# Lightcycle Arena

Computer Game Design

Team 12

COLOMBE Benjamin

VELLARD Thomas

DUA Guillaume

# Summary:

## Project Presentation

* 1. ) What is it?
  2. ) Inspiration.

## Gameplay

* 1. ) How to Play
  2. ) Other Feature

## Conception and technical details

* 1. ) Ogre 3D
  2. ) Object conception

## Possible improvements

# Project presentation

* 1. ) What is it?

Lightcycle Arena is a strategy game where the player(s) control an equivalent of a motorcycle in a virtual world. This motorcycle or “Lightcycle” moves in a Grid and creates a wall behind it which draw its path. The goal to win is to make crash the opponents against the wall of the Lightcycle or the walls of the grid (or Arena).

* 1. ) Inspiration.

Lightcycle Arena is a fan game inspired by the movies “Tron” (1982) and “Tron: Legacy” (2010).

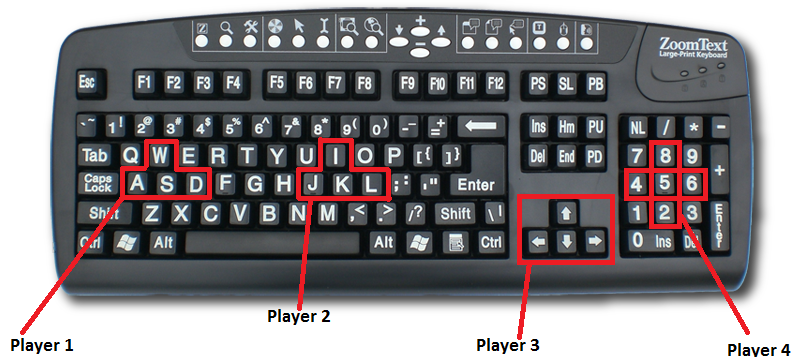


In each movie, there is a scene where the main character controls these Lightcycle which turn in right angle in the first movie and in curve in the second movie.

In the internet, we can find similar projects like GLTron (<http://www.gltron.org/>) but just based on the first “Tron”. So our game is an improved version of GLTron because we control the Lightcycle in a way directly inspired by the second movie.

# 2. Gameplay

2.1 ) How to Play

It’s really easy to play to this game, you just have to use 4 inputs (per player) :

W / I / Up / 8 to speed up

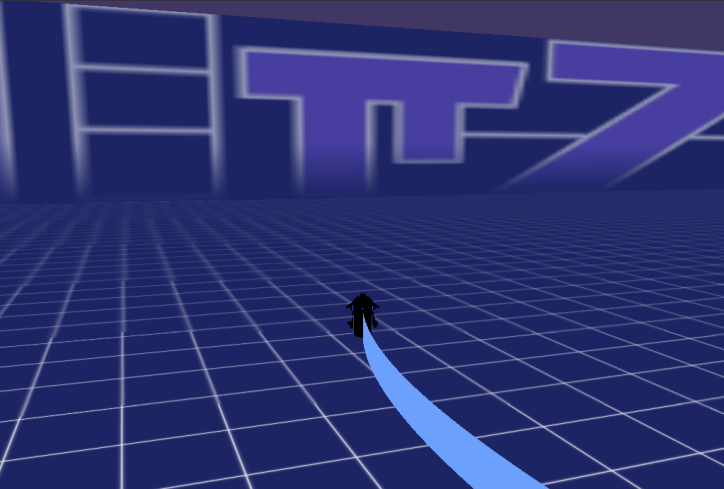
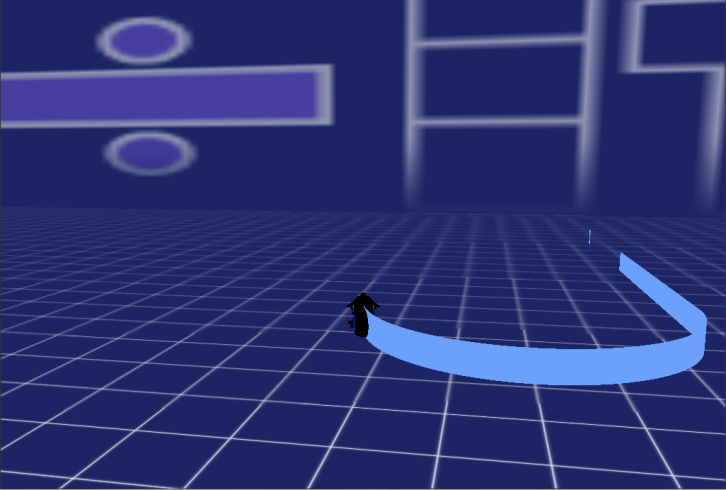
S / K / Down / 2 to slow down

A / J / Left / 4 to turn left

D / L / Right / 6 to turn right

ESC to quit the game (for now, it’s the only to quit the game)

During the race, when the player speeds up or slows down, the speed of his turn is directly impacted so it can help him to move and avoid collisions:

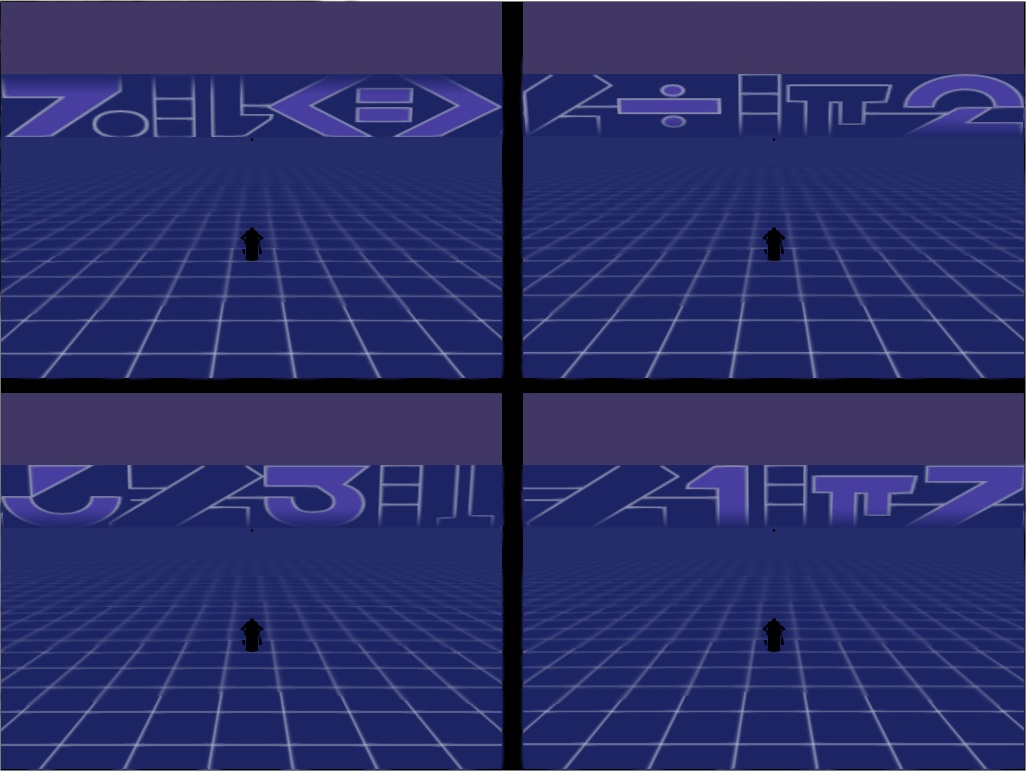
 

Turn when speed up Turn when slow down

This control gives to the player a maneuverability that let him free to decide what he wants to do, speed up to catch up his enemies or slow down to avoid crash.

2.2) Other Feature

The other feature of the game is that it is multiplayer in a split screen:



Not many things to say about it. The game allows a maximum of four players to play in the arena. However it makes the game lag a little bit and the collision system is disrupted.

# Conception and technical details.

* 1. ) Ogre 3D



Ogre is a C++ multi-platform open source graphic engine that supports Direct3D and OpenGL. Ogre uses a SceneManager containing all the objects that will be displayed. These objects, called Entities, will be attached to an object called Node. These Nodes will be used to apply all the movements and transformations we want to see. Several Entities can be attached to the same Node, which allows moving several objects with a single command. Each Entity can also receive a Mesh, a specific Ogre format that will give shades and colors to an Entity.

* 1. ) Object Conception

To make this Project, we have used an oriented object programing language (C++) to design each part of the game and organize the code. We used Visual Studio 2010 to design the game.



The project has 3 mains object:

* class Rwindow : where all the object of the game are created and the render loop is executed
* class Player : which contains all about the playable object (input capture, model, calculation of direction and position, wall creation)
* class Arena : which contains the collision system and where the player moves.

The class Player is the object where all the animation is done. We create the Ogre::Entity, the Ogre::Camera and attach them to an Ogre::SceneNode. Then, we can move the node to move all the objects. It also contains the method Player::UpdateSpeed() called in the render loop, update the position and orientation of the node and create the object Wall behind it :

void Player::UpdateSpeed()

{

[…]

***vitesse\_z*** = cos((float)(M\_PI \* (orient) / 180)) \* t;

***vitesse\_x*** = sin((float)(M\_PI \* (orient) / 180)) \* t;

node->setPosition(node->getPosition().x + vitesse\_x, node->getPosition().y, node->getPosition().z + vitesse\_z);

node->yaw(Ogre::Degree(r));

***mur->createWall***(node->getPosition().x, node->getPosition().z, orient);

}

(“vitesse” mean speed)

In this code example we can show how the model moves in the map with two speed vector by updating the position in terms of the orientation of the node. We can see also the method of the Wall object, Wall::createWall, which creates it just at the position of the player.

* 1. ) Collision System

The collision system is the simplest system in this project because we did it in the hurry of the dead line. It is a simple bool tab of the size of the arena, and each case is set as true when the player crosses this case. Then the player is destroyed when the arena detect that he crosses a case already set as true.

# Possible improvement

Due to time limitation, this game is not finished but here’s a list of different features that can be implemented to improve the gameplay or the ease of use.

* Playing through network with a client/server model. It would divide the number of calculations for each client, and doing so, improving the performance of the game, and decreasing the memory needed by the program.
* Another improvement could the possibility for the player to jump with his lightcycle, allowing him to avoid some walls.
* Added to the last point, creating a relief map would allow jumping further, and increase the strategic interest.
* Another possibility could be to add some “bonuses” like weapons, shields or turbo in order to, for example, destroy a part of a wall, or going through a wall without being destroyed.
* Another thing would be to add screens like a menu, a victory screen or a point’s counter, this would help the player in using the game.

So it’s easy to find some ideas of new features to give this game a real interest and make it a game that a game company could sell. It’s just a story of motivation.