J&J PRESENTS

QUIZ MO KO !!!

A GAME AND USER MANUAL

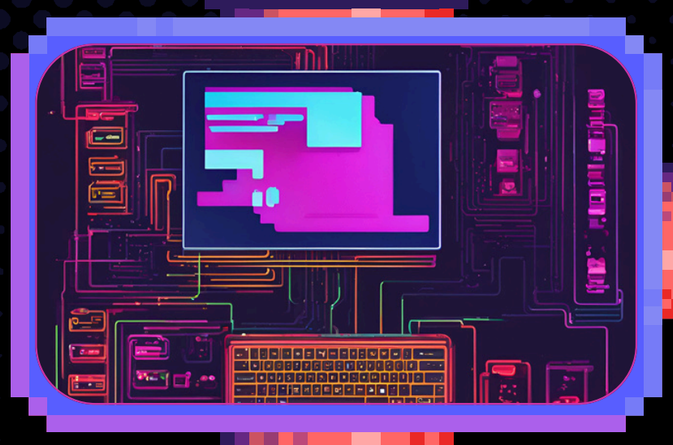
TABLE OF CONTENTS

INTRODUCTION

THE HISTORY OF THIS GAME DATES BACK IN OUR CMSC 21 CLASS, WHEN WE WERE ASSIGNED TO MAKE A GAME AS A FINAL PROJECT. THIS GAME IS A TRIVIA GAME FOUNDED UPON THE IDEA OF LEARNING WHILE HAVING FUN. THIS GAME OFFERS VARIOUS CATEGORIES AND DIFFERENT LEVELS OF DIFFICULTY AS THE PLAYER PROGRESS THROUGH THE GAME. THERE ARE ALSO POWER-UPS AND GAME TRICKS THAT CAN HELP THE PLAYER WIN THE GAME.

HENCE,QUIZ MOKO IS AN ENGAGING AND EDUCATIONAL TRIVIA GAME DESIGNED TO FOSTER LEARNING THROUGH FUN AND INTERACTIVE GAMEPLAY. WE AIM TO CREATE AN EXCITING AND CAPTIVATING EXPERIENCE THAT CHALLENGES PLAYERS OF ALL AGES WITH A DIVERSE RANGE OF CATEGORIES AND VARYING DIFFICULTY WHILE ENSURING QUALITY ENTERTAINMENT AND INTELLECTUAL GROWTH AND ENHANCEMENT OF KNOWLEDGE.

GAME STRUCTURE AND SOFTWARE BACKGROUND

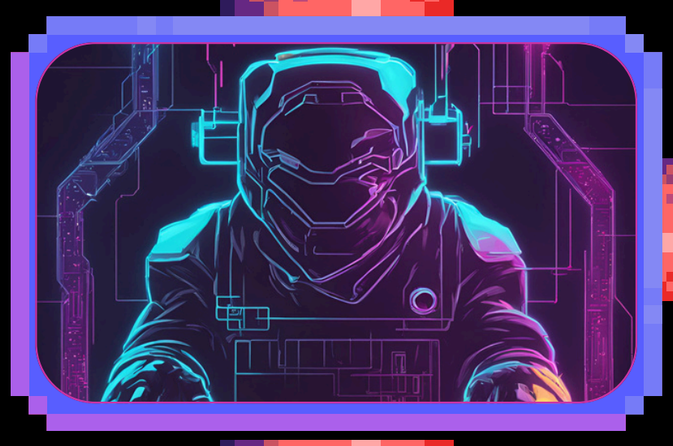


FILE MANIPULATION

Primarily used in handling the game's leaderboard. This ensures that the player's game data and scores are maintained.

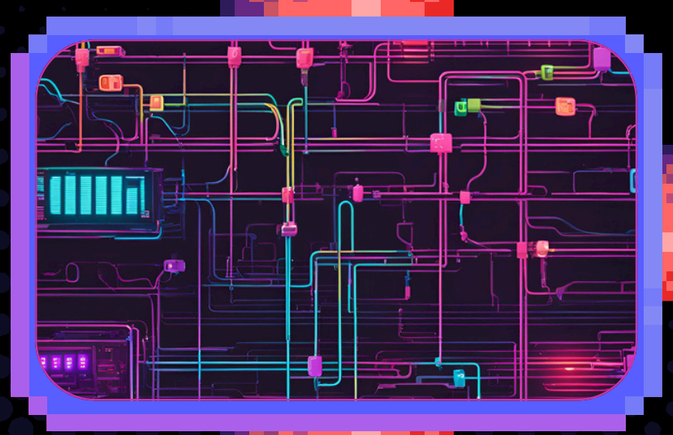
MEMORY ALLOCATION

This dynamically allocates memory for player's data such as scores, name, and others. This allows the game to be flexible in handling player data.



STRUCTS

This game employs the concept of structs to group related data such as player's name and score for easier management and access.



LINKED LIST

This game employs the concept of linked list to efficiently handle dynamic game elements such as in showing and removing categories.

HEADER FILES AND SOURCE FILES

GAME.H HEADER FILE

GAME.H IS A HEADER FILE THAT DEFINES THE STRUCTURES, CONSTANTS, ENUMERATIONS, AND FUNCTION PROTOTYPES USED THROUGHOUT THE GAME.

```
#pragma once

#include <stdbool.h>

#define MAX_QUESTIONS 10
#define TOTAL_QUESTIONS 10

#define MAX_OPTIONS 4
#define TOTAL_OPTIONS 4

#define MAX_CATEGORIES 5
#define TOTAL_CATEGORIES 5

#define MAX_POWERUPS 3
#define TOTAL_POWERUPS 5

#define MAX_NAME_LENGTH 10
```

CONSTANTS

- DEFINE LIMITS AND AND SIZES FOR VARIOUS GAME ELEMENTS.

STRUCTS AND ENUMERATIONS

- STRUCTS GROUP RELATED DATA TOGETHER. WHILE ENUMERATIONS ARE USED TO DEFINE LEVEL DIFFICULTY.

```
enum Difficulty {
    EASY,
    AVERAGE,
    DIFFICULT
};

typedef struct {
    char name[20];
    int score;
} Score;

typedef struct {
    char* level;
    char* category;
    char* question;
    char* option[4];
    char* correct_answer[4];
} Quiz;
```

FUNCTION PROTOTYPES AND EXTERNAL VARIABLES

```
61          int used_powerup[5];
62          int clicked_powerup[5];
63
64          int timer;
65      } Game;
66
67
68      Quiz create_quiz(const char* level, const char* category, cons
69      Powerup create_powerup(const char* name, const char* descripti
70
71      void init_contents(Game* game);
72      void add_quiz(Game* game, const Quiz* quiz);
73      void add_powerup(Game* game, const Powerup* powerUp);
74
75      void init_player(Game* game);
76      void confirm_name(Game* game);
77
78      void play_quiz(Game* game, int start_index, int *que_index, in
79      int init_timer(Game* game);
80      void init_difficulty(Game* game);
81      int choose_category(Game* game);
82      void mark_category_used(int categoryIndex, Game* game);
83      void init_index(int *array, int size, int modulo, int multiple
84      int is_duplicate(int *array, int size, int value);
85      int is_correctanswer(char user_answer, Quiz quiz);
```

```
88      void center_item(int size);
89      void add_spaces(int size, int max_size);
90      void clear_screen();
91
92      void write_score(char *name, int score);
93      void read_score(Score **scores, int *numScores);
94
95      void print_top_scores(Score scores[], int numScores, int n);
96
97
98      void press_continue();
99
100     extern char* separator;
101     extern char* title_screen;
102     extern char* welcome_screen1;
103     extern char* welcome_screen2;
104     extern char* welcome_screen3;
105     extern char* welcome_screen4;
106     extern char* welcome_screen5;
107     extern char* welcome_screen6;
108     extern char* cat_easy_average;
109     extern char* cat_hard;
110     extern char* leaderboards;
111     extern char* lose_screen;
112     extern char* win_screen;
```

```
113     extern char* about_screen;
114     extern char* show_category;
115     extern char* easy_remark1;
116     extern char* easy_remark2;
117     extern char* easy_remark3;
118     extern char* average_remark1;
119     extern char* average_remark2;
120     extern char* average_remark3;
121     extern char* difficult_remark1;
122     extern char* difficult_remark2;
123     extern char* difficult_remark3;
124
125
126     extern int cat_flg1;
127     extern int cat_flg2;
128     extern int cat_flg3;
129     extern int cht1;
130     extern int cht2;
131     extern int dif1, dif2, dif3;
132     extern int opt[4];
133     extern bool used_easy_category[10];
134     extern bool used_average_category[10];
135     extern bool used_hard_category[10];
136
137     extern int used_powerup[3];
```

- FUNCTION PROTOTYPES DEFINE THE FUNCTIONS THAT WILL BE USED IN THE SOURCE FILES.

- EXTERNAL VARIABLES ARE VARIABLES DECLARED IN THE HEADER FILE THAT WILL BE ACCESSIBLE TO MULTIPLE SOURCE FILES.

GAME.C SOURCE FILE

THE GAME.C FILE CONTAINS THE MAIN IMPLEMENTATION OF THE GAME LOGIC. CONTAINS THE DEFINITION OF THE DECLARED FUNCTION PROTOTYPES IN THE GAME.H HEADER FILE.

```c
void write_score(char *name, int score) {
    FILE *file = fopen("assets/scores.txt", "a");
    fprintf(file, "%s %d\n", name, score);
    fclose(file);
}

void print_top_scores(Score scores[], int numScores, int n) {
    puts(separator);
    printf("\n\n");
    puts(leaderboards);
    printf("\n");
    center_item(53);
    printf("----TOP 5 SCORERS----------
    center_item(45);
    printf("RANK");
    add_spaces(strlen("RANK"), 10);
    printf("NAME");
    add_spaces(strlen("NAME"), 20);
    printf("SCORE\n");
    center_item(53);
    printf("----------
    center_item(45);
```

FILE HANDLING FUNCTIONS (FILE MANIPULATION)

- READ AND WRITES SCORES FOR LEADERBOARD MAINTENANCE.

QUIZ MANAGEMENT (POINTERS APPLICATION)

- HANDLING OF QUIZ GAMEPLAY, PLAYER INTERACTION, AND GAME PROGRESSION.

```c
Quiz create_quiz(const char* level, const char* category, co
    Quiz quiz;
    quiz.level = strdup(level);
    quiz.category = strdup(category);
    quiz.question = strdup(question);
    for (int i = 0; i < 4; i++) {
        quiz.option[i] = strdup(option[i]);
        quiz.correct_answer[i] = strdup(correct_answer[i]);
    }
    return quiz;
}

void add_quiz(Game* game, const Quiz* quiz) {
    game->quizzes = realloc(game->quizzes, (game->num_quizzes
    game->quizzes[game->num_quizzes] = *quiz;
    game->num_quizzes++;
}

Powerup create_powerup(const char* name, const char* descripti
    Powerup powerup;
    powerup.name = strdup(name);
    powerup.description = strdup(description);
    return powerup;
}
```

```
119         for (int i = 0; i < *numScores - 1; i++) {
120             for (int j = i + 1; j < *numScores; j++) {
121                 if ((*scores)[i].score < (*scores)[j].score) {
122                     Score temp = (*scores)[i];
123                     (*scores)[i] = (*scores)[j];
124                     (*scores)[j] = temp;
125                 }
126             }
127         }
128     }
129
130
131     void init_contents(Game* game) {
132         Quiz quiz_data[] = {
133             {"EASY", "MATHEMATICS", "CHOOSE A FACTUAL OPTION", {"ALL
134             {"EASY", "MATHEMATICS", "WHAT IS THE ONLY PRIME NUMBER TH
135             {"EASY", "MATHEMATICS", "HOW MANY SHORT STRAIGHT LINES AR
136             {"EASY", "MATHEMATICS", "WHAT NUMBER COMES FIRST WHEN ALL
137             {"EASY", "MATHEMATICS", "CHOOSE THE CORRECT OPTION.", {"S
138             {"EASY", "MATHEMATICS", "IS ZERO A WHOLE NUMBER", {"YES",
139             {"EASY", "MATHEMATICS", "WHAT IS THE DERIVATIVE OF A CONS
140             {"EASY", "MATHEMATICS", "WHAT IS THE ONLY NUMBER THAT IS
141             {"EASY", "MATHEMATICS", "WHAT IS THE LONGEST SIDE OF A TR
142             {"EASY", "MATHEMATICS", "WHICH OF THE FOLLOWING POSITIVE
143
```

INITIALIZATION OF QUESTIONS (ARRAY APPLICATION)

- INITIALIZES THE CONTENTS OF THE QUIZ.

PLAYER INITIALIZATION (UTILIZATION OF DYNAMIC MEMORY ALLOCATION)

- ALLOCATING MEMORY DYNAMICALLY DURING RUN TIME SINCE THE NAME OF PLAYER IS VARYING IN LENGTH.

```
398         game->player.name = malloc((MAX_NAME_LENGTH + 1) * sizeo
399         scanf("%10s", game->player.name);
400         printf("");
401
402         for (int i = 0; game->player.name[i]; i++) {
403             game->player.name[i] = toupper(game->player.name[i]);
404         }
405
406         game->player.score = 0;
407         game->player.lives = 3;
408
409         memset(game->clicked_powerup, 0, sizeof(game->clicked_powe
410         memset(game->used_powerup, 0, sizeof(game->used_powerup));
411         for (int i = 0; i < 10; i++) {
412             used_easy_category[i] = false;
413             used_average_category[i] = false;
414             used_hard_category[i] = false;
415         }
416         cat_flg1 = 0;
417         cat_flg2 = 0;
418         cat_flg3 = 0;
419     }
420
421     void confirm_name(Game* game) {
```

```
430     myNode* get_node(SLList* list, int index) {
431         myNode* current = list->head;
432         for (int i = 0; i < index; i++) {
433             current = current->next;
434         }
435         return current;
436     }
437
438     myNode* create_node(char* x) {
439         myNode* a = (myNode*)malloc(sizeof(myNode));
440         a->value = strdup(x);  // Allocate memory for the string
441         a->next = 0;
442         return a;
443     }
444
445     void init_list(SLList* list) {
446         list->head = 0;
447         list->tail = 0;
448         list->size = 0;
449     }
450
451     void populate_list(SLList *list, int *index, Quiz *quizzes, i
452         if (current < num_items) {
453             insert_item(list, current, quizzes[index[current]]
```

SHOWING AND REMOVING OF CATEGORIES (LINKED-LIST)

- USED IN SHOWING AND REMOVING CATEGORIES. IT REMOVES A CATEGORY WHENEVER THE PLAYER CHOOSES IT.

MAIN.C SOURCE FILE

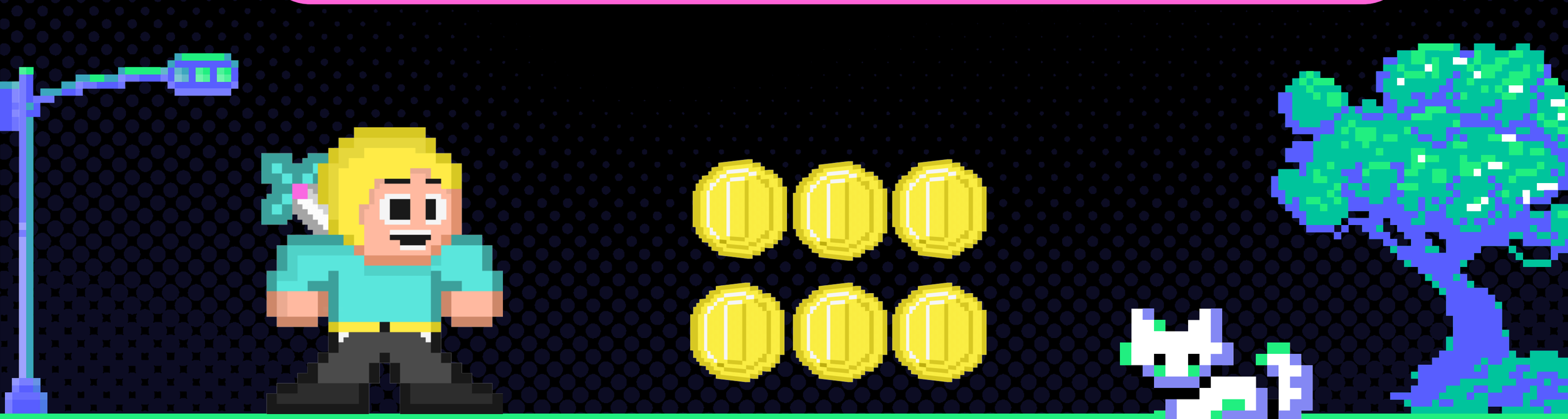
THE MAIN. C FILE IS THE MAIN DRIVER OF THE GAME. IT SETS UP THE INITIAL GAME STATE. HANDLE USER INPUT, MANAGES THE MAIN GAME LOOP, AND TRANSITIONS BETWEEN DIFFERENT GAME INTERFACE OR STATE.

```c
#include <stdio.h>
#include <time.h>
#include <conio.h>
#include <windows.h>

#include "game.h"

int main() {
    srand(time(NULL));

    Game game;
    Score* scores = NULL;

    init_contents(&game);

    while(1) {
        start:
        system("cls");

        puts(title_screen);
        PlaySound(TEXT("assets/intro.wav"), NULL, SND_FILENAME | SND_ASYNC

        char input;
        do {
            input = getch();
        } while ((input < '1') && (input > '4'));

        int pow_index[MAX_POWERUPS];
        init_index(pow_index, MAX_POWERUPS, TOTAL_POWERUPS, 1);
```

INCLUDES, GLOBAL VARIABLES, AND THE MAIN FUNCTION

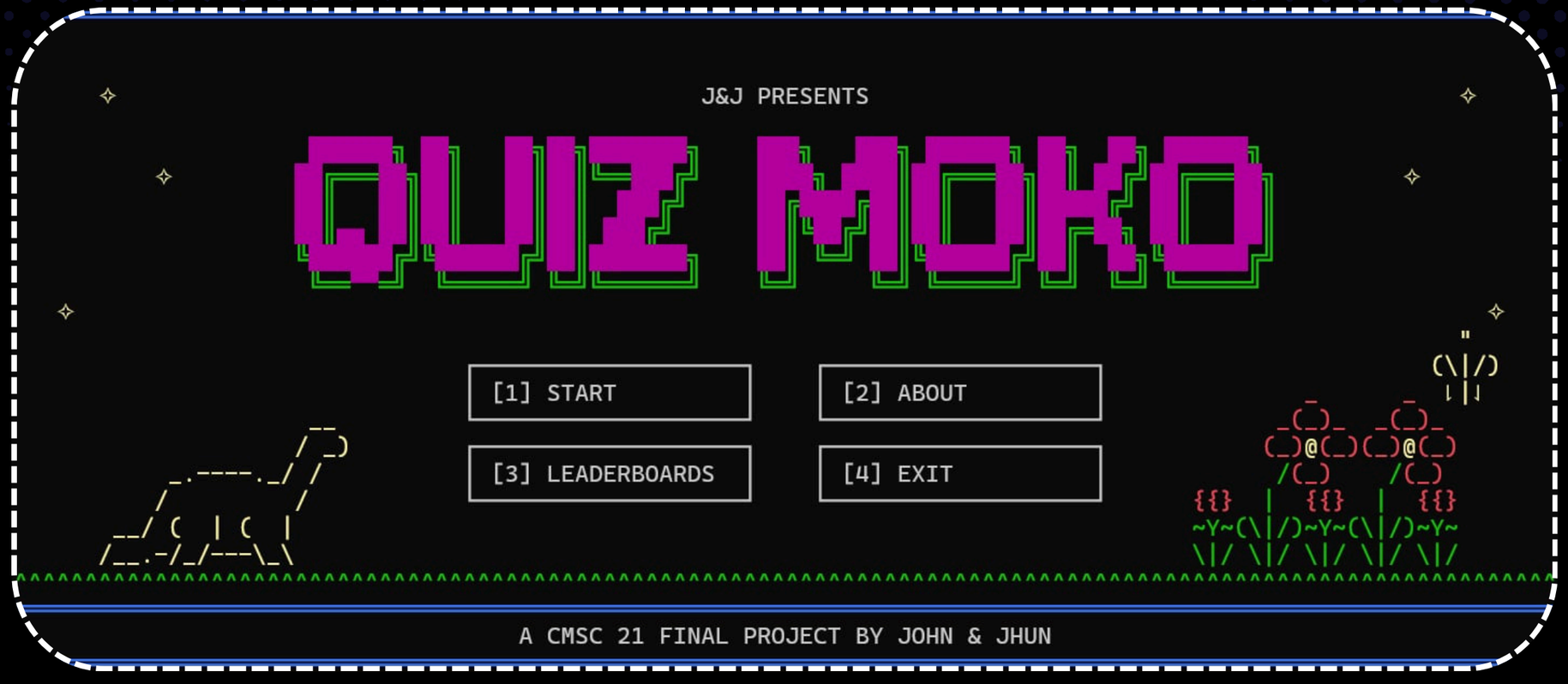
- INCLUDES ENABLE THE PROGRAM TO UTILIZE STANDARD AND CUSTOM LIBRARIES WHILE GLOBAL VARIABLES PROVIDE A WAY TO MAINTAIN SHARED STATE ACROSS THE PROGRAM.

IF YOU WISH TO PLAY OUR GAME OR VIEW OUR PROGRAM IN A MORE DETAILED MANNER, YOU MAY VISIT THIS LINK [HTTPS://GITHUB.COM/JLESCARLAN11/CMSC-21-PROJECT] AND DOWNLOAD THE ZIP FILE OF OUR SOURCE CODES. IN THE LINK, YOU WILL FIND THE EXE FILE AND INSTRUCTIONS FOR COMPILING IT.



GAME CONCEPTS AND MECHANICS

MAIN MENU INTERFACE



THE MAIN MENU PROVIDES YOU WITH OPTIONS TO NAVIAGATE THROUGH THE GAME. YOU MAY DO THE FOLLOWING TO NAVIGATE THE GAME:

PRESS 1: TO START THE GAME
PRESS 2: TO KNOW MORE ABOUT THE GAME, E.G, THE GAME MECHANICS
PRESS 3: TO ACCESS THE LEADERBOARDS AND SEE THE HIGH SCORES
PRESS 4: TO EXIT THE GAME IF DONE PLAYING

ABOUT SECTION

PRESSING 2 WILL GIVE YOU ACCESS TO THE ABOUT SECTION OF OUR GAME. THIS PART WILL SHOW YOU THE OVERVIEW, BACKGROUND, GAME MECHANICS, AND ANYTHING ABOUT OUR GAME.

HEY Y'ALL
THIS GAME WAS MADE BY LESTER AND MERL AS PROJECT IN CMSC 21

PRESS [SPACEBAR] TO CONTINUE.

WHAT IS THIS ALL ABOUT?

QUIZ MO KO CHALLENGES YOUR KNOWLEDGE AND INTUITION WITH MULTIPLE CHOICES ACROSS VARIOUS CATEGORIES. AND GUESS WHAT? YOU'RE NOT ALONE IN THIS. A ZEALOUS GAME MASTER IS ALWAYS THERE TO WELCOME YOU AND KEEP THE ENERGY HIGH THROUGHOUT THE GAME.

PRESS [SPACEBAR] TO CONTINUE.

GENERAL MECHANICS

1) THE GAME SHALL CONSIST OF THREE CONSECUTIVE LEVELS: EASY, AVERAGE, AND DIFFICULT.
2) AT THE START OF THE GAME, THE PLAYERS ARE ASKED TO INPUT THEIR NAME. THIS WILL BE USE TO RECOGNIZE ACHIEVEMENT OF THE PLAYERS AT THE END OF THE GAME.
3) AFTER THE PLAYER INPUT THERE NAME, THEY WILL BE GIVEN THREE RANDOM CATEGORIES TO TO CHOOSE AT. AS OF NOW, THE ONLY CATEGORY AVAILABLE ARE: MATHEMATICS, SCIENCE, COMPUTER SCIENCE, GEOGRAPHY, AND HISTORY AND LITERATURE. FOR EACH CATEGORY THERE EXISTS TEN MULTIPLE QUESTIONS THAT WILL BE ANSWERED BY THE PLAYERS.
4) BY DEFAULT, THE PLAYER WILL HAVE THREE LIVES AND WILL BE BE GIVEN A THREE RANDOM POWER-UPS THAT WILL HELP THEM THROUGHOUT THE GAME.

PRESS [SPACEBAR] TO CONTINUE.

GENERAL MECHANICS

5) AFTER CHOOSING A CATEGORY, THE QUESTIONS UNDER EACH CATEGORY WILL BE PRESENTED, AND THEY WILL PRESS THEIR ANSWERS FROM A TO D.
6) EACH CORRECT ANSWER WILL BE AWARDED WITH CORRESPONDING POINTS AND ANSWERING INCORRECTLY WILL RESULT IN LOSING A LIFE. EACH QUESTION SHALL BE TIMED, AND THE LESS TIME CONSUMED BY THE PLAYER, THE HIGHER THE POINTS WILL BE GIVEN.
7) AFTER GARNERING A TOTAL OF 2000 POINTS AT THE EASY ROUND, THE PLAYER WILL NOW MOVE TO THE AVERAGE ROUND. AFTER GARNERING A TOTAL OF 6500 POINTS, THE PLAYER WILL THEN BE ABLE TO LEVEL UP TO THE DIFFICULT ROUND. FINALLY, THE PLAYER WINS ONCE THEY ACHIEVE A TOTAL OF 17500 POINTS.

PRESS [SPACEBAR] TO CONTINUE.

ABOUT SECTION

GET TO KNOW YOUR POWER-UPS

THESE POWER-UPS WILL GIVE YOU THE UPPER HAND IN WINNING THE GAME. USE THEM WISELY!

TIME FREEZE	- THIS POWER-UP ALLOWS THE PLYER TO PAUSE THE TIMER OF THE GAME AND ALLOWS THE PLAYER TO ANSWER THE QUESTIONS WITH EASE.
ERASER	- THIS POWER-UP ELIMINATES ONE WRONG OPTION AMOUNG THE CHOICES.
DOUBLE JEOPARDY	- THIS POWER-UP DOUBLES THE PLAYER'S SCORE IF THE QUESTION IS ANSWERED INCORRECTLY.
PASS	- THIS ALLOWS PLAYER TO PASS AND ANSWER THE QUESTION LATER.
IMMUNITY	- THIS ALLOWS THE PLAYER TO ANSWER THE SAME QUESTION TWICE WITHOUT LOSING A LIFE IN THE CASE OF A WRONG ANSWER.

PRESS [SPACEBAR] TO CONTINUE.

THE ABOUT SECTION ALLOWS YOU TO BECOME FAMILIAR WITH IMPORTANT GAME DETAILS AND CONCEPTS THAT WILL COME HANDY AS YOU PROGRESS THROUGH THE GAME.

LEADERBORDS

LEADERBOARDS

----TOP 5 SCORERS----------

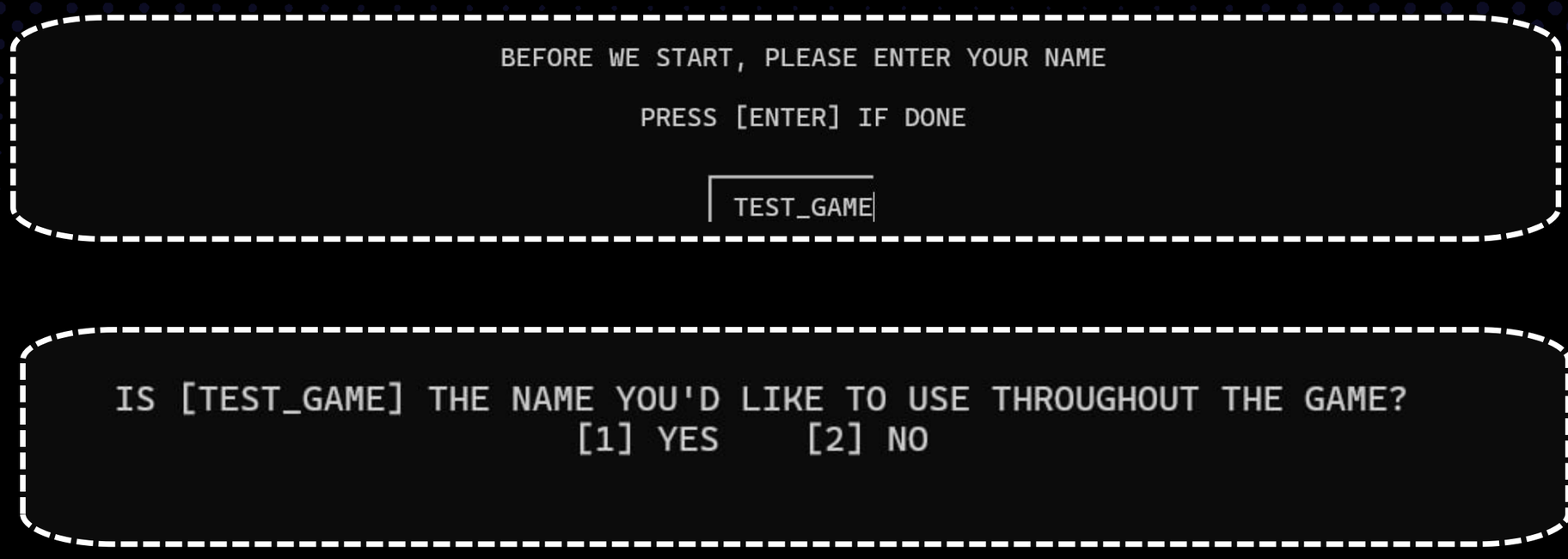
RANK	NAME	SCORE
[1]	JKHJ	17718
[2]	LHKLHJ	16500
[3]	LKJKL	00435
[4]	LKJAL	00249
[5]	KJHLKJ	00090

PRESS 1 TO GO BACK

PRESSING 3 WILL ALLOW YOU TO ACCESS THE LEADERBOARDS OF THE GAME. THIS GAME'S LEADERBOARDS ONLY SHOW THE TOP 5 HIGHEST POINTERS OF THE GAME. ALSO, IN THE CASE THAT THERE STILL NO HIGH SCORERS, A MESSAGE INDICATING NO RECORDS WILL BE SHOWN.

THE START OF THE GAME

AFTER PRESSING 1, YOU WILL BE ASKED TO RECONFIRM YOUR CHOICE TO START THE GAME. AND YOU WILL BE ASKED TO ENTER THE YOUR DESIRED NAME TO USE THROUGHOUT THE GAME.

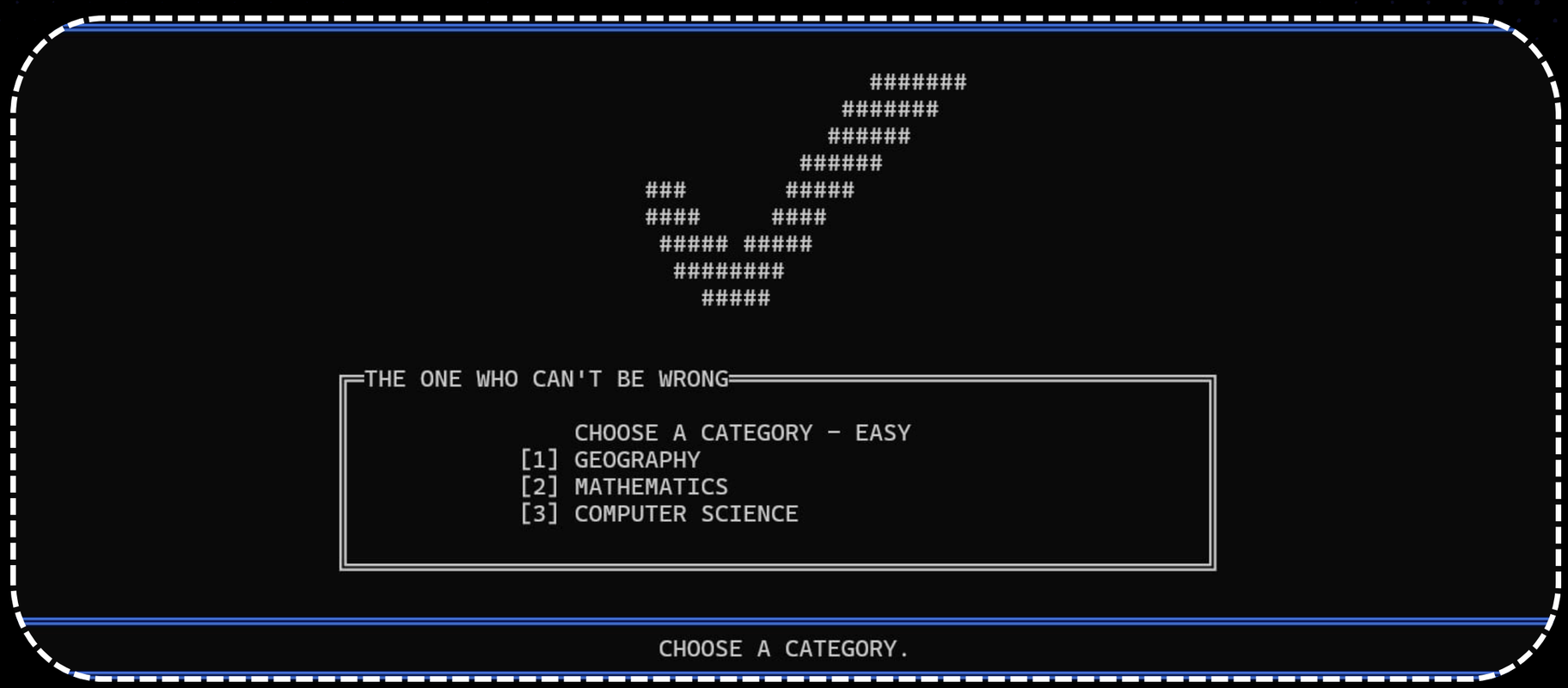


AFTERWARDS, A WELCOME SCREEN WILL BE SHOWN ON THE SCREEN, AND THE GAME MASTER WILL INTERACT WITH YOU AND GUIDE YOU THROUGH THE REST OF THE GAME. YOU JUST HAVE TO FOLLOW THE GAME MASTER'S DIRECTION TO EASILY NAVIGATE THROUGH THE GAME.

AFTER THE WELCOME REMARKS, YOU WILL AUTOMATICALLY BE DIRECTED TO THE GAME MECHANICS INTERFACE SHOWN IN THE ABOUT SCREEN.

GAMEPLAY OVERVIEW

AFTER FOLLOWING THE GUIDE OF THE GAME MASTER, YOU WILL BE THEN ASKED TO CHOOSE A CATEGORY. AS OF NOW, THERE ARE A TOTAL OF FIVE CATEGORIES. NAMELY, MATHEMATICS, SCIENCE, COMPUTER SCIENCE, GEOGRAPHY, AND HISTORY AND LITERATURE. AMONG THESE CATEGORIES, ONLY THREE PER ROUND IS GIVEN TO YOU AS AN OPTION. AND YOU SHALL ONLY PICK ONE.



NOTE: THE GAME STARTS AT AN EASY LEVEL AND YOU CAN LEVEL UP AS YOU PROGRESS AND GAIN POINTS BY ANSWERING EACH QUESTION CORRECTLY. THERE ARE THREE LEVELS FOR THIS GAME, THE EASY, AVERAGE, AND THE DIFFICULTY LEVEL.

- GATHERING A TOTAL OF 2000 POINTS IN THE EASY ROUND WILL ENABLE YOU TO PROCEED TO THE AVERAGE ROUND.
- GATHERING A TOTAL OF 6500 POINTS IN THE AVERAGE ROUND WILL ENABLE YOU TO PROCEED TO THE AVERAGE ROUND.
- FINALLY, YOU CAN WIN THE GAME AFTER GATHERING A GRAND TOTAL OF 17500 POINTS.

ALSO, TAKE NOTE THAT THE POINTS YOU GET PER QUESTION VARY. THE SCORING SYSTEM DEPENDS ON THE TIME CONSUMED TO ANSWER THE QUESTION. THE LESSER THE TIME CONSUMED, THE GREATER THE POINTS RECEIVED.

GAMEPLAY OVERVIEW

AFTER PICKING A CATEGORY, THE QUESTIONS WILL NOW BE PRESENTED TO YOU. AT THE START OF THE GAME YOU ARE GIVEN THREE LIVES, AND THREE POWER-UPS. NOTE THAT THERE ARE A TOTAL OF FIVE POWER-UP. HOWEVER, YOU ARE ONLY GIVEN THREE OF THEM THROUGHOUT THE GAME AND IT CAN ONLY BE USED ONCE.

YOU CAN ANSWER EACH QUESTION BY ENTERING THE LETTER OF YOUR CHOICE. EACH CORRECT ANSWER WILL MERIT YOU POINTS,

YOU WILL LOSE A LIFE WHEN:

- YOU ANSWER INCORRECTLY.
- YOU CONSUMED THE TIME WITHOUT ANSWERING.

THIS PART OF THE INTERFACE WILL ACT AS YOUR STATUS BAR. HERE YOU CAN SEE YOUR GAME INFORMATION AND GAME STANDING.

POWER-UPS

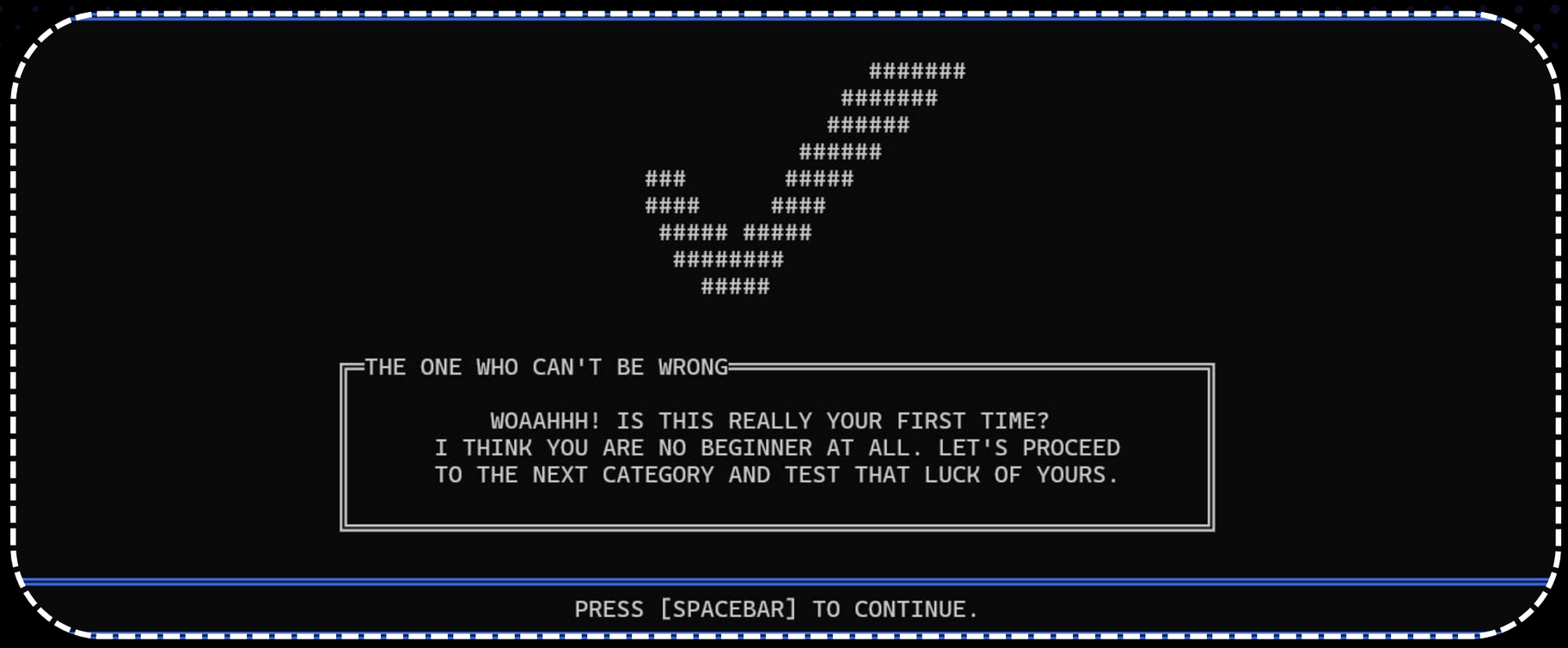
[1] PASS [2] IMMUNITY [3] DOUBLE JEOPARDY

AGAIN, THERE ARE FIVE POWER-UPS IN THIS GAME. HOWEVER, YOU WILL ONLY BE GIVEN THREE OF THEM PER GAME. THERE FUNCTIONS ARE AS FOLLOWS:

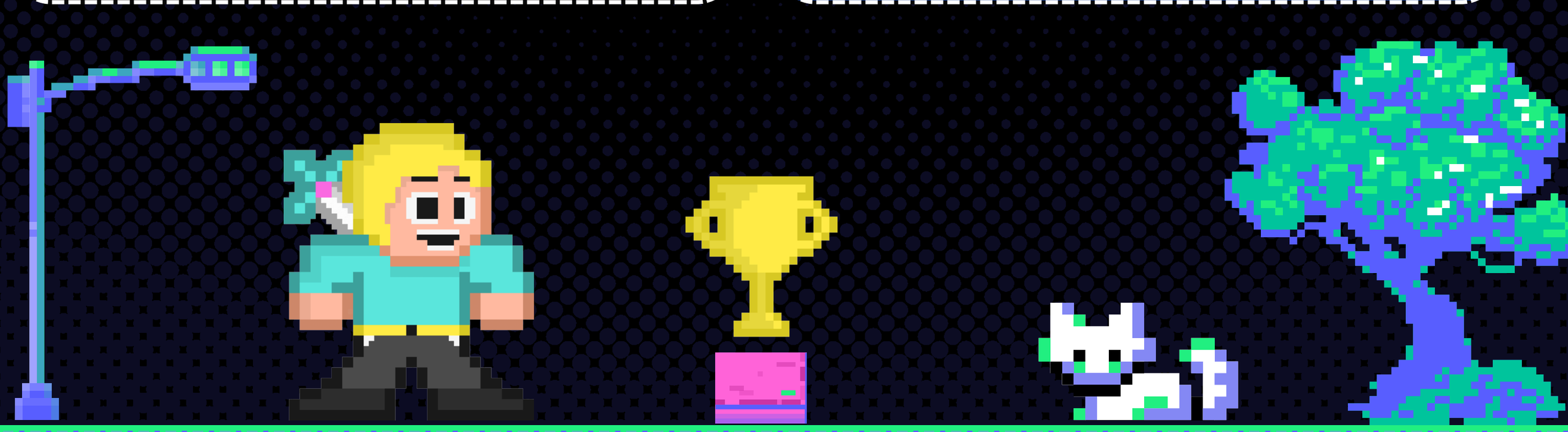
- TIME FREEZE - GIVES THE AUTHORITY TO FREEZE TIMER WHILE ANSWERING THE QUESTIONS.
- ERASER - THIS POWER-UP ELIMINATES ONE WRONG OPTION FROM THE CHOICES.
- DOUBLE JEOPARDY - THIS POWER-UP DOUBLES THE PLAYER'S SCORE IF THE QUESTION IS ANSWERED INCORRECTLY. HOWEVER,THE PLAYER WILL LOSE ALL THEIR POINTS IF THEY ANSWER INCORRECTLY.
- PASS - THIS POWER-UP WILL GIVE THE PLAYER THE AUTHORITY TO PASS AND ANSWER THE QUESTION LATER.
- IMMUNITY - THIS ALLOWS THE PLAYER TO ANSWER THE SAME QUESTION TWICE WITHOUT LOSING A LIFE IN THE CASE OF A WRONG ANSWER.

WINNING OR LOSING

IN THE CASE THAT YOU HAVE SUCCESSFULLY ACCOMPLISHED A LEVEL, A CONGRATULATORY INTERFACE WILL APPEAR ON YOUR SCREEN



AND AFTER GOING THROUGH ALL THE QUESTION. YOU WILL EITHER WIN OR LOSE. AND IF YOU WIN, THE WINNING SCREEN WILL BE PRESENTED. AND IF YOU LOSE, THE LOSING SCREEN WILL BE SHOWN.



CREDITS

J&J STUDIO AND COMPANY

PRODUCED BY:
JOHN LESTER N. ESCARLAN
MERLJHUN C. CATIQUISTA

OVERALL GAME STRUCTURE AND PROGRAM:
JOHN LESTER N. ESCARLAN
MERL JHUN C. CATIQUISTA

OVERALL GRAPHICS:
JOHN LESTER N. ESCARLAN

GAME CONTENT AND MECHANICS:
MERL JHUN C. CATIQUISTA
JOHN LESTER N. ESCARLAN

GAME MANUAL:
MERL JHUN C. CATIQUISTA