Design Document for:

Wroom

Tha car game.

School Project for the Group; Single Rainbow Uicorns,

For our Game Programming class.

Written by Single Rainbow Unicorns

Friday, 21/12-2012
(Stil alive after the end of the world)

Table of Contents

Game Overview

What is the game?

(short: we envisioned an Micromachines clone (updated to 3d and with some new gameplay elements)

Its a casual racing game made for quick. easy and short gaming sessions. Its designed to fit into short down times, like recess, commutes and coffee breaks.

Why create this game?

We are making this as a school project and the game have some of the elements we all like in games (we all get something we want in games, from this game).

Where does the game take place?

The game takes place in the normal world we all live in, but you are racing tiny remote controlled vehicles on normal surfaces (like tables, bathtub, sandbox, etc).

What do I control?

You control a small RC-vehicle (one of 2-4)

What is the main focus?

As this is a racing game, the main focus is on “winning” the race, now you do that by getting too 10 points first (when you are way in the lead you get 1 point (everyone else lose 1 point) and the race puts all racers back on a shared start spot, and its the first to get the lead again).

Feature Set

General Features

Short game sessions
Easy to learn
Fast gameplay
3D graphics
Top down camera

Multiplayer Features

(Original:
Up to 4 players
Easy to find a game
Create game lobbys
Easy to find your pal in huge world
)

Multiplayer online got Cut:
We decided to focus on getting the singleplayer part of the game working, so we had a game to deliver.
Time was the reason for cutting multiplayer, but as it is now, you can make and connect to a server, and as the client "make" lobbies, but you can't connect to the lobbies.

Multiplayer on 1 computer:
Instead of online multiplayer, we added so 2 players can play at the same time on 1 computer.
they each control a car with one on "wasd", and the other on the arrows.

Gameplay

You create a vehicle
You are a racer
Follow the track
Try to be the fastest

Changed or cut:
Score the most points
(Changed to first person to score 10)

Beat your friends
(Only offline multiplayer mode, so you can only beat 1 friend)

Shoot friends in the back
Avoid the rockets
Demolish
(We removed the rocket or net as it was planned to be because of the lack of time)

The Game World

The Physical World

Overview

Its a world where people race tiny RC-vehicles for great fame and fortune!

Travel

Steering the vehicle and controlling its speed

Scale

The cars will look normal size and everything else will look huge.

Objects

Cars
Obstacles
Terrain differences
We now have walls around the waypoints so it's a walled in rectangular track with some boxes as general rubble and a jump.

Time

Will only be used to record race times.

Camera

Overview

The camera will be top down view and will follow the leading cars.

Camera Detail #1

It will be centered on the 2 cars that is the further ahead mainly, but will try to get every ar into the view.

When a car dies the camera stops trying to get it into the frame.

The World Layout

Overview

world is an endless track with a “micro” theme and obstacles, its 2d birds eye so, not a lot of the world will ever be seen by the player outside of the track.

Changed:
Time restraints made us decide to use 1 big track instead for our car, marked by waypoints that the cars have to follow.
So now our world is one big table, and we just add waypoints to make a track, we can add as many as we want, anywhere we want and the cars have to follow the given order (n to n+1 etc).
We are able to do this by first generating an vector (v1) between the waypoint the car is moving to (p1), and the point its going too (p2). so v1 = p2-p1. Then we take the position of the car and make the dot product between v1 dot (position - p1). so what we got is the length of distance to the vector that is perpendicular to the waypoint after the one we are going to. if that product is positive, then we are still behind the waypoint, and if it is negative then we have passed the waypoint.
In other words, we can make an infinite track by making waypoints, in any direction, and we have an easy way to find out if someone have passed a waypoint.
(Infinite track that can look anyway you want, and all you have to change is the waypoints.
together with the editor makes it easy to make infinite different tracks.)

Track generation

Track generation is one of the areas that went through several major revisions. Our first idea was to generate random track on the fly. That is good theory, but problematic implementation. First problem was how to make sure that the track won’t overlap. After some researching we decided to use Catmull-Rom splines. Later we discovered that although that creates interesting track, there is no elegant way how to get bounding box of result. Moreover there were problems with placing random obstacles on track. So as alternate way we decided to make track composed of precreated segments and have them ordered by random. Our first intention was to use Irrlicht ability to load scene nodes directly from XML files, but our benchmarking showed significant freeze upon during loads. So we naturally switched to custom loader since we (wrongly) presumed that delay has something to do with plenty of redundant attributes original XML included. In this stage we also introduced definition of segments in external files, since we would otherwise need to duplicate our code for loading physics and graphics representation, so we figured this will pay of in long run. After finishing our implementation we were quite surprised by fact that the lag experienced before was also present in our implementation, although shorter one. After serious code digging it was shown that the lag occurs during first render of previously not seen node. We made few tries to circumvent this but nothing worked on 100%. That was quite serious problem since the segments were supposed to be loaded on the fly. Preloading them all didn’t work thanks to nature of Irrlicht cache. Since these problems went with us until late phases of project there were no time to change track once again, so we changed our “segment” to be actually whole track. And that was the third, and last, implementation used.

Game Characters

Overview

Vehicles that can be customised by the player.
Race car, off road vehicle and a boat.
Changes:
We have the race car made for the demo, but you can't customize it anyway.

Creating a Character

You will have some control on key parts of your vehicle like speed, acceleration, turning speed, etc.

Changes:
The user can't customize the vehicles anyway directly in the demo. They can however meddle with definition file in config directory.

Enemies and Monsters

Only enemies will be the other players .
Changes:
We decided later on that we needed an AI as well. The AI was first made without using look direction of the vehicle, but since that didn't work as well as hoped, we are now using look direction.

User Interface

Overview

Menu > practice, multi,option, exit. ingame menu, and scoreboard.
Done:
all the menu's outside of the gameplay is implemented.

Lobby interface #1

Left side will be the options (track, nr of obstacles ect) that the creater can change (for the rest it will just show what the leader picked),
and on the right side will be the players, and they will each be marked with a red car for what car you are, and the rest marked as green cars (you can see what car you are by the color) and each of the joined players will have a "ready" button that will turn a red light into a green to show they are ready.

Server interface #2

the server won't have an interface, more of a rolling text to show important events (nr of active games / create and close of lobbies etc.

Weapons

Cut:
Plan was to have a net to hinder the other players,
but thanks to time limits we were not able to implement it.

Overview

1 net that is on a short CD

Weapons Details #1

It will “trap” some for 0.5-1 sec, just to slow them down.

Weapons Details #2

the weapon will have unlimited “shoots” but will be on a cooldown.

Musical Scores and Sound Effects

Overview

Background theme + car sounds.

Red Book Audio

We are using mp3 for the background theme, and wav for the test sound. Those two sounds are the only sounds being used for now.

3D Sound

The sound API we're using is called IrrKlang as it is highly compatible with Irrlicht. We don't use the 3D feature for the background theme as it is not needed, but we do use it for other sounds. The background theme is currently played with 2D sound.

Sound Design

The sound we are thinking of using is just a generic background noise, and then have sound effects from the net and car sounds and collisions and things.

Single-Player Game

Multiplayer Game

Single player and multi is the same game.

simple put, single will be more of a practise vs 4 “bad” ai players, and the multiplayer will be vs 1-3 other humans (and 2-0 bots).

Overview

Race vs each other, from waypoint to waypoint.

Point of the game is to be the fastes cars, when the other players fall to far behind the leader gains a point.

When 1 player gains a point, all the rest loses a point (first to 10 wins, cant get less then 0)

Max Players

2-4 player games.

Servers

All multiplayer games will be hosted on a server. Players will make lobbies that other players can join (max 4 at a time) then players will be able to join the games.

Customization of the race mode

Picking the track theme, size, race win condition, etc.

Gaming Sites

As this is a school project, we won't include or share it on any sites.

Persistence

Its not a world per say, so its not really persistent.

Saving and Loading

Game wins will be saved between races in the lobby, but everything will be lost when the lobby is closed.

Rendering

Overview

For all video output we are using Irrlicht capabilities. It can provide several so called drivers for rendering with DirectX and openGL being most known among them. For our needs we decided to go for openGL, since we wanted game to be multiplatform.

Irrlicht uses quite common approach to handle scene as graph, allowing nodes to be nested and effectively tying their position with parents. Since our approach to game flow used tight representation of physical models, we mostly didn't make any use of nesting capabilities and left any constraints that could appear (ie vehicle chassis-wheel) to be resolved by physics engine.

Vehicles

The graphical representation of vehicle is composed of one chassis and several wheels. The number of wheels is not currently limited as for providing broader possibilities to future models.
Both chassis and wheel model were created in Blender with UV mapping enabled.

Particles

We are using Irrlicht implementation of particle engine for creating effects. For now the effects are only used for denoting the position of waypoints

Other objects

Our file format for tracks currently supports only pregenerated box and arbitrary mesh model. Both can be skinned and some support for bump mapping has been implemented as well.

World Editing

Overview

After our decision to move data to external data files, it become apparent that hand editing was not viable option. Since support tools can do whole lot of difference we decide to pursue that and try - for us until now unknown - area of supporting tools. As with game menus, the Irrlicht gui code made a lot of difference and allowed us throw out first prototype quite soon. However work on this got delayed as we tried to solve another game-crucial aspects of the game, such as AI.

Controlling

Not being content with modelers like controlling we prototyped on FPS like camera and we are content with outcome. To enter movement it's necessary to hold right mouse button.

Editing

From editing perspective there is implemented only on click selection and basics of toolbox are shown.

Physics system

Overview

After some initial research we implemented our physics on Bullet engine, that is well known and even used in several AAA games. As the solution to notifying about updated position Bullet uses MotionStates. It's special class that can be attached to each body and may be used for writing and reading body position. That way we can save time by not updating/syncing nodes, that hasn't moved.

Supported Object

As for our initial implementation we decided to go only for boxes, spheres and cylinders collision shapes. Even complex meshes as vehicle chassis are, from perspective of the Bullet, reduced to their bounding boxes.
The vehicle is using special representation commonly named as raycast vehicle. It's name that, because the wheels (again, only from bullet point of view) are considered only as rays that that shoots out from chassis to detect how far is the ground. The rendered wheels representation is only visual candy which position is calculated from damping values that are applied to vehicle.