IN628 2020 Practical 05 – Linked List

This week you will implement the rainbow spitting chicken. In this game, the user moves the chicken left and right using the arrow keys. When the spacebar is pressed, the chicken produces a randomly coloured “pellet” from its beak. You will use a linked list to manage the pellets. Pellets travel up the screen. When a pellet moves off the top of the screen, it is no longer ‘alive’ and should be removed from the linked list to conserve memory. A count of the number of pellets currently on the screen (and thus in the linked list) is displayed.

1. Exercise 1

a. Create class Pellet to represent the coloured circles. Each ***Pellet*** must know its colour, position, velocity and size. It must have access to a ***Graphics*** object on which to draw itself. It needs a ***Random*** object to select its colour. **Important**: **Since you will be managing the Pellets in a linked list, it must have a Next pointer.** It must be able to draw itself and to move itself. **Refer to the PowerPoint for Pellet class declaration.**

b. Create a linked list class to manage the ***Pellet*** instances. As discussed in lecture, the primary data properties of a linked list are its ***head*** and ***tail*** pointers.

   Like all linked lists, yours must provide methods to add and delete nodes. Your ***AddPellet*** method should accept a prepared ***Pellet*** as an argument. Your ***DeletePellet*** method should accept as an argument the ***Pellet*** to delete (or, more accurately, a handle to the ***Pellet*** to delete).

   In addition, this linked list needs methods to move, draw and count its pellets. It also needs a method to remove all "dead" pellets from the list. Each pellet has an ***IsAlive*** property which is true while the pellet is on the screen, and should be set to false when the pellet goes off the screen. Think carefully about the correct place to change the value of ***IsAlive***.

   Note that the most common bug in this application is a crash when the last pellet moves off the top of the screen, or when launching a new pellet after the previous pellets have all exited. This occurs when you don't handle all the special cases correctly in the method to delete a single node from the list. Make sure that your list maintains its integrity when you delete the first node, the last node and/or the only node in the list. **Refer to the PowerPoint for my PelletList class declaration.**

c. Create a ***Form*** with a ***Label*** (to display the count), a ***PictureBox*** (to display the chicken) and a ***Timer*** (to drive the game cycle).

d. Write handlers for the ***MyForm_Load*** and ***MyForm_KeyDown*** events, and for the timer.

MyForm_Load*:** Create the ***Random instance, the ***Graphics*** instance, and the global ***PelletList***. (I also created a 'global' SolidBrush with the same colour as the background of the Form. Think about what that is for.)

MyForm_KeyDown*:** In response to the left and right arrows keys, move the chicken. In response to the space bar, "fire" a pellet. Recall that the user-pressed key can be read from the ***KeyDownEventArgs e that is passed into the handler when the ***KeyDown*** event is raised. As an example, my ***KeyDown*** handler contains the following statement (pbxChicken is the name of the ***PictureBox***):

```
if (e->KeyData == Keys::Left)
    pbxChicken->Left -= 15;
```

Think carefully about the code you need to execute when the spacebar is pressed. To fire a pellet, a new Pellet needs to be created and added to the linked list so that the list will draw it at the next game cycle (timer tick).

timer1_Tick: The timer code is responsible for erasing the previous display state, and issuing the necessary commands to move the pellets, delete pellets that have gone off the screen, draw the live pellets and update the count label.

2. Optional Extension

a. Add a ***SoundPlayer*** data member to ***MyForm.*** Each time the spacebar is pressed, play the sound. A **.wav** file in provided in the **sounds directory**.