**Flappy Bee**

**Technical Documentation.**

**Program** class:

The main entry point for the application.

Contains the Main() method, which enables visual styles, sets the compatible text rendering, and runs the Form1.

**Layer** class:

Represents a visual background layer in the game.

Includes properties and methods for managing the layer's position and rendering.

The **Layer**() constructor takes cut1_pos, width, and speed parameters to initialize the layer object.

The **layerReset**() method resets the layer's position and speed to their initial values.

The **SetLayerPosition**() method updates the layer's position based on the current speed and width.

**ScoreManager** class:

Manages the player's score and the maximum score achieved.

Includes properties Score and Max_score for tracking the current score and the maximum score.

The **ScoreManager**() constructor initializes the score to 0 and loads the maximum score from the "Max_score.txt" file.

The private **LoadMaxScore**() method loads the maximum score from the "Max_score.txt" file.

The **UpdateMaxScore**() method updates the maximum score if the current score exceeds the previous maximum score and saves it to the "Max_score.txt" file.

Below are various WinForm objects

```
        private System.Windows.Forms.PictureBox pipeTop;
        private System.Windows.Forms.PictureBox bird;
        private System.Windows.Forms.PictureBox pipeBottom;
        private System.Windows.Forms.PictureBox ground;
        private System.Windows.Forms.PictureBox pipeBottom2;
        private System.Windows.Forms.PictureBox pipeTop2;
        private PictureBox heart1;
        private PictureBox heart2;
        private PictureBox heart3;
         private System.Windows.Forms.Label scoreText;
        private System.Windows.Forms.Timer gameTimer;
        private System.Windows.Forms.Label GameOverLabel;
        private Timer Bird_tick;
        private PictureBox roof; // invisible roof for collision
        private System.Windows.Forms.Label MaxScore;
```

**Form1** class:

Inherits from the Form class and represents the main form of the application.

Includes various member variables and objects used in the game.

The class constructor initializes the member variables and sets up the form.

Contains event handlers and methods for game logic and interaction.

Overrides the Paint method to draw the background layers.

Member Variables:

random: An instance of the Random class for generating random numbers.

pipeSpeed: An integer representing the speed at which the pipes move at the start of the game.

game_over: A boolean indicating whether the game is over or not.

hearts: An integer representing the number of remaining lives.

layer_1, layer_2, layer_3: Image variables for the background layers.

layer1, layer2, layer3: Instances of the Layer class representing the background layers.

Scores: An instance of the ScoreManager class for managing the game score.

stopwatch: A Stopwatch object for measuring elapsed time. It is necessary to check the time of immortality

elapsedMilliseconds: A long variable storing the elapsed milliseconds.

verticalVelocity: A float representing the vertical velocity of the bird.

angle: An integer representing the angle of rotation for the bird image.

flipImageUp, flipImageDown: Image variables for the rotated bird images.

Methods:

**RotateImage**(): A static method that rotates an image by a given angle.

**gameTimerEvent**(): An event handler for the game timer tick event. Each tick shifts the pipes and the backdrop by a few pixels.

Respawn pipes if they have gone beyond the game boundary.

Updates the game score.

Speeds up the game if the player has reached 5.

**PipesOutOfBound**(): Checks if a pipe is not out of the game window.

**MovePipesLeft**(): Moves the pipes to the left by a specified speed.

**RespawnPipes**(): Respawns the pipes at a given X-coordinate with a random Y-coordinate.

**BirdTick**(): An event handler for the bird timer tick event.

**BirdIsNotOutOfBound**(): Checks if the bird is within the game window.

**RotateBird**(): Rotates the bird image based on the bird's vertical velocity.

**Intersection**(): Checks if there is an intersection between two PictureBox objects.

**MoveBirdBetweenPipes**(): Moves the bird up or down between two pipes. (Occurs when a bird crashes into a pipe)

**gameKeyisdown**(): An event handler for key down events. The bird jumps if you press the space bar, restarts the game if the game is over.

**endgame**(): Handles the end of the game when the bird collides with a pipe, ground, ceil. Called when a bird encounters obstacles, reduces lives, ends the game if there are no lives left.

**RestartGame**(): Restarts the game after the game over condition.

The function resets the counter, puts all objects in the starting positions.

**CountDown**(): Implements a short delay before restarting the game.

**Form1_Load**(): An event handler for the form's load event. It doesn't work without this function.

**Form1_Paint**(): An event handler for the form's paint event, used to draw the background layers. renders the background every Invalidate()

**Ideas**.

Increase lives by 1 (maximum 3) every n pipes passed.

Implement the start, pause button, make a menu.

You can also change the image of an object.

You can add a function that, when colliding with an object, stops the game for a second and respawns the bird, but it seemed to me a good idea to leave everything like that, because the pace of the game does not get off.

You can add sounds of collision with an object, the sound of a jump.

Make a smooth turn of the bird (+45 degrees to -45 degrees depending on Velocity)

**Existing bugs (Features)**

The bird jumps strangely on the floor if you don't press the space bar. The bird starts with some kind of delay, freezing (I'm not talking about CountDown() after restart)