HARIBIRD'S TINY WING ESCAPE

GROUP 3

BSCS 2-1

MEET THE TEAM

BD FD

Venelyn

UI BD

Angeli

UI FD

Lorelie

UI BD

Alexi

Shawn

FD BD

Christian

FD BD

Aliyah

FD D

Maverick

FD D

INTRODUCTION

Inspired by Flappy Bird, a development team created Haribird, an arcade adaptation written entirely in assembly language. This low-level programming approach allowed direct communication with computer hardware using symbolic machine instructions. The developers aimed to showcase the technical precision and optimization possible with assembly language. They also wanted to preserve the original game’s fast-paced, addictive gameplay and minimalist graphics. Haribird serves as a tribute to retro gaming culture and low-level programming capabilities. A new team is now refining Haribird into a version called HariBird's Tiny Wing Escape. This updated version keeps the core gameplay mechanics of the original while improving functionality and performance. Enhancements include more responsive controls, better collision detection, and additional visual and audio elements. The improvements aim to create a smoother and more immersive gaming experience. Overall, the project continues to blend nostalgic arcade charm with modern game design standards, emphasizing both technical skill and creative innovation.

ENHANCEMENTS

1. Pause / Restart / Quit Options

2. Power-Ups
    - Extra Life
    - Invincibility (Press “I”)

3. Audio Feedback
    - Background Music
    - Flap Sound
    - Collision Sound
    - Score Sound
4. High Score

DEFINITION OF TERMS

ADDICTIVE MECHANICS

Game design elements that strongly encourage repeated play due to their engaging or challenging nature.

ARCADE ADAPTATION

A version of a game designed to mimic or run on arcade-style machines, often with fast-paced and skill-based gameplay.

COLLISION DETECTION

A programming technique used to determine when two objects in a game interact or come into contact.

CORE GAMEPLAY MECHANICS

The essential actions and rules that define how a game operates and how players interact with it.

DEFINITION OF TERMS

HIGH-LEVEL PROGRAMMING LANGUAGE

A programming language like Python, Java, or C++ that is closer to human language and abstracts away hardware details.

ASSEMBLY LANGUAGE

A low-level programming language that provides direct control over a computer's hardware using symbolic representations of machine code.

LOW-LEVEL PROGRAMMING

Programming that interacts closely with hardware, typically using assembly or machine code. It allows for high performance but is more complex and error prone.

DEFINITION OF TERMS

OPTIMIZED EXECUTION

Code that is written or refined to run more efficiently, using less memory or processing power.

FRAMEWORK

A base structure used by developers to build and maintain applications, often providing pre-written code and libraries.

TECHNICAL PROFICIENCY

A high level of skill or expertise in programming or system design, often involving detailed knowledge of computer systems.

USER ENGAGEMENT

The degree to which players are involved and interested in a game, often influenced by design, mechanics, and feedback systems.

DEFINITION OF TERMS

CREATIVE INNOVATION

The act of introducing new ideas or methods in design and development to create unique or enhanced user experiences.

INDIE GAME DEVELOPMENT

The process of creating games by individuals or small teams without financial support from large publishers.

NASM

An open-source assembler for the x86 architecture that translates assembly language code into machine code or binary executable files. It is widely used for programming in low-level languages on Intel and AMD processors.

HOW TO PLAY

MAIN MENU

After the game starts, you'll see the main menu. Here, you can:

- Start a new game
- Choose a difficulty level (Easy, Medium, Hard)
- How to play
- Quit the game

LEVELS

- Easy – Slower game speed, ideal for beginners.
- Medium – Moderate game speed for a balanced experience.
- Hard – Fast game speed for a more intense challenge.

HOW TO PLAY

CONTROLS

- Press any key (except key controls) or click the mouse to flap and lift the bird upward.
- Tap repeatedly to keep flying and avoid crashing into pipes. Timing is key.

POWER-UPS

- Extra Life: Grants one respawn if you crash. It brings you back to where you left off.
- Invincibility (Press “I”): Temporarily makes the bird immune to collisions.

HOW TO PLAY

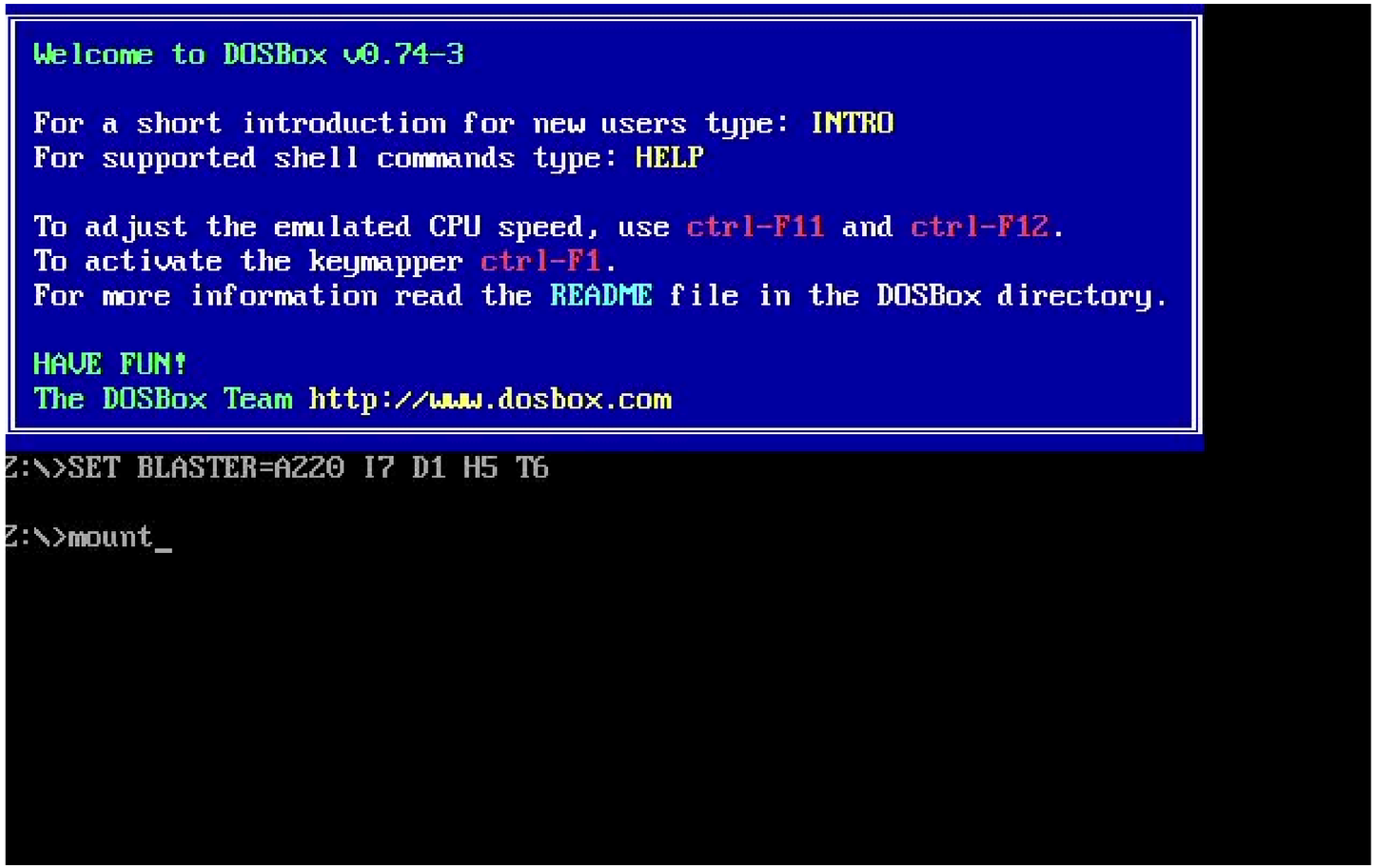
OPTIONS

- Pause/Resume – Pause the game anytime and continue later.
- Restart – Start the current game over from the beginning.
- Quit – Exit the game and return to the main menu.

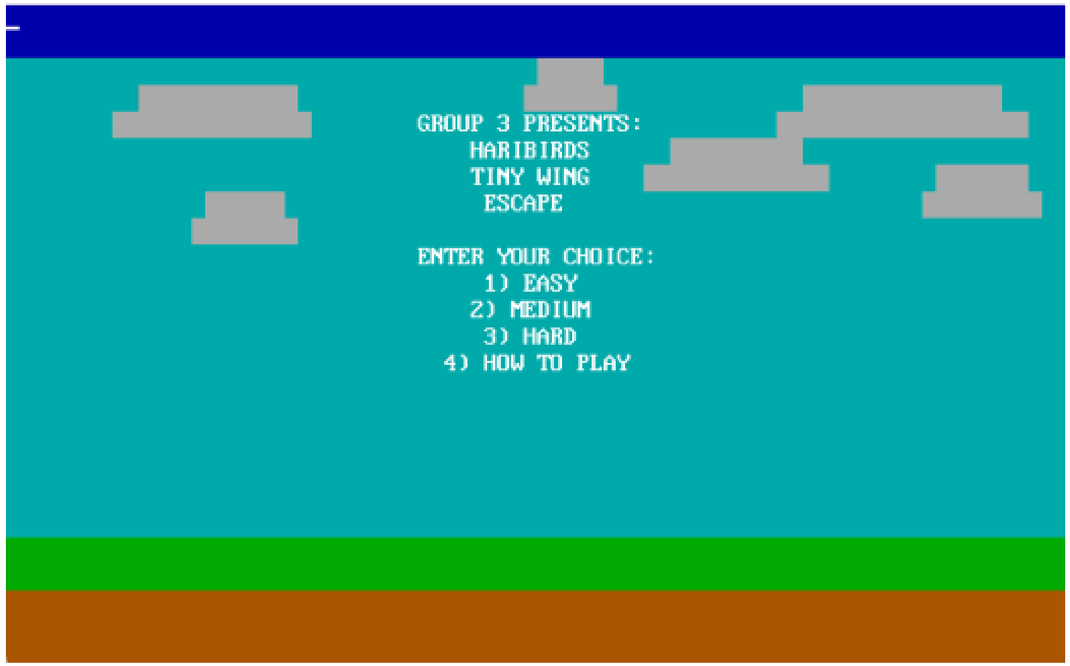
SCORING

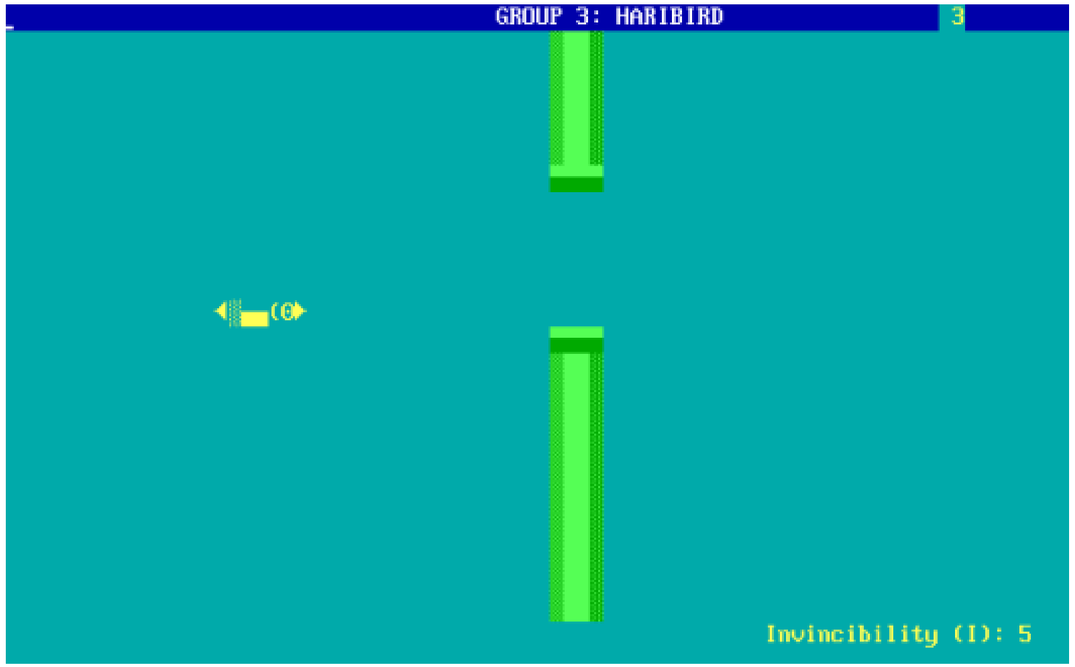
- You earn 1 point for every pair of pipes you pass through.
- When the game ends, you will see your score and current high score.

VIDEO DEMO


Welcome to DOSBox v0.74-3
For a short introduction for new users type: INTRO
For supported shell commands type: HELP
To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.
HAVE FUN!
The DOSBox Team http://www.dosbox.com
Z:\>SET BLASTER=A220 I7 D1 H5 T6
Z:\>mount_

UI DESIGN


GROUP 3 PRESENTS:
HARIBIRDS
TINY WING
ESCAPE
ENTER YOUR CHOICE:
1) EASY
2) MEDIUM
3) HARD
4) HOW TO PLAY


GROUP 3: HARIBIRD
3
Invincibility (I): 5

HOW TO INSTALL

TO COMPILE AND RUN THE GAME ON DOSBOX, FOLLOW THESE:

Download and Install NASM (Netwide Assembler)

1. Ensure **NASM** is installed on your system. You can download it through the link below:
    a. **bit.ly/4kpwUa4**
2. Specifically, Download the **Assmsoft** Zip File from the link

Get the Haribird's Tiny Wing Escape Source Code

1. Install it through our official **GitHub** link and specifically download the **Haribird.asm** file:
    a. **https://github.com/223344556611/Haribird**

HOW TO INSTALL

Move the Folder to Drive C

1. After installing the **Assmsoft** zip file, Extract the Assmsoft folder to your C Drive for easy-access
2. Insert the **Haribird.asm** inside the Assmsoft Folder

Assemble the Source Code

1. Once ready, open **DOSBox** and type the Instructions below line by line:
    a. The first step is to mount any particular virtual drives that are not currently in use **“mount x \assmsoft”**
    b. Then compile the source code using the command **“x: nasm Haribird.asm -o Haribird.com -l Haribird.lst”**
    c. Lastly, run the compiled code using the command **“Haribird.com”**


GAME IS OVER!
THANK YOU