Author: Mustafa Khalid. Date: 12 January 2026. Course: Introduction to Programming

Flappy Bird Console Game – Detailed Section-by-Section Explanation

1. Libraries Used

- **iostream**: Standard input/output operations.
- **cstdlib & ctime**: Random number generation.
- **thread & chrono**: For pausing game frames.
- **Platform-specific headers**:
- Linux/macOS: <sys/ioctl.h>, <termios.h>, <unistd.h>
- Windows: <conio.h>, <windows.h>

2. Constants

- `width` (30) and `height` (10): define the size of the game area.

3. Functions Explained

showMenu()

Displays the main menu options: Start Game, Instructions, Exit.

showInstructions()

Shows the instructions: how to jump ('w' or Space), gravity, scoring, and avoiding pipes.

startGame(int difficulty)

- Initializes bird position at the middle.
- Sets pipe X at the far right.
- Randomly generates pipe gap.
- Sets game speed based on difficulty.
- Main game loop:
- Clears the screen.
- Draws the bird, pipe, and score.
- Reads user input non-blockingly.
- Updates bird position (gravity + jump).
- Moves pipe left.
- Resets pipe and increments score if passed.
- Checks collision.
- Ends game if collision occurs.

drawGame(int birdY, int pipeX, int gapY, int score, int highScore)

- Loops over the grid.
- Draws bird ('O') at fixed X and current Y.
- Draws pipe ('|') except for gap.
- Displays score and high score.

updateBird(int &birdY, char input)

- Applies gravity by increasing birdY.
- If 'w' or Space pressed, birdY decreases (jump).
- Clamps birdY between 0 and height-1.

checkCollision(int birdY, int pipeX, int gapY)

- Checks if bird X is at pipe X.
- If bird Y not in gap, returns true (collision).

clearScreen()

- Clears terminal screen using system-specific commands or ANSI escape codes.

sleepMs(int ms)

- Pauses game for a number of milliseconds between frames.

4. Input Handling

- Non-blocking using _kbhit() and _getch().
- Works for Linux/macOS and Windows.
- Player can jump using 'w' or Space.

5. Game Mechanics

- **Gravity**: Bird falls automatically.
- **Jump**: Player input moves bird up.
- **Pipe Movement**: Pipes move from right to left, reset at screen edge.
- **Scoring**: Passing pipe increases score.
- **Collision Detection**: Bird hitting pipe ends the game.

6. Game Flow Summary

1. Display menu.
2. Select Start, Instructions, or Exit.
3. Start Game:
4. Initialize bird, pipe, gap, speed.
5. Loop:
    - Clear screen

        - Draw frame
        - Read input
        - Update bird
        - Move pipe
        - Check collisions
        - Update score
        - End game if collision
    6. Return to menu.

This structure demonstrates: - **Functions** for code organization. - **Selection statements** (if, else, switch) for decisions. - **Loops** (while) for game frames. - **User input handling** without blocking. - **Randomization** for pipe gaps. - **Console graphics** for simple game visualization.

The project is fully playable on Linux terminals and online compilers, with bird jumping using 'w' or Space and automatic pipe movement.