Mr. Giansante

C++ Programming

Tron Light Cycles

August 2018

Tron Light Cycles

Design a C++ program that simulates the **Light Cycles** portion of the arcade game **Tron**.

Tron was an 80's arcade game based on the 1982 Disney movie by the same name. Disney is planning on releasing Tron Legacy in December 2010.

In the original movie, the MCP (Master Control Program) wanted to control the entire system for evil purposes. User Programs (the good guys) were forced to compete in games. Programs that lost were deresed (deresolutioned).

In one of the games, devices resembling motor cycles were used to race on a grid. As they crossed the grid, they left a trail (Jet Walls) where they had been. These trails are like walls. If a Program on a Light Cycle ran into a wall, he crashed and was deresed.

Two Light Cycles leave the playing field from opposite sides, heading for each other. One Light Cycle leaves a blue trail, and the other a gold (or yellow) trail. The goal of the game is to stay alive longer than the other player. Players die by running off the edge of the playing field, or running into the trails left by their opponent's Light Cycle or their own Light Cycle.

A running score should be kept and displayed.

Red Player should use: [Z] [X]
Blue Player should use: [<] [>]

Variations to Game

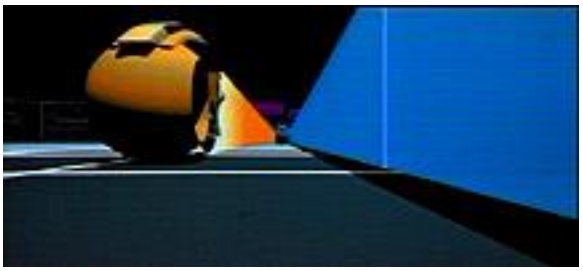
Two Key Controls
Players use only two keys to control lightcycle. One key turns them to the left and one to the right (Note: left and right will depend on the direction they are going). This requires a little bit more code, but makes the game easier and more exciting.

Option to turn on "obstacles"
This can be accomplished with only a few lines of code since it only involves drawing rectangles (or circles) onto the PictureBox.

Four-player game
This does not require much more code (mostly cut-and-paste). In the four player game, the game should continue until only one player is left. In other words, the game should not end as soon as the first player dies. When a player dies, their lightcycle should no longer move, but their trail should remain.

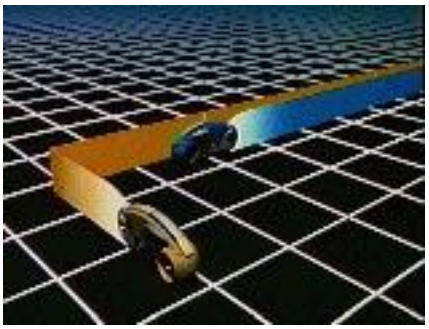
Tron Light Cycles

The following are screenshots from the Light Cycle scene of the movie Tron ...



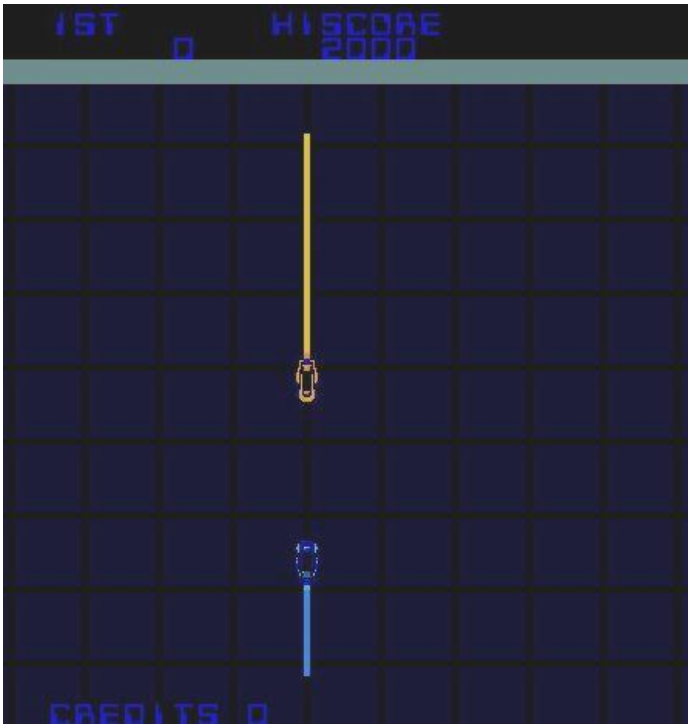
Yellow Light Cycle zips beside the trail of the blue light cycle.

Two Light Cycles collide.



Yellow Light Cycle tries to cut off blue light cycle.

The following is a screenshot from the actual arcade game Tron ...



Tron Light Cycles

Libraries to Include

You will need to include the following libraries:

```
#include <winbgim.h>
#include <conio.h>
#include <windows.h> // for Sleep(milliseconds) function
```

Variables

You will need the following variables:

```
// Global Variable Declarations
int red_x, red_y, red_dir;
int blue_x, blue_y, blue_dir;
bool red_alive, blue_alive;
bool gameover;

const int up = 1;
const int right = 2;
const int down = 3;
const int left = 4;
```

Note: The constants are used to make the code more readable.

Capturing Input from the Keyboard

The problem with using `getch()` is that it waits for a key to be pressed before if continues executing the program code. In the case of Tron Light Cycles, we want the dots to keep moving regardless of whether or not the players press keys.

Therefore, we use the `kbhit()` function. It returns true or false depending on if a key was pressed or not. If no key was pressed, we skip the `getch()` function.

```
char choice;

if (kbhit())
{
     choice = getch();
}
```

Note: the value stored in the variable choice will be the ASCII code for the letter key pressed. An ASCII Code table is available on my website.

Tron Light Cycles

The newgame() Function

```
void newgame()
{

     cleardevice();

     outtextxy(5, 5, "Tron Light Cycles ~ Red [Z][X] ~ Blue [<][>] ~ Quit [Q]");

     setcolor(15);
     rectangle(0,19,399,399);

     gameover = false;

     red_x = 2;
     red_y = 200;
     red_dir = right;
     red_alive = true;

     blue_x = 398;
     blue_y = 200;
     blue_dir = left;
     blue_alive = true;

     putpixel(red_x, red_y, 4); // Red Player
     putpixel(blue_x, blue_y, 1); // Blue Player
}
```

Changing the Player's Directions

This example uses only two keys to control the player. 'Z' turns the red player left and 'X' turns the red player right.

The directions circle clockwise from up (1) to right (2) to down (3) to left (4).

To turn a player left, increase their direction variable by 1. If it reaches 5, set it back to 1.

To turn a player right, decrease their direction variable by 1. If it reaches 0, set it back to 4.

```
     if(choice == 90 || choice == 122) // User chose 'z' or 'Z' ie. Red Player Turn Left
     {
          red_dir = red_dir - 1;
          if(red_dir == 0){ red_dir = 4;}
     }

     if(choice == 88 || choice == 120) // User chose 'x' or 'X' ie. Red Player Turn Right
     {
          red_dir = red_dir + 1;
          if(red_dir == 5){ red_dir = 1;}
     }
```

Tron Light Cycles

Updating the Player's x and y coordinates

Depending on the direction variable, update the player's position variables.

```
if(red_dir == left){red_x--;}
if(red_dir == right){red_x++;}
if(red_dir == up){red_y--;}
if(red_dir == down){red_y++;}
```

Check if a Player has Lost

Check if the player has run off the screen, and if so, set their alive variable to false.

```
if(red_x < 1){red_alive = false;}
if(red_x > 398){red_alive = false;}
if(red_y < 20){red_alive = false;}
if(red_y > 398){red_alive = false;}
```

Check if the player has run over a track (ie. the spot they are moving to is not black).

```
if(getpixel(red_x,red_y) != 0){red_alive = false;}
```

Check if the players have hit head-on.

```
if((red_x == blue_x) && (red_y == blue_y)){red_alive = false; blue_alive = false;}
```

Display if there is a Winner

Remember the case where both players die simultaneously and it is a tie.

```
if((red_alive == false) && (blue_alive == true))
{
     outtextxy(150,190,"Blue Player Wins!");
     outtextxy(125,210,"Press [Y] to play again.");
     gameover = true;
}
```

Draw the player on the screen

```
putpixel(red_x, red_y, 4); // Red Player
```

Regulate the Speed of the Game

Pause execution of the program (in this case for 50 milliseconds) so the game is not too fast.

```
Sleep(50);
```

Repeat the Entire Code

All the code above must be placed in a loop so that it executes over and over again.

Tron Light Cycles

Learning Outcomes

Ontario

Programming Concepts and Skills
A1. demonstrate the ability to use different data types, including one-dimensional arrays, in computer programs
A2. demonstrate the ability to use control structures and simple algorithms in computer programs
A3. demonstrate the ability to use subprograms within computer programs
A4. use proper code maintenance techniques and conventions when creating computer programs

Software Development
B1. use a variety of problem-solving strategies to solve different types of problems independently and as part of a team
B2. design software solutions to meet a variety of challenges
B3. design algorithms according to specifications
B4. apply a software development life-cycle model to a software development project

Please fill out all the information in this column using a pen before getting this assignment marked.

_____
Name

_____ _____
Date Class

Academic Honesty

The work I am submitting is completely my own creation and has not been copied from anyone else's work. If I have received help on this project, the names of those who have assisted are listed below.

_____
Signature

_____

_____

Pre-Marking

The following people have pre-marked this assignment: (minimum of two)

_____

_____

User Interface

- ❑ Appropriate Title, Author Information, etc.
- ❑ Efficient and Esthetically-Pleasing User Interface
- ❑ Spelling and Grammar are correct

Code / Programming Style

- ❑ Variables are declared and logically named
- ❑ Code is commented where appropriate
- ❑ Code is indented and spaced to show stucture
- ❑ Code is efficient

Work Ethic / Problem Solving Skills

- ❑ Problem Solving Skills are demonstrated
- ❑ Makes Productive Use of Time
- ❑ Only Seeks Help when Necessary

Program-Specific Criteria

- ❑ Game starts with all players displayed on screen
- ❑ Players move automatically
- ❑ Players controlled using keyboard (2 or 4 keys)
- ❑ Trail is left as player moves
- ❑ Running off the screen results in player losing
- ❑ Running into a trail results in player losing
- ❑ Players cannot go through trail
- ❑ Head-to-head crash results in a tie
- ❑ Winner is displayed
- ❑ Screen is cleared after each game
- ❑ Players start new game in proper position
- ❑ Score is kept track of

Teacher Comments

_____

_____

Level 1 - Limited ability to meet standard and limited effectiveness
Level 2 - Some ability demonstrated and moderately effective
Level 3 - Considerable ability demonstrated, considerable clarity or accuracy
Level 4 - Thorough, high degree of skill demonstrated, insightful, highly accurate

Level