Tron Spring 2017: Project Outline (1 of 3)

1) The game takes place on an $n \times n$ grid.

2) In each game, there are 4 teams. Each team is controlled by one student's code or by the instructor's code.

3) Each team starts with a single Lightcycle in a single location.

4) Each turn, each cycle must move in one of the cardinal directions.

5) Every place where a cycle moves it leaves a trail.

6) If a cycle moves into a space with a trail it explodes.

7) If a cycle does not move it explodes.

8) If a cycle moves into the grid perimeter, it explodes.

9) If a cycle moves into the same space as another cycle, the one with the greater average CPU time explodes.
10) Twice during the game, a team may create a new cycle at any location along its path.

11) When a cycle explodes, it and its trail are instantly removed from the game.

12) A cycle's trail is the set of spaces from its most recent move back to the location of the first branch point reached on the backward trace. If no branch point is reached, then the trail extends back to the cycle's starting point.

13) Each turn, the MCP tells each player the current location of all cycles.

14) Each turn, each team tells the MCP the location of any new cycles (up to 2 per game) and the direction each cycle moves.

15) If a team fails to give its move instructions before the timeout, then all cycles of that team explode.

16) Timeout for the first turn is 5 seconds.

17) Timeout for each other turn is 0.1 seconds average (not counting turn 1).
MCP / Player Interface

1) Each player has 3 "public" functions: _init(), _move(), and _getName()

2) At the start of a new game, the MasterControlProgram (MCP) calls each team's _init() function giving it the game configuration:
   a) Each team's name and color.
   b) Each team's initial cycle coordinates.

3) Each timestep, the MCP will call team's _move()
   ■ *Telling* each team the current location of all cycles.
   ■ *Telling* each team the average CPU time of each team.
   ■ *Asking* each player if and where to spawn a new cycle and the direction each of its cycles wants to move.

Spring 2017: Game Rules

1) Only the MCP may call a player's _init(), _move(), or _getName() functions.

2) Any player that causes the program to exit with a runtime error is disqualified.

3) Any invalid moves cause the cycle to explode.

4) Player code that can trick the MCP without causing the program to crash might be able to give itself an advantage.
**Grading Rubric**

[+20 Points]: Your AI wins 1 of 1 verses 3 RandomWalk bots.

[+20 Points]: Your AI wins 2 of 3 verses 2 RandomWalk bots and a LineBot.

[+20 Points]: Your AI wins 2 of 3 verses 2 RandomWalk bots and a LineBot_BranchNearDeath.

[+20 Points]: Your AI wins 2 of 3 verses 3 LineBot_BranchNearDeath.

[+20 Points]: Your AI wins 2 of 3 verses 2 RandomWalk bots and a NerfedAttackBot.

---

**AFF Spring 2017: Extra Credit**

+10 Points: Tournament Finalist.

+25 Points: Bronze tournament Winner.

+50 Points: Silver tournament Winner.

+100 Points: Gold tournament Winner.

---

**Tourney**
Thursday May 4
12:30 - 1:45
CEC Basement Lab
Snacks will be served
Interface between MCP and AI

Each student must implement the three, non-static ("public") functions:

```c
char* firstname.lastname_getName()
void firstname.lastname_init(struct InitData *data)
void firstname.lastname_move(struct MoveData *data)
```

**Important:** Do not save the addresses:

```c
struct InitData *data
struct MoveData *data
```

The memory will be freed after your function returns. Any *values* you want to save must be copied **value-by-value** to **static** local or **static** file-scope variables or structures.

---

tron.h: _getName

```c
char* firstname.lastname_getName()
```

Returns a pointer to a character array containing your player name.

The returned name may contain any non-whitespace, printable ASCII character (base-10 codes 33 through 126).

If more than one student in the class returns a particular name, the MCP will force each to be unique by appending numbers.

A player's _getName() function must always return the same name.

When _getName() returns, the MCP will copy the name into its own storage and never again use the returned pointer.

No more than **MAX PLAYER NAME LEN** characters (not including the terminating '\0') will be copied from the returned pointer.

Name must be NULL-terminate.
Compiling Tron

1) Place your AI code in the file: 
   yourFirstName_yourLastName.c

2) In a clean directory with your source code, copy from the class website the files tron.h, libmcp.a (the MCP library), and classDummy.c.

3) Edit classDummy.c by commenting out each _getName(), _init() and _move() method with your name.

4) Compile all .c files in the directory and link with the static MCP library, libmcp.a:
   
gcc *.c -L -lmcp -lm -lpthread `sdl-config --libs` -lSDL_ttf -o tron

This compiles all .c files in the directory, links with libmcp.a and creates a stand-alone executable with the name tron.

Running the MCP with your AI

- After you have compiled your code and linked with the MCP, run by entering the command:
  
  ./tron

- With no arguments, tron will display a usage screen telling you what arguments to use to specify the, an optional random number seed, which players are to compete, etc.

- When tron runs successfully with correct arguments, it will produce a file that contains all moves of the game: movelist.txt

- Running tron with a graphics display will show a realtime SDL (Simple DirectMedia Layer) window.
Representing Direction

#define NORTH 1
#define EAST 2
#define SOUTH 4
#define WEST 8

given in tron.h

By using powers of 2, one integer variable can represent more than one direction. For example:

static int getDirBits(int x, int y)
{
    int dirBits = 15;  //All 4 flags on: =N|E|S|W
    if (grid[x][y-1] != WHITE) dirBits -= NORTH;
    if (grid[x][y+1] != WHITE) dirBits -= SOUTH;
    if (grid[x-1][y] != WHITE) dirBits -= EAST;
    if (grid[x+1][y] != WHITE) dirBits -= WEST;
    return dirBits;
}

Why not use if, else if, else?

Use Many Small Helper Functions

It is often useful to know the number of choices into which a particular cell may.

static int getNumberOfOpenDirections(int dirBits)
{
    int openCount = 0;
    if (dirBits & UP) openCount++;
    if (dirBits & DOWN) openCount++;
    if (dirBits & LEFT) openCount++;
    if (dirBits & RIGHT) openCount++;
    return openCount;
}
**Unit Test** your Helper Functions

```java
static void test_getNumberOfOpenDirections()
{
    // 1) Clear the grid.
    // 2) Set, using a short series of assignment statements,
        // a **few** specific walls and player colored cells
        // showing in a small area of the grid.
    // 3) Call `getNumberOfOpenDirections()` on the
        // various cases and print results.
    // 4) Verify your results with hand drawing on graph paper.
}
```

---

**Test Small!**

- When developing your AI, change the grid size to something you can print out (maybe 20x20) and actually read.
- After you have the bugs worked out on a small size, THEN try the full size.
- When you find a bug on a full size grid, go back to a small size and try to reproduce it.
C keyword: static

- In C, a variable declared as static in a function is initialized once, and retains its value between function calls.
- The default initial value of an uninitialized static variable is zero.
- If a function or variable is declared static, it can only be accessed in that file.

Use static to protect your variables and function calls from other teams.

Suggested Structure for Grid

Create a file scope grid that includes the boundary wall.

Your file scope variables will not be visible to the Master Control Program nor to other student's code.

Shown is a 7x7 grid which includes the boundary wall cells shown in gray.

Moving walls are only on grid sizes ≥ 20×20.
Suggested Data Structure for Cycle Trails

```c
static int grid[MAX_GRID_SIZE][MAX_GRID_SIZE];
```

File scope sized for the biggest possible grid.

- Cycle path must persist between calls to `_move`.
- The locations of the boundary
- Empty cells must be identifiable.
- Cells containing lightcycle trails must be identifiable.
- When a cycle is terminated, there must be a way of removing its path back to the correct branch without disturbing other lightcycle trails.

How will you _represent_ , _access_ and _update_ this information?

---

Strategies

- Deep look-ahead.
- Attack.
- Run away.
- Team Formations.
- Trap Building.
- Maximizing open space.
- Fill an area while leaving an escape route.
Looking Ahead

In the Tron game, each cycle as at most 3 choices of where to move: standing still and turning back are certain death. It is often useful to know the number of useful choices. You may, for example, have an expensive look-ahead algorithm. However, if a cycle is in atari, then there is no need to run through your AI code: just take the one liberty.

Helper: getCycleCode

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "tron.h"

static int getCycleCode(int cycleIdx)
{
    if ((cycleIdx < 0) || (cycleIdx >= CYCLE_COUNT))
    {
        printf("JoelRobot.c: ***ERROR*** getCycleCode(%d)\n", cycleIdx);
        exit(0);
    }

    int code = 1<<cycleIdx;
    return code;
}
```

- What is this "code" being used for?
- How am I defining it?
- Why am I defining it in this way?
- Is it needed?
- Are there other ways to define it?
Gotta Love That `printf`

Prints up to a 20×20 grid within a single, default size PuTTY window.
Try printing your grid just before and just after a cycle crash.

```c
static void printGrid(void)
{
  int x, y;
  for (y=0; y<gridSize; y++)
  {
    for (x=0; x<gridSize; x++)
    {
      printf("%3d ", grid[x][y]);
    }
    printf("\n");
  }
}
```

Prints up to a 20×20 grid within a single, default size PuTTY window.
Try printing your grid just before and just after a cycle crash.

```c
static void printGrid(void)
{
  int x, y;
  for (y=0; y<gridSize; y++)
  {
    for (x=0; x<gridSize; x++)
    {
      printf("%3d ", grid[x][y]);
    }
    printf("\n");
  }
}
```

Cell encodes enter direction

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>256</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>64</td>
<td>64</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>256</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>64</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>256</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>64</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>256</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>64</td>
<td>64</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>256</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>64</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>256</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>3</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>256</td>
</tr>
<tr>
<td>256</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>3</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>256</td>
<td>32</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>3</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>256</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>3</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>256</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>3</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>256</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>256</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>256</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>256</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>256</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>