Flappy Dragon Game Report

# HTML Code

The HTML code sets up the structure of the game. It includes the following elements:

* Audio Element: This element plays background music for the game. The 'src' attribute points to the audio file ('1-72. Hall of Fame.mp3').
* Canvas Element: This is the main drawing area for the game. The 'id' attribute is 'canvas1'.
* Div Elements: Two div elements ('frame-1' and 'frame-2') are used for additional frames or decorations.
* Button Elements: Two buttons ('Play' and 'Help') are provided for user interaction.
* Script Elements: Four script files ('bird.js', 'obstacles.js', 'particles.js', and 'app.js') are included to handle the game logic and animation.

# CSS Code

The CSS code styles the game with the following components:

* Global Styles: Resets margin, padding, and text alignment for all elements.
* Body Styles: Sets the background color to black to create a dark theme for the game.
* Canvas Styles: Sets dimensions (800x600px), border, position (relative), and background image for the game area.
* Button Styles: Defines a set of CSS variables for glowing effects, font styling, padding, border, and positioning.
* Button Hover and Active States: Defines transitions and glowing effects for buttons when hovered or clicked.
* Specific Styles: Sets styles for the 'play' and 'help' buttons, and frame elements (frame-1 and frame-2).

# JavaScript Code

The JavaScript code handles the game logic and animation. It includes the following components:

* Canvas Setup: Initializes the canvas and drawing context. Sets the canvas height and width, and displays the game title 'Flappy Dragon' using fillText.
* Background Handling: Manages the background image ('BG.png') and its movement to create a scrolling effect.
* Game Animation: Handles the main game loop and rendering using the 'animate' function, which updates and draws all game components.
* Game Start and End: Defines 'startGame' and 'endGame' functions to control the game's start and end behavior, including resetting variables and playing/pausing the audio.
* Event Listeners: Adds event listeners for the 'Play' button and space key to start the game and control the bird's movements respectively.
* Obstacle Handling: Manages obstacle creation, drawing, and updates using the 'Obstacle' class and 'handleObstacles' function.
* Collision Detection: Checks for collisions between the bird and obstacles using the 'handleCollisions' function.
* Bird Class: Defines the bird's properties, update logic, and drawing using the 'Bird' class. The bird's movements are controlled by the space key.
* Particle Handling: Manages particle effects for visual enhancements using the 'Particle' class and 'handleParticles' function.