C.V. RAMAN GLOBAL UNIVERSITY

BHUBANESWAR, ODISHA, INDIA

Chrome Dinosaur Game – Detailed Report

C++(Group-8)

PROJECT GROUP-6

NAME	CRANES REGD.NO.
VINAY PRABHAKAR	CL2025010601870917
PRIYANSHU KUMAR BHADANI	CL2025010601886345
SUNNY KUMAR	CL202501060189431
KUNAL CHIRANIA	CL2025010601887456
VIVEK KUMAR	CL2025010601888264

UNDER THE SUPERVISION OF

SIKANDER SIR

C. V. RAMAN GLOBAL UNIVERSITY ,
BHUBANESWAR, ODISHA,
INDIA 2025-26

CONTENTS

INTRODUCTION

The Chrome Dinosaur game is a well-known browser-based offline mini-game. This project is a C-based terminal implementation of that game, using the Ncurses library to handle graphical rendering within a text-based interface. The game allows players to control a T-Rex, making it jump over obstacles while the game gets progressively harder.

This report provides a detailed breakdown of the code structure, game mechanics, features, compilation process, and potential enhancements.

CODE STRUCTURE & FILES

This project is structured using three main C source files and a README.md file for documentation.

2.1. main.c (Entry Point)

This is the main driver of the game. It:

- Initializes Ncurses, enabling terminal-based graphical display.
- Calls functions from other files (appearance.c and compute.c).
- Begins the game loop by calling startMenu().
- Cleans up and exits Ncurses when the game ends (endwin()).
- Key Functions in main.c:
- initscr(): Initializes Ncurses mode.
- start_color(): Enables color functionality.
- curs_set(FALSE): Hides the blinking cursor for a cleaner UI.
- startMenu(): Loads the game's start menu.

2.2. appearance.c (Graphics & Rendering)

This file handles all the visual elements of the game using Ncurses.

Sprites & Objects Rendered:

Object	Function Name	Description
Dinosaur (T-Rex)	`dinasour1()`, `dinasour2()`	Displays the dinosaur in two frames (for simple animation).
Cactus (Obstacle)	`cactus1()`, `cactus2()`	Draws two different cactus types in the game.
Sun & Moon	`sun()`, `moon()`	Displays a day/night cycle depending on score.
Game Over Screen	`showLoss()`	Displays a "Game Over" message when the player loses.
Title Screen	`showTrex()`	Displays the game's title screen.
Clearing Functions	`clearCactus1()`, `clearDinasourUp()`, `clearDinasourDown()`	Erases objects from the screen when needed.

Example of Dinosaur ASCII Art (dinasour1()):

```
void dinasour1(int y, int x) {
    mvprintw(y-4, x, "          e-e  ");
    mvprintw(y-3, x, "       /(\\_/)");
    mvprintw(y-2, x, ",___.--` /'-` ");
    mvprintw(y-1, x, " '-._, )/'");
    mvprintw(y, x, "     \\/");
}
```

- This function uses mvprintw(y, x, "text") to print ASCII art at a specific position.
- The dinosaur has two states (dinasour1 and dinasour2), allowing it to appear animated.

2.3. compute.c (Game Logic & Mechanics)

This file contains all the game logic required to run the game. It:

- Manages user input (jumping, shooting).
- Controls game speed (progressively increasing difficulty).
- Implements collision detection (checking if the dinosaur hits an obstacle).
- Handles score and high score tracking.

Key Functions in compute.c:

Function Name	Purpose
`startMenu()`	Displays the menu and collects player information.
`startEngine()`	Main game loop, controlling movement, scoring, and interactions.
`checkGame()`	Checks if the dinosaur collides with an obstacle (game over logic).
`computeTime()`	Adjusts game speed as the player progresses.
`computePrize()`	Determines if the player earns a projectile (prize system).
`showDinasour()`	Alternates between two dinosaur sprites for animation.
`endGame()`	Manages game-over behavior (restart or quit).

Collision Detection (checkGame())

```
int checkGame(int y, int x, int diY, int diX) {
    if (diY == y) {
        if (abs((diX+14)-x) <= 4) {
            return 0;  // Collision detected, game over
        }
    }
    return 1;  // No collision, continue game
}
```

- This function checks if the dinosaur is at the same Y position as the cactus.
- If their X positions are within 4 characters of each other, a collision occurs, and the game ends.

Speed Adjustment (computeTime())

```
int computeTime(int delayTime) {
    if (delayTime >= 250000) {
        delayTime -= 900;
    } else if (delayTime >= 200000) {
        delayTime -= 600;
    } else {
        delayTime -= 200;
    }
    return delayTime;
}
```

- The game speed increases over time, making it more challenging.
- This function reduces the delay between frames, causing obstacles to move faster.

GAMEPLAY FEATURES

3.1. Game Controls

Jump → Space
Fire Arrow → k (when prize is available)
Restart After Loss → r
Quit Game → q

3.2. Dynamic Difficulty

The game starts slow but speeds up as the player scores points.
Day & night alternates based on the score.

3.3. High Score System

The game stores the highest score in a file (highScore.txt).
If a player beats the previous high score, it gets updated.

3.4. Prize System

At certain score thresholds, the player earns a projectile.
This destroys obstacles, making it easier to survive.

COMPILATION & EXECUTION

4.1. Requirements

- **Install Ncurses library:**
  sudo apt-get install libncurses5-dev
- **Compile the code:**
  gcc main.c -lncurses -o main.out
- **Run the game:**
  ./main.out

POTENTIAL ENHANCEMENTS

While this game is fully functional, it can be improved in various ways:

5.1. Enhanced Graphics

- Replace ASCII art with Unicode characters for better visuals.
- Implement sprite scaling for different terminal sizes.

5.2. Sound Effects

- Integrate sound libraries (e.g., SDL2_mixer).
- Add jump, collision, and background music.

5.3. Additional Obstacles & Power-ups

- Introduce flying enemies (birds).
- Implement speed boosts or extra lives.

5.4. Multiplayer Mode

- Allow two players to compete in split-screen mode.

CONCLUSION

This C-based Chrome Dinosaur game effectively recreates the experience of the browser version using Ncurses. It includes features like dynamic difficulty, collision detection, a high score system, and basic animations. With further improvements, it could become an even more engaging terminal-based game.

REFERENCES

1. **Ncurses Library Documentation**
- GNU Ncurses: https://invisible-island.net/ncurses/
- **Ncurses Reference Guide:** https://tldp.org/HOWTO/NCURSES-Programming-HOWTO/
2. **C Programming Language**
- **C Standard Library Reference:** https://en.cppreference.com/w/c
- **The GNU C Library Manual:** https://www.gnu.org/software/libc/manual/
3. **Game Development in C**
- **Terminal Game Programming**: https://gamedev.net/tutorials/programming/general-and-gameplay-programming/creating-a-simple-terminal-game-in-c-r4515/
4. **Compilation & System Programming**
- **Linux System Programming:** https://www.advancedlinuxprogramming.com/
- **GCC Compilation Guide:** https://gcc.gnu.org/onlinedocs/