Tron Arena: A game for mobile devices

Hallgrimur Th. Bjornsson [*] Miguel Machado [†]

Lund University
Sweden

Abstract

Tron Arena is a tron based game designed for mobile devices such as a cell phone. The game is simple, resulting in a fun, easy logic and short-time challenge to the user. Several obstacles were encountered and overcome.

1 Introduction

The game was designed according to two main factors we considered to be of great importance when it comes to mobile gaming. The primary factor is game logic simplicity, since we dont want mobile gaming device users to spend too much time learning how to play the game. A too complex game may counterbalance its attractiveness as the typical user is a sporadic player looking for short entertainment. The other goal we seeked was to make a fun game so that people can easily find it enjoyable to play. In addition, we wanted to make a game which had considerable replay value.

2 Description

Tron Arena is a tron-based game. There is a square platform surrounded by walls where the player and the enemies' car move. All cars are in continuous motion and leave a random-colour trail along their path. The victory condition is to be the last standing car in the game without having hit any of the walls on the platform.

The player camera follows the player's car along the platform of the game, watching it always from behind with a wide field of view so that the player also sees to the sides (as shown by the screenshots). However, as soon as the player loses the camera will be set higher, giving a full-screened view of the game until some car wins.

3 Results

We present some screenshots of Tron Arena in the Appendix.

4 Discussion

The following sub-sections cover the several aspects of the game implementation and fits them into categories according to its efficacy and efficiency.

4.1 What works

Every feature included in the game works, even though it has some simple implementations of collision detection and illumination.

The illumination of the game is made of N+1 lights, N being the number of enemies. There is one low-intensity light above the platform allowing to see the shape of objects on the platform. Plus, every car has its own headlight, making it easier to identify the colours as the car goes along. The effect, however, isnt such a piece of eye-candy as we aspired for because the light is computed per vertex (our meshes only have very few vertices).

The collision detection algorithm used in the game is very simple and consists of only one checking point. If that point ever collides with any other trail, car or wall, the car is considered to have crashed and an explosion occurs.

4.2 What works really well

The game as a whole works as expected. It fulfills the objectives initially determined and provides a fun piece of entertainment.

4.3 Optimizations

Severel optimizations were attempted with the code in several aspects such as the number of objects/meshes created and the number of polygons loaded to draw the cars. This fact was particularly important since the collision detection algorithm's performance greatly depends on the number of polygons the "intersection ray" intersects. The car model optimization was done using `3D Max Studio 7 (trial version)` and consisted mainly of deleting unnecessary polygons, polygon optimizing and then exporting to m3g format.

4.4 Difficulties and obstacles

The first difficulty about this project assignment was to choose what kind of game to implement. Besides all the typical application-killer bugs, there were also some problems concerning exportation of 3D models to m3g format. In what comes to implementation itself, the major obstacle was getting the trail drawn correctly every frame without killing performance. Sounds was added but resulted in stuttering. Therefore, the sounds were not included.

4.5 What can be improved

Several aspects of Tron-Arena susceptible of improvement are listed below:

- Slow startup: this can be improved by using a model with no more than a few hundred polygons and smaller footprint. This way, the application would not take so long to load the meshES.
- Low FPS rate: This problem is due to having too many meshes on screen simultaneously. Reducing the car complexity would help, but not as much as having dedicated graphics hardware in the mobile device.
- Menus at the beginning and end of the game, including options such as choosing the number of enemies (default 2), showing highscores, choosing wether to play again, choosing the car-model to use, etc.
- The collision detection system could be based on a bounding box around the car where a ray is fired from every corner. This would make the collision more precise and realistic.

[*] e-mail: hallgrimur@gmail.com
[†] e-mail: mmachado@student.dei.uc.pt

This would, however, greatly diminish the performance of the game because of the number of rays.

- The artistic effects, so that the images are of better quality resulting in better explosions.
- Sounds can be added.

5 Conclusion

Designing and implementing the game has been a fun educational experience.

Any future work on this game should follow the guidelines above in ”What can be improved”.

6 Acknowledgements

- http://3dcafe.com/
- Java M3G Documentation
- http://developer.sonyericsson.com/site/global/techsupport/tipstrickscode/mobilejava3d/p_java3d_tutorial_part1_compliments_redikod.jsp
- Hybrid Graphics Forums: forum.hybrid.fi
- Mobile Graphics LTH website: http://www.cs.lth.se/EDA075/

A Appendix