Vrije
Universiteit
Brussel

Computersystems

Assembly Project Tron: Report

De Groot Jan & Radmal Khaled

Table of Contents

1 Description

Tron is a 2 player game where the players each control a character. Each character is a point on the screen that leaves a trail in the player's colour. This trail, as well as the walls, can be considered obstructions. The player can win by navigating his point in such a way that the other player is forced to hit an obstruction.

The difficulty of the game can be changed by placing random obstructions on the playing field. Another way to customise the difficulty is by adjusting the player's movement speed.

An extra feature to this game is the introduction of jump points. With a jump point the player can leap over his own or his opponent's trail. However, by using the jump function, one point will be subtracted from the player's score.

A screenshot of the game can be seen in Figure 1: Tron.

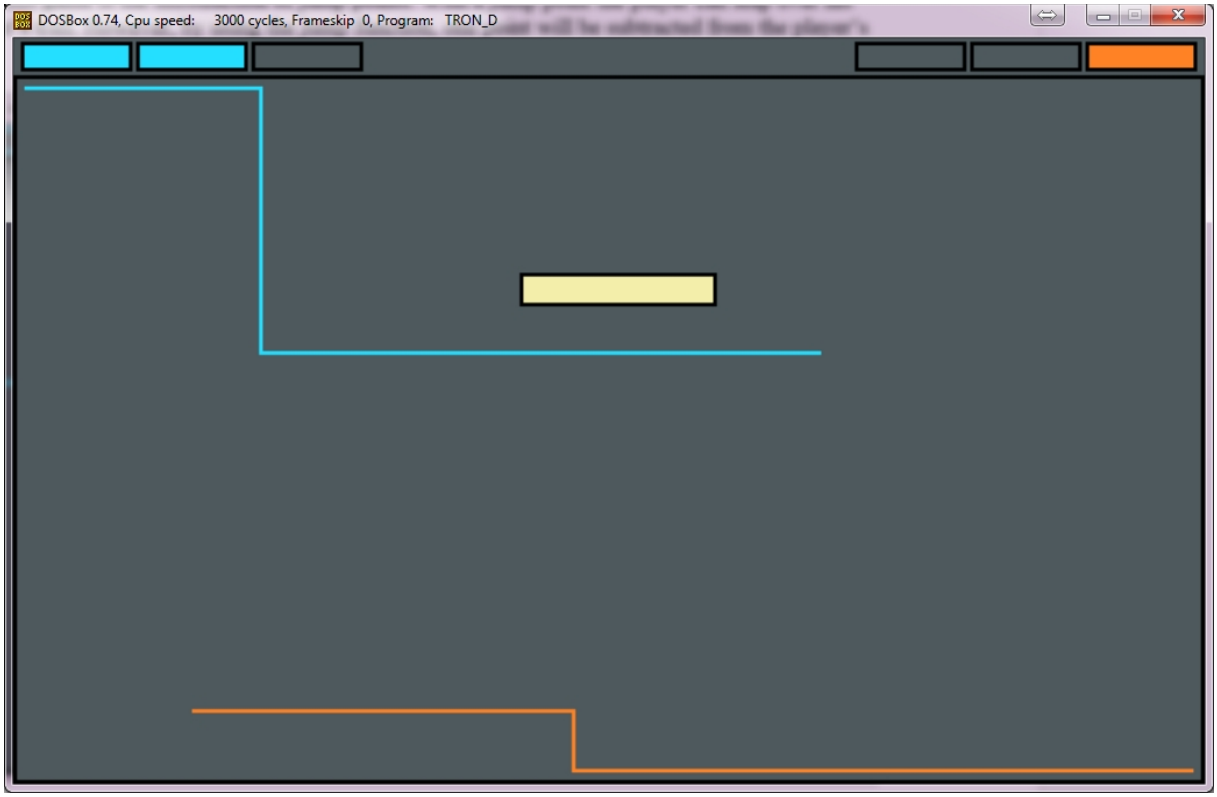


Figure 1: Tron

2 Features / Commands / Functionalities

2.1 In-game

During playtime no customization is possible. Instead this section will describe the actions that the player can take and the effects of these actions. It will also describe the functionalities this game offers in the standard configuration. For more information about customizing the game see 2.2 Extra configuration settings.

When a new match is started the game will wait for player input. Once the player hits the start button the characters start moving. If, at any point during the game, the players want to quit they can press escape and the game will halt immediately.
For temporary interruptions to the game the pause button can be pressed.

Each player can move his character up, down, left or right. The player cannot invert his direction immediately because this would result in an inevitable crash of the player with his own trail.
Apart from basic spatial movements, the player can also jump. By jumping, the player moves into the third dimension, thus disappearing from the screen. In this third dimension the player can still move freely and will be exempt from hitting trails. As a drawback to jumping the player loses one score point.

The score is displayed in the upper left and right corners of the screen. Each player has a scoreboard in the same colour as his character.
When a player gets the maximum score a victory message is displayed indicating which player won.

These features can bee seen in Figure 2: Tron functions.

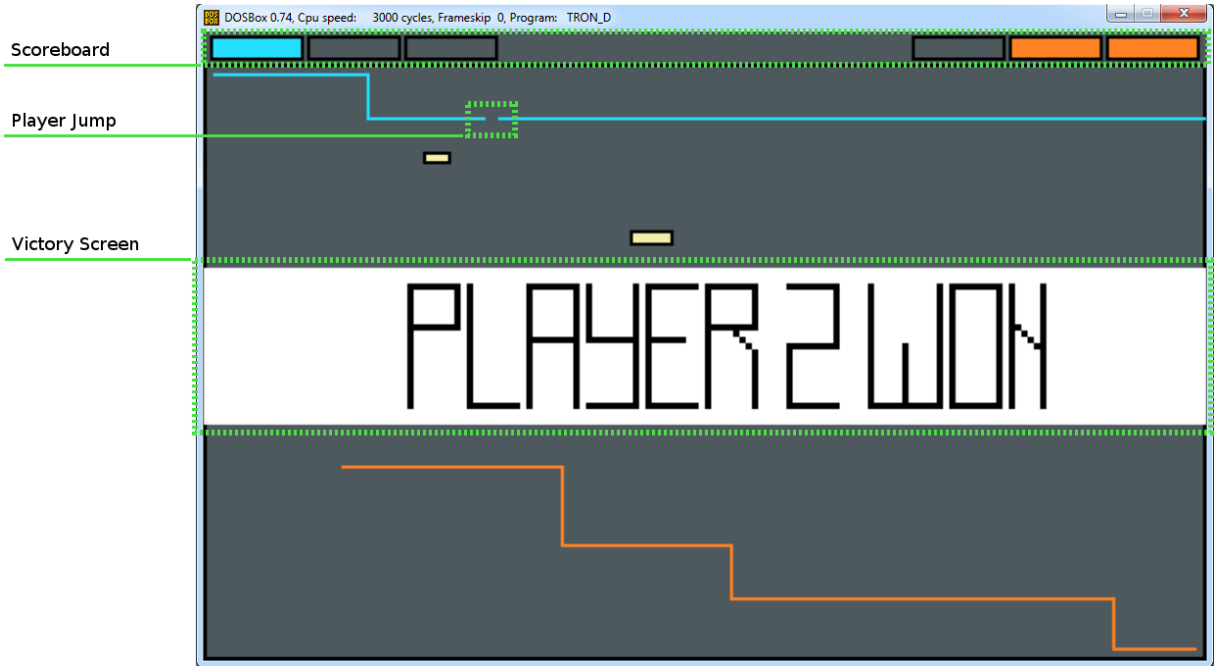


Figure 2: Tron functions

2.2 Extra configuration settings

The following list contains all the editable setting grouped per category. These settings can be found in the Settings.ASM file. Once a setting has been modified the game must be rebuilt.
If a setting in the Settings.ASM file is not mentioned in this list, it should not be modified.

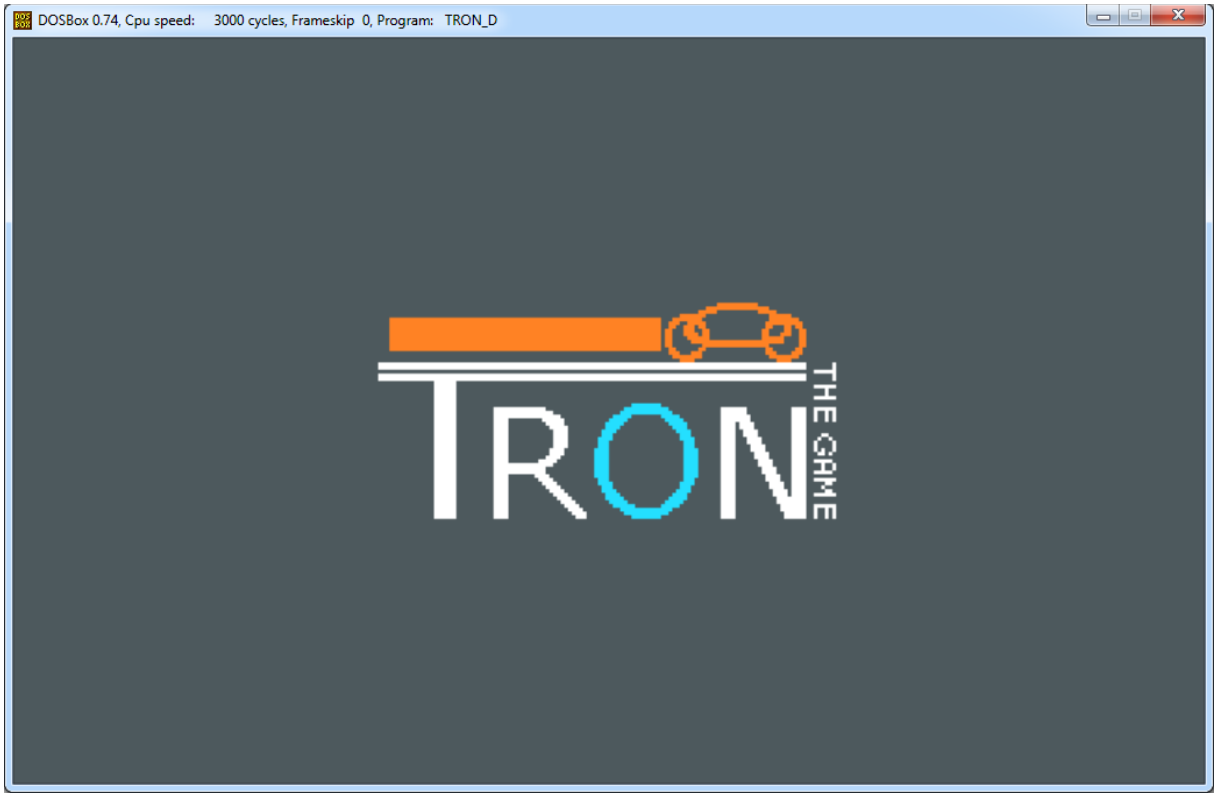
Game settings		
Name	Description	Default
matchNumberHandicap	The match number the game starts to count from.	0
player1ScoreHandicap	The number of score points player 1 starts the game with. This should always be lower than maxWin.	0
player2ScoreHandicap	The number of score points player 2 starts the game with. This should always be lower than maxWin.	0
player1JumpHandicap	The number of jump points player 1 starts the game with. This should never exceed the score handicap.	0
Player2JumpHandicap	The number of jump points player 2 starts the game with. This should never exceed the score handicap.	0
maxWin	When a player reaches this many score points he wins the game.	3
maxMatch	Once this many matches have passed the game is ended. To remove the limit this value can be set to 0.	0
minObstacles	The minimum amount of obstacles to appear on the screen.	1
maxObstacles	The maximum amount of obstacles to appear on the screen.	3
maxHeightObstacle	The maximum height of one obstacle in pixels.	20
maxWidthObstacle	The maximum width of one obstacle in pixels.	80
playerJumpTime	How far the player jumps. This is a distance in pixels.	5
Graphical settings		
Name	Description	Default
backgroundColour	The colour index for the background. This is a colour specified in palette.	0
player1Colour	The colour index for player 1. This is a colour specified in palette.	2
player2Colour	The colour index for player 2. This is a colour specified in palette.	3
obstacleColourBody	The colour for the filling of obstacles. This is a colour specified in palette.	4
colourEdge	The colour for all the edges in the games. This includes the edges of the score containers, the edge of the playing field and the edge of obstacles. This is a colour specified in palette.	5
palette	To define a new colour: - Choose your desired colour in an editor like GIMP. - Get the RGB values from 0→255. - Take the R value and divide it by 255. Now multiply it by 63. - Take the G value and divide it by 255. Now multiply it by 63. - Take the B value and divide it by 255. Now multiply it by 63. - Insert these values correspondingly into the colour you want to replace. - Make sure that you insert integer numbers only (0->63).	19, 22, 23, 63, 63, 63, 09, 55, 63, 63, 32, 09, 60, 59, 42, 00, 00, 00

screenW	The width of the screen in pixels.	320
screenH	The height of the screen in pixels.	200
Timing settings		
Name	Description	Default
splashScreenDelay	The amount of time the splash screen is shown. By using a multiple of ticksPerSecond it is possible to “accurately” determine how many seconds it should be displayed.	ticksPerSecond * 3
victoryDelay	The amount of time the victory screen is shown. By using a multiple of ticksPerSecond it is possible to “accurately” determine how many seconds it should be displayed.	ticksPerSecond * 2
playerSpeed	The speed at which the player moves. 1 being the highest speed possible.	1
Input settings		
Name	Description	Default
player1Up	The key that makes player 1 move in the up direction.	key_w
player1Down	The key that makes player 1 move in the down direction.	key_s
player1Left	The key that makes player 1 move in the left direction.	key_a
player1Right	The key that makes player 1 move in the right direction.	key_d
player1Special	The key that makes player 1 jump.	key_space
player2Up	The key that makes player 2 move in the up direction.	key_up
player2Down	The key that makes player 2 move in the down direction.	key_down
player2Left	The key that makes player 2 move in the left direction.	key_left
player2Right	The key that makes player 2 move in the right direction.	key_right
player2Special	The key that makes player 2 jump.	key_l
gameStart	The key to start the match.	key_enter
gameQuit	The key to quit the game.	key_escape
gamePause	The key to pause the match.	key_p

3 High level and abstract design

3.1 Program flow

The game can be divided into four logical sections. The first, and simplest, of these is the boot section. Its only function is to display a splash screen that lets the user know he's about to play the most awesome game in his life. An image of this splash screen can be seen in Figure 3: Splash screen.

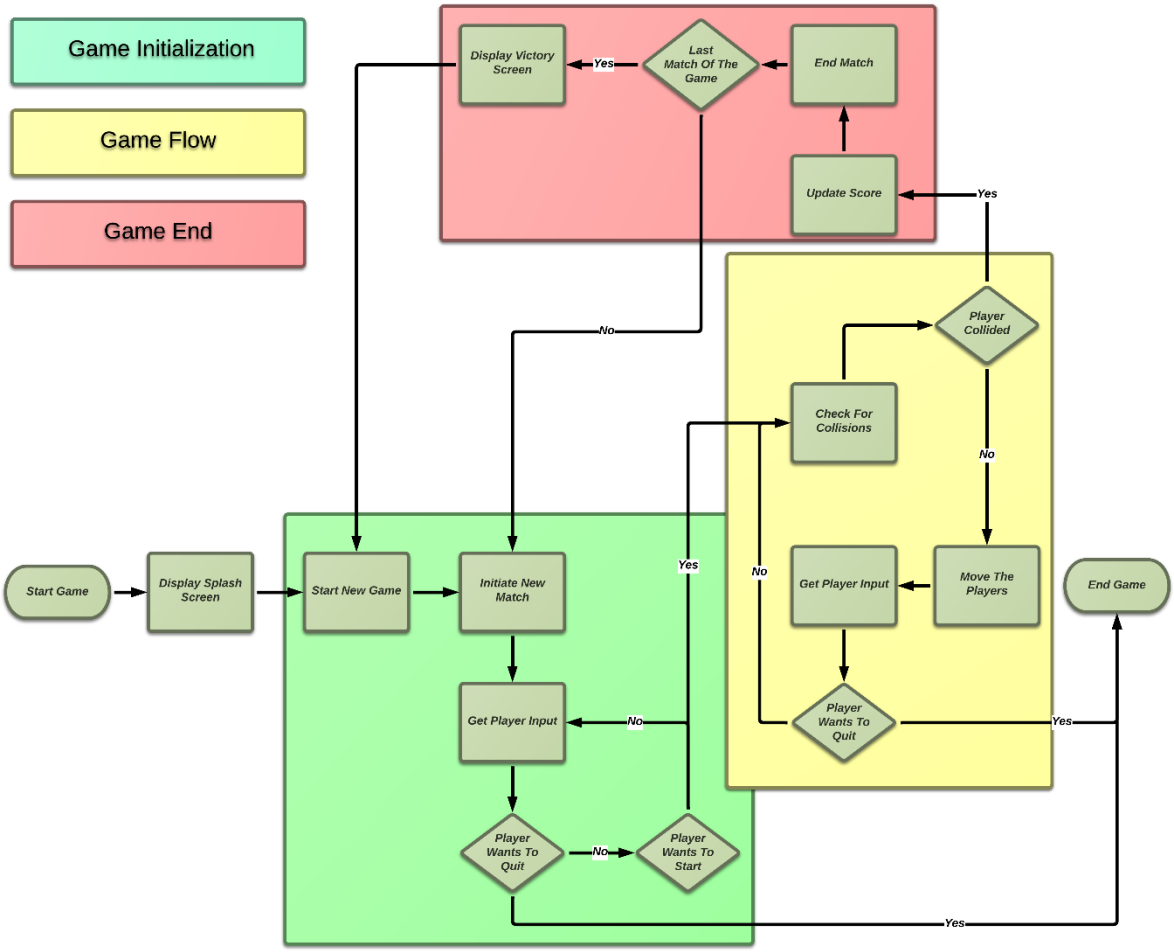


Following the boot section is the game initialization section. This handles the setting up of the initial game parameters. It generates the random objects, updates the screen to reflect the starting state and then waits for input. If the user presses start, the control is passed on to the main game flow.

During the main game flow the player's character is moved according to the direction chosen by the user. Every time the character is moved, the game checks to see if any collisions have occurred. It also checks to see if a player is currently jumping, because in that case, the player can only hit edges.

The last section is the game ending logic. In this logic block the players' score is updated and the match ends. It then goes on to check if this was the last match in the current game. Depending on the case, it will either start a new match or display a victory screen and go on to start a new game.

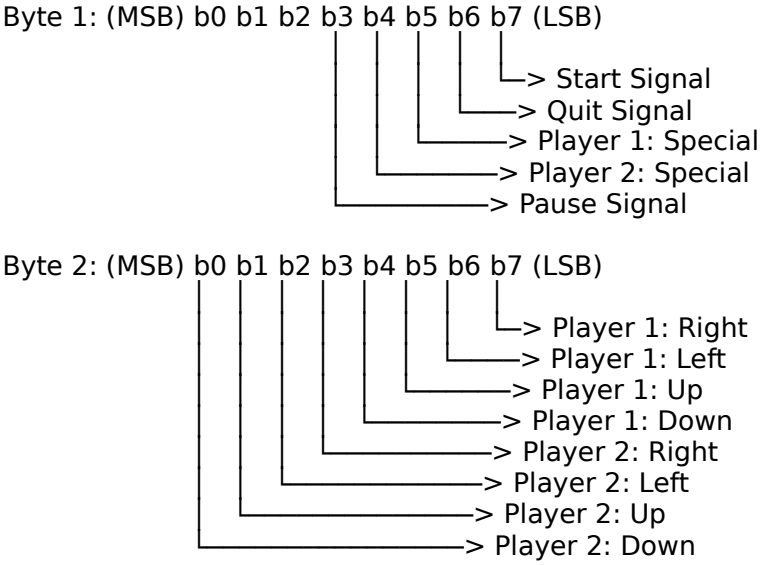
A schematic overview of this program flow can be found in Figure 4: Flowchart.


Game Initialization
Game Flow
Game End
Start Game
Display Splash Screen
Start New Game
Initiate New Match
Get Player Input
Player Wants To Quit
Player Wants To Start
Yes
No
Check For Collisions
Player Collided
Move The Players
Get Player Input
Player Wants To Quit
Yes
No
Update Score
End Match
Last Match Of The Game
Yes
Display Victory Screen
End Game
Yes

3.2 Functional description

Each assembly file has a particular set of functions assigned to it. This way the program can maintain a neat and logical structure. The game is made up out of 8 files: CORE.ASM, GUI.ASM, INPUT.ASM, MISC.ASM, SETTINGS.ASM, VAR.ASM, VIDEO.ASM and TRON.ASM. The following will provide an overview of functions assigned to each file.

INPUT.ASM:
This file checks if the user has pressed a key. It then converts this keypress to a 16-bit value representing its function with the following specifications:



VIDEO.ASM:
In VIDEO.ASM all interactions with the screen are handled. It is responsible for initializing the graphical mode and restoring it back to normal. It also clears the screen and draws colours on it. Its last functionality is reporting the colour of a particular pixel.

GUI.ASM:
All functions regarding graphical feedback are handled in GUI.ASM. It displays intermediary screens as well as the main game. If there's anything being displayed on the screen, this file gave shape to it.

CORE.ASM:
This is the heart of the game. It contains all the individual game sections. It handles starting, ending and playing the game. However, it does not contain the glue to tie all the pieces of the game together.

TRON.ASM:
In TRON.ASM all the game logic gets fit together. It makes all the pieces tick and work in harmony.

MISC.ASM:
MISC.ASM contains all macros and functions that have no real place in other files. This is usually when a function is accessed from more than one place and has a generic function.

VAR.ASM:
The global variables are kept in VAR.ASM. This way all files that need to access them can do so without much hassle.

SETTINGS.ASM:
User editable (and non-editable) settings are kept here. If you want to modify the game in any way, this is the place to be.

4 Used applications

DosBOX	Emulation program for the DOS environment run on an 8086 (micro)processor.
MASM	Converts assembly files into object files.
Link	Convert and concatenate object files into executable DOS files.
Notepad++	Text editor for writing code.
GIMP 2	To make fancy images.
Libreoffice Writer	To bodge this awesome report together.
http://www.lucidchart.com	Making of the flow charts.