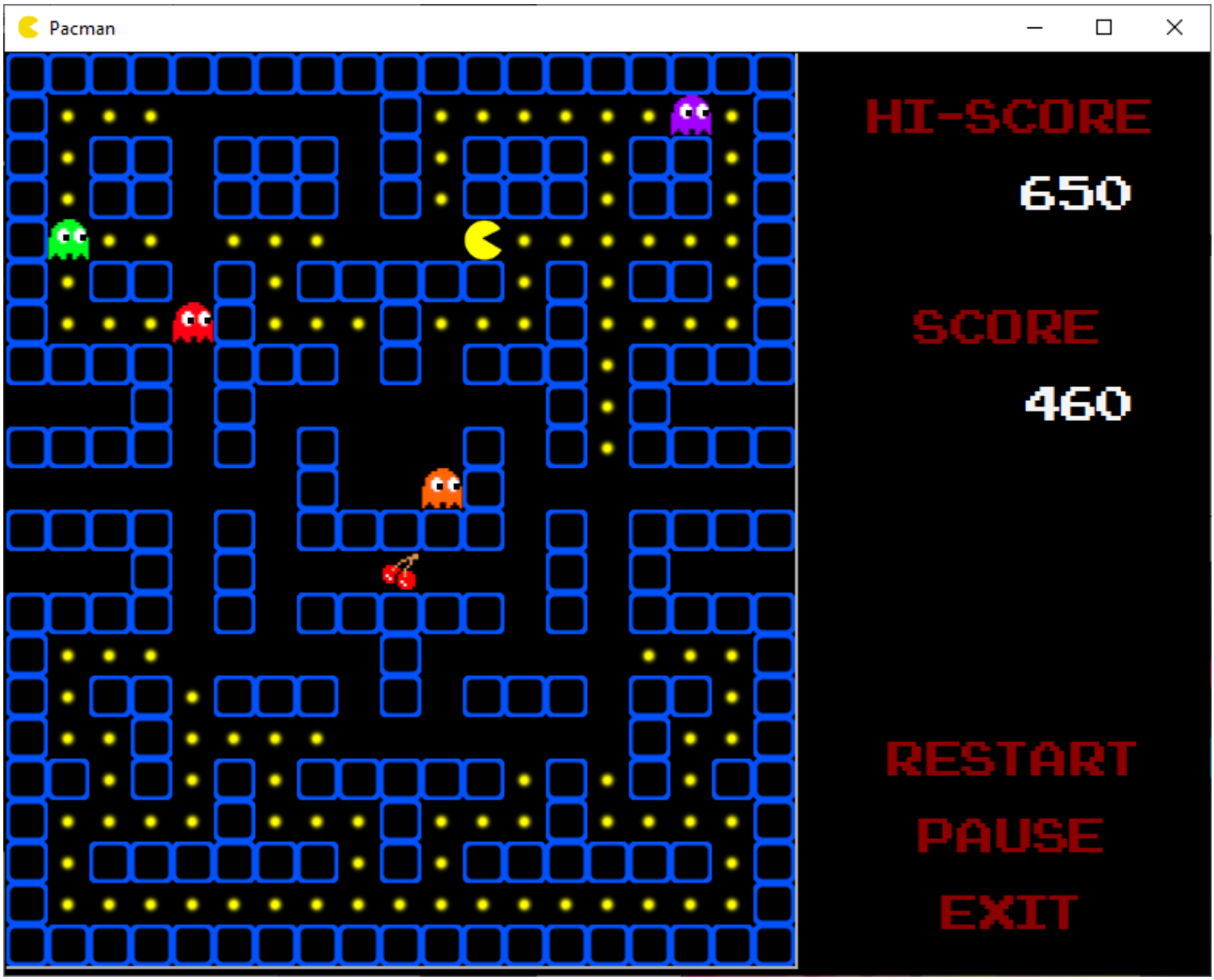
Powerups, lives, levels, high scores, pause and resume, bonus points, ghosts chasing Pacman

- What are the curly issues that will still need to be resolved?

Extras, Collision with the walls, Ghosts passing through Pacman, Ghost AI, Replacing the pallets with blank space.

Step 2: Form Design




Pacman
HI-SCORE
650
SCORE
460
RESTART
PAUSE
EXIT


Pacman
HI-SCORE
650
SCORE
530
YOU LOSE..
PLAY AGAIN ?
RESTART
EXIT

Step 3: Abstraction

Controller:
The Controller class is used to start, end and control the game, and pass through win/loss conditions to the form.

eBoard:
The eBoard class is an enumeration that is used to store the values of the board parameters so that they can be easily changed if required.

eDirection:
The eBoard class is an enumeration that is used to store the directional parameters so that they can be easily referenced by other classes.

ErrorMessage:
The ErrorMessage class is an enumeration that is used to carry error messages that are references by the rest of the program for detecting game states such as wins and losses.

Player:
The Player class is a parent that is used to draw Pacman and the ghosts to the screen, as well as determine the position of the sprites on screen and detect collision with the walls and boundaries of the game field.
Ghost and Pacman share the same code for draw and teleport methods, so they are declared in the Parent class here.
Since Player is a parent, the methods in this class are inherited by the child classes; pacman and ghost.

Pacman:
The Pacman class is a child of the Player class and is used to control movement of Pacman around the game field, keep track of when Pacman eats a pallet or cherry, and animating Pacman during movements in game.

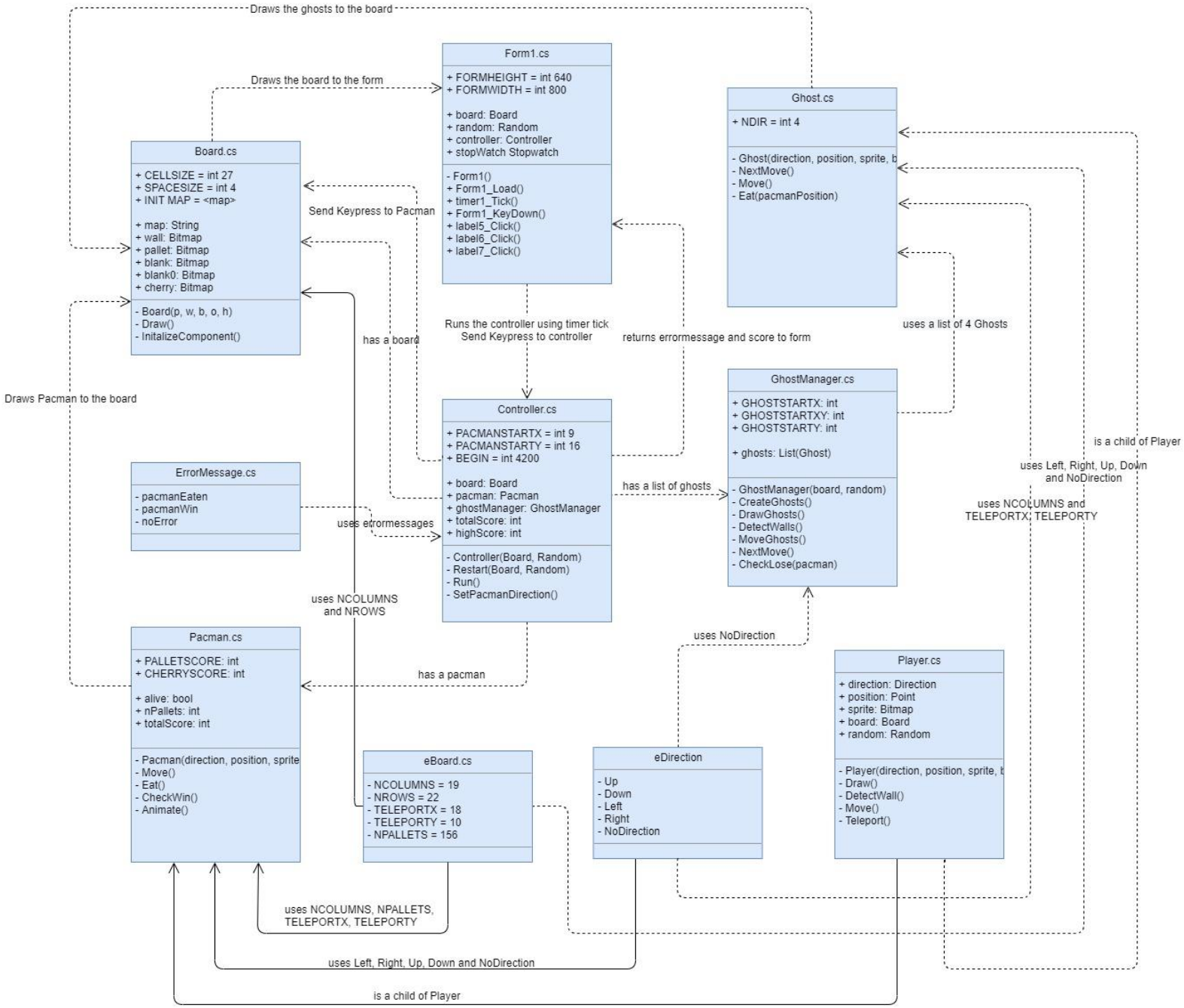
Ghost:
The Ghost class is a child of the Player class and is used to control movement of the ghost sprites around the screen and determine game actions when they eat Pacman.

GhostManager:
The GhostManager class is used to create a list of ghosts that are drawn to the form individually, run from the Ghost class.

Board:
The board class is used to determine the layout of the game board, the number of items on the board (kibble, powerups etc) and draw the game board to the screen.

Form1:
Form1.cs is the class that is directly associated with the form that is drawn onto the screen, and contains the code for when the user interacts with the form to directly control the game

Step 4: Encapsulation



Step 5: Iterative Refinement

Controller:

Controller – The constructor initializes the fields required to run the game as well as passing fields that are needed to run the game
Restart – Re-initalizes all the fields to restart the game.
Run – Runs the game, initializes all required components of the game board, updates the highscore file and checks for win/loss conditions to return to the form
SetPacmanDirection - passes the keypress from the form to pacmans class to control his movement during gameplay

eDirection:

eDirection – Contains the enumerations the directions of movement for Pacman and the ghosts.

eBoard:

eBoard – Contains the enumerations of the parameters for the game board, such as the columns and rows, the locations of the teleport tunnels and the total number of pallets on the game board

ErrorMessage:

ErrorMessage – Enumeration that contains the error messages used for detection of game win and loss states.

Player:

Player – Initializes the fields required for the Player classes and passes through the fields required to allow the Ghost and Pacman classes to operate
Draw – Draws the sprites to the game board using bitmap Images
Move – Controls the movement of the sprites around the game board on the screen.
Teleport – Detects when the position of a sprite is in the teleport tunnel location and teleports that sprite to the exit position on the opposite side of the game board
DetectWall – Detects when the position of a sprite is trying to move into a wall so that it doesn't pass through the wall on the game board

Pacman:

Pacman – Initializes the fields required for the Pacman class and passes through the fields required to control Pacman on the game board.
Move – Controls the movement of Pacman around the game board, taking directional inputs from the arrow keys on the keyboard.
Draws the Pacman sprite to the game board using the row and cell positional values
Eat – Checks when pacman is in the same position as a pallet or cherry and sets that cell of the game board to be blank while incrementing the cherry or pallets eaten up by one, also calculates the total score from eating these items to pass to the controller.

CheckWin – Compares the amount of pallets eaten to the total number of pallets on the gameboard, the returns the win Boolean when the player has eaten all pallets on the game board.
Animate – Animates pacman using multiple bitmap images to simulate an eating motion as Pacman moves around the game board and a Boolean located in the Move method to switch between open and closed.

Ghost:

Ghost – Initializes the fields required for the Ghost class and passes through the fields required to draw the Ghosts on the game board.
NextMove – Uses random number generation to determine the next movement direction of a ghost once it hits a wall and become stationary
Move – Controls the movement of the Ghosts around the game board
Eat – Detects when the ghost has encounters pacman and returns a Boolean when he is "eaten"

GhostManager:

GhostManager – Initializes the fields required for the Ghost Manager class
DrawGhosts – Draws the ghosts to the screen using bitmap images
DetectWalls – Executes the Teleport and Wall detection methods for each ghost per timer tick
MoveGhosts – Executes the Move method for each ghost per timer tick
CheckLose – Checks if a ghost has eaten pacman on every timer tick and returns the value to the controller

Board:

Board : DataGridView – Defines the fields and specifies the size and layout of the game board using a string constant of characters that plot the brick and pallet positions on the game board using the DataGridView
Board – Initializes the fields and set up the string layout of the game board to be drawn to the screen as bitmap images
Draw – Draws the game board to the screen using the specified parameters above.

Form1.cs:

Form1 – Initializes the form and the fields required to be displayed and passed to the rest of the program so the user can control the game from the form.
Form1_Load – Additional method that is run on form load after initialisation, this is required to pass in the embedded font and set the correct panel visibilty for gameplay
timer1_Tick - The game is controlled by the timer, as it executes all of the functions in this method on every tick.
Form1_KeyDown - Allows the form to take key inputs from the user during gameplay
Label5_Click - The "Start/Restart" button, starts the stopwatch timer for the game and toggles all the relevant buttons, panels and controls that are needed to play the game.
Label6_Click – The "Pause/Resume" button, disables the timer which in-turn pauses the game and displays the how to play panel. Disables itself on use and enables the resume button to un-pause the game.
Label7_Click - The "Exit" button, uses application.exit to close the form.

Step 6: Now you can start coding!

Step 7: Identify extra functionality that will improve your solution.

Restart, Exit, Pause and Resume functionality

Bonus Points Cherry

High score system that is persistent across Application Exits

Teleporting tunnels at the left and right side of screen
