Traffic Racer Game – Project 9 Submission Document

Concept:
Traffic Racer is an engaging single-player game where players control a car, navigating through traffic, avoiding obstacles, and collecting **dollars** to achieve the highest possible score. Built as part of **Project 9 in the Nand2Tetris course**, this game demonstrates a seamless integration of low-level programming and creative design, offering players a dynamic and progressively challenging experience.

Architecture:

Main.jack:
Acts as the game's entry point, managing its high-level execution. It initializes the core game engine by creating an instance of TrafficRacerGame, starts the game loop using main(), and ensures proper resource cleanup. Afterward, it clears the screen and halts the system, guaranteeing a clean shutdown. This class primarily serves as a launcher and delegator, leaving the core game logic to other classes.

GameSetup.jack:
Prepares and visually sets up the game environment before gameplay begins. The constructor (new) draws the screen layout, including lane dividers, sidewalks, and tree decorations on both the left and right panels. It also displays instructions for the player, such as "Press space to start" and "Use left/right arrows to move". The method dispose() ensures proper cleanup of tree objects and memory deallocation. This class focuses on creating an aesthetically organized game interface and setting the stage for smooth gameplay.

Car.jack:
Handles player-controlled car functionality, specifically movement. Key actions include allowing directional movement (moveLeft, moveRight) and ensuring smooth control of the car across the game screen.

Barrel.jack:
Represents obstacles in the game. The key method startPointBarrel manages the initial configuration or placement of barrels on the game screen. Collision detection with the car is handled here, determining whether the player has hit a barrel and triggering the appropriate game-over event counter, which is then used in the TrafficRacerGame class.

Cone.jack:
Represents cone-shaped obstacles, with methods startPoint1 and startPoint2 defining the initial placement points for each cone. Collision detection with the car is managed here, ensuring proper interaction outcomes similar to those described in the Barrel.jack. Both cones and barrels share a downward movement method, with their speed increasing progressively as the game advances.

Dollar.jack:
Governs collectible dollar behavior and scoring logic. The method startPointDollar sets the

initial placement for dollar collectibles. Collision detection with the car is handled here, and successful collection triggers score updates via ScoreBoard.jack. The dollar feature adds an additional layer of motivation for players, encouraging them to progress further in the game.

ScoreBoard.jack:
Tracks and dynamically displays the player's score throughout the game. Additionally, it updates the high score during gameplay, preserving player progress and reinforcing their motivation to continue improving their performance.

Failed.jack:
Manages the game-over state and provides functionality for restarting the game. This class is invoked in TrafficRacerGame when the player collides with a cone or a barrel, triggering the game-over screen. From there, the player can choose to restart the game or exit, ensuring smooth continuity and user control.

Random.jack:
Introduces randomness into gameplay mechanics by generating varied positions for obstacles and collectibles. The key method betweenOneAndFour() randomizes the appearance of obstacles and dollars within their respective lanes, adding unpredictability to gameplay.

TrafficRacerGame.jack:
Serves as the central game engine, managing the primary game loop, synchronizing gameplay events, and handling visual updates. The boolean method run() integrates all classes into a cohesive gameplay experience, which is then instantiated and executed through Main.jack.

Tree1.jack & Tree2.jack:
These files utilize the bitmap editor to create static or animated tree visuals, enhancing the game's aesthetic appeal and providing decorative environmental elements.

Motivation:

We chose to create **Traffic Racer** because it allowed us to combine our passion for game design and programming into a dynamic and exciting project. This game not only reflects the technical concepts we've learned throughout the **Nand2Tetris course**, but also challenges players with progressively difficult gameplay mechanics. The increasing speed of obstacles keeps players engaged, pushing them to improve their reflexes and strategies.

Google Drive link to your video:

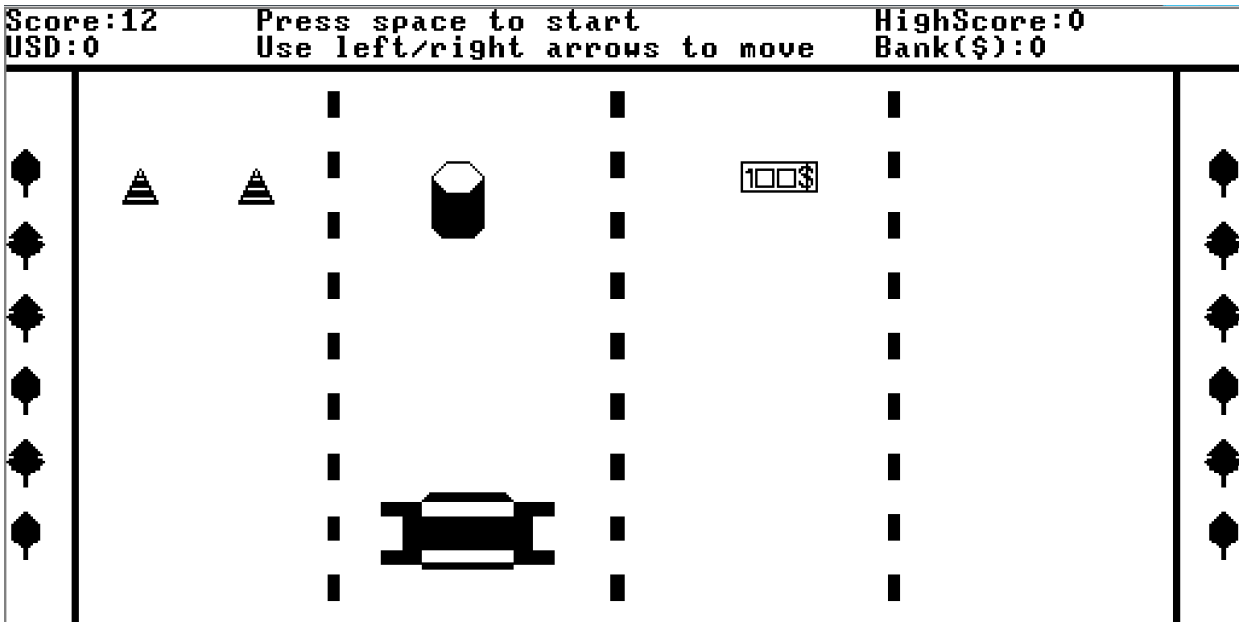
https://drive.google.com/file/d/1jfF3s7Ixz-eheqHtRInSOSUI5TwAaScC/view?usp=sharing

Names and Emails:

By Yotam Markman and Amit Feldman

Mail : yotam.markman@post.runi.ac.il , amit.feldman@post.runi.ac.il

ID : 322632266, 319086054



GAME OVER

Your Score is : 57

Your Balance in $ is : 100

Press R to Restart, Q to Quit

Made by Amit Feldman and Yotam Markman