Game Programming Laboratory --- 251-257-00L

SpeedThugs

Formal Game Report

Jens Puwein

Pascal Rota

15.04.2007

Table of Contents

Table of Figures

Game Description

SpeedThugs is a high-speed racing game with hovercraft-like vehicles. As an exemplary game you might think of the WipEout series, the F-Zero series or the Extreme-G series, but only in the wide sense.

Detail

Game Modes

The goal of the game is to finish the track as fast as possible or crossing the finish line in first place, depending on the game mode. If you play in the time attack mode, you have to finish the track as fast as possible without exceeding a certain time limit. Single player race lets you drive against computer guided enemy vehicles and in the two player mode you can race against your friends. The possibility to tune vehicles in terms of their properties would be even more fun. Good results in the time attack mode let you unlock better vehicles, giving you the chance to master harder time attack challenges and again get a better vehicle...

The Track

We intend to make one track with curves, jumps and possibly movable obstacle. The track should be a challenge, including shortcuts and alternative ways, where shortcuts are harder to make than the normal route or may only be reached by using a nitro boost. Power ups are placed regularly and sometimes there will be power ups which are hard to reach but worth the effort. Gravity fields along the track force you to adapt your driving style. Not making a jump to the other side results in a reset of the vehicle on the track, which will cost time. Imagine the track looking like a roller coaster. Background details around the track could be added.

The vehicles

We intend to have two different hovercraft vehicles, each with different properties (e.g. Acceleration, handling etc.). They should behave physically correct in the sense of Rigid Body motion and have typical features like nitro boosts, rocket launchers, energy guns, mines etc., each one allowing you to either drive faster than your opponents or to slow your opponents down. Of course there is only a limited amount of each special feature, but you may collect power ups during the race to refill ammo and nitro. The vehicles have a limited energy shield. If you take damage, the shield is reduced. An empty shield will result in a reset of the vehicle and therefore cost time. Power ups to refill shield may be collected.

Weapons, Nitro and Energy

Initially the vehicle has no nitro and no weapons, but it has a full energy shield. Everything can be refilled by collecting power ups. Using a nitro boost results in higher acceleration and higher maximum speed. The boost only lasts a limited time. Energy gun shots look like pulsing, light emitting balls. They do medium damage and are also available as a multi shot version, where three shots are fired at once into three different directions. Rockets have a homing ability and they avoid collisions with the track boundaries and obstacles, tracking down the next enemy

vehicle. Mines are placed behind the vehicle and they explode when a vehicle drives over them, damaging that vehicle.

Collisions

Collision detection is performed between the different vehicles, obstacles and the track itself. Collisions may result in damage to the vehicle or in a slow down. If you get too close to the boundary of the track, you will be slowed down. If you hit an obstacle, the boundary of the track or another vehicle, you will take damage and be slowed down.

Rigid Body Engine

A good Rigid Body engine should allow us to place movable obstacles and jumps along the track. Hitting objects is fun and therefore should not always be penalized with damage or a slow down. Varying the gravity at different places along the track should look cool and make the track more challenging since players have to adapt to alternating environments. Imagine for example regions of low gravity and regions where the direction of gravity changes and therefore you will be draged towards the track boundary or you even drive “upside down”.

Graphics

The game comes in a cel shading look, giving it a futuristic arcade look. Additional shaders may support effects like motion blur at high speeds and other effects for explosions and weapons. Especially motion blur helps to make the player “feel” the speed. The camera will be set behind the vehicle.

Sound

Sound effects are supposed to add to the experience as they demonstrate the power of a rocket or a nitro boost, for example. Varying style and speed of the background music gives a better feeling of the current situation, for example when enemies are close or time is running out.

Mock-Ups

Abbildung 1: WipeOut Pure

Abbildung 2: F-Zero GX, Gamecube

Abbildung 3: Extreme-G, Nintendo64

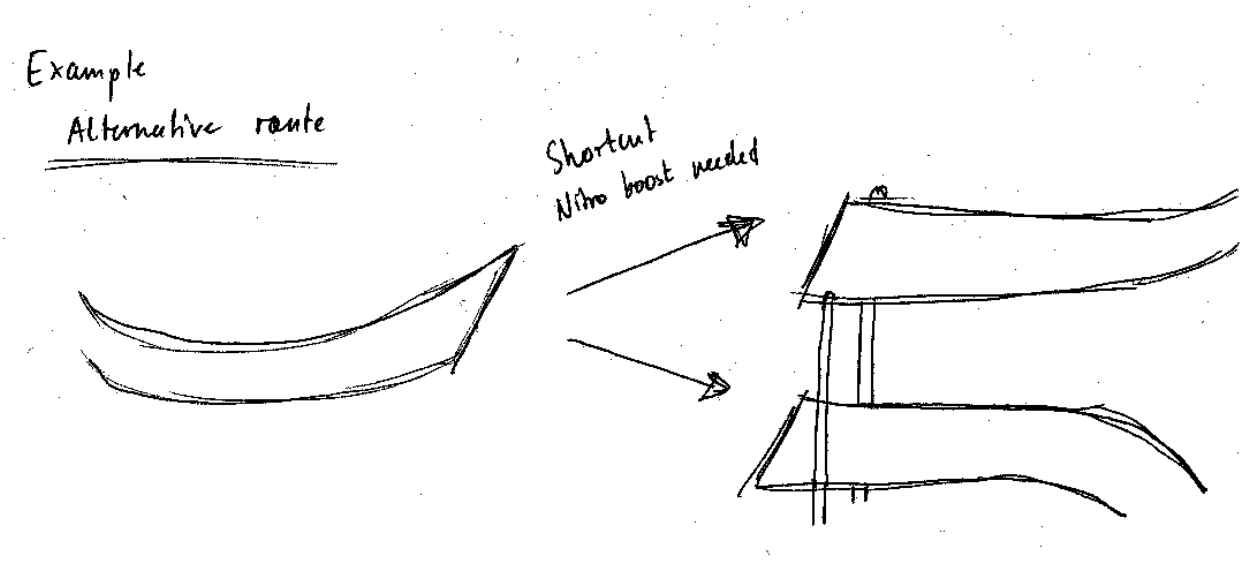


Abbildung 4: Shortcut illustration

Energy gun shot:

Missile:

Development Schedule

Layered Development Description

Functional Minimum:

Basic physics including collision detection. The user can drive a vehicle on something that looks like a racing track. The vehicle has an energy shield and power ups to refill the shield can be collected. Time will be measured. Nitro boost and nitro boost power ups to refill the nitro are included.

Low Target:

Cel-shading is implemented. The energy gun is implemented. Hitting another vehicle with an energy gun shot will damage that vehicle. Energy gun power ups to refill ammo can be collected. The track and vehicle models look acceptable, the track is not too simple.

Desirable Target:

Split screen support for two players, (dumb) AI allowing for computer controlled enemies. Sound effects and background music are included, background music does not adapt to the current situation. The rocket launcher and mines are included as well as the corresponding power ups. Motion blur shader effects.

High target:

Really cool, detailed and witty tracks, good AI. Background music adapts to the situation. Shaders for explosions and other effects.

Extras:

“Excellent” AI, particle effects like smoke, additional track/vehicles, background details for the track.

Detailed Schedule

	hours	
XNA (getting started)	10	
Physics	40	
Track	25	
Vehicle	15	
Camera	15	
Cel-Shading	30	
Weapon	25	
AI	25	
Sound	30	
Motion Blur	15	
Split Screen	5	
Small Effects	20	

	3.4	10.4	17.4	24.4	8.5	15.5	22.5	29.5	5.6
						α		**β**	
Physics	P	P	P						
Track			J	J					
Vehicle			P						
Camera				P					
Cel-Shading/Getting started	J	J	J						
Weapon				P	P				
Play Test				J/P	J/P	J/P	J/P	J/P	J/P
AI				J	J				
Sound					P	P			
Motion Blur					J	J			
Split Screen							P		
Small Effects						J	J		
Presentation					J/P				

Assessments

The game should give the player the feeling of speed. Everyone who likes high-speed racing games or arcade racing games in general should like the game. Besides that it should look cool, of course. If the movable obstacles are included, it should also be fun to destroy and hit other vehicles and obstacles.

We will consider our game a success if it is fun to play and keeps the player coming back, entertaining him for more than just a minute or two. Additionally it should convincingly simulate speed.

Reports

Interim Report I -15.04.2007

Physics Engine – Pascal

The Physics Engine was quite a challenge. First we searched the internet for physic library and we found lots of them. To name some: Ageia PhysX, ODE, Bullet Physics Library and lots of other stuff.

The problem with those libraries was that none of them was designed for C# or even for languages without pointer arithmetic or other low level optimizations. So we decided to port build our own physics engine with only what we need.

Later on, we needed an LCP Solver for resting contacts and we found one in a library called “Simpack” (Details can be found at simulator.wordpress.com or sourceforge.net/projects/simpack). Our first thought was that we won the first price. An LCP Solver in Java, let’s port it to C#!

Our decision was not optimal, because porting was not as easy as thought and the differences between Java and C# are big. Mainly the differences in the class inheritance structure and the deep nesting of the “Simpack” library gave me some headache and sleepless nights.

While porting was seeing an ending and some feature already implemented (rigid body motion, collision detection and resting contacts (nearly finished)) our assistant informed us about a project called XNADev.Ru which ported the “Bullet Physics Library” to XNA. Well I spent some hours for nothing and ended with a library, which is after testing it, good suited for us.

At the moment, I am working on the vehicle physics and a demo of it will be hopefully shown on Tuesday.

Cel Shading – Jens

Cel shading was not as difficult to implement as we thought. Getting started took some time, though. To quantize the lighting we did not directly use the factor obtained from the diffuse and the specular lighting as a scaling factor. We used it as a texture coordinate to a 1D grey value texture instead. This works fine if there is no color interpolation. Otherwise the color has to be quantized as well, which is not implemented yet (in fact we tried to quantize the different color channels separately, but that didn't look good).

The black lines at the object boundaries are achieved by rendering the whole scene a second time, only rendering backfacing faces, moving the vertices along their normal and using black color.

The fine tuning of the shader has to be done at a later stage when we have the models etc ready. Things like different orientations of the triangles, shading coefficients etc have to be set.

Track – Jens

Designing and modeling a track has turned out to be much more difficult than initially assumed. We started using Blender, but we are not sure if we really want to stick with it or change to Maya (if there are no bad surprises, it should be possible to export the models made so far...). Orientation of faces (CW/CCW) have to be made consistent.

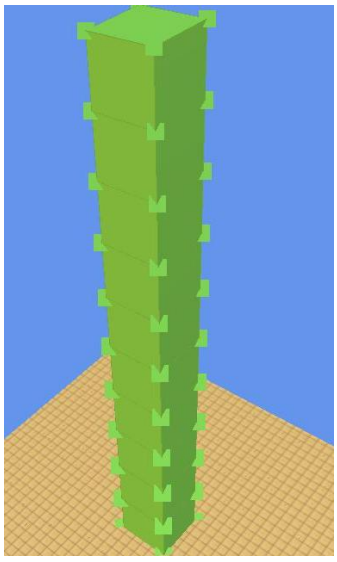
Making straight lines is easy of course, just stitching base elements together. Designing curves and twists is what's on the schedule now... Once a prototype is ready, the design of the final track can begin (using pen and paper first, using elements we know we can model and integrate).

Progress

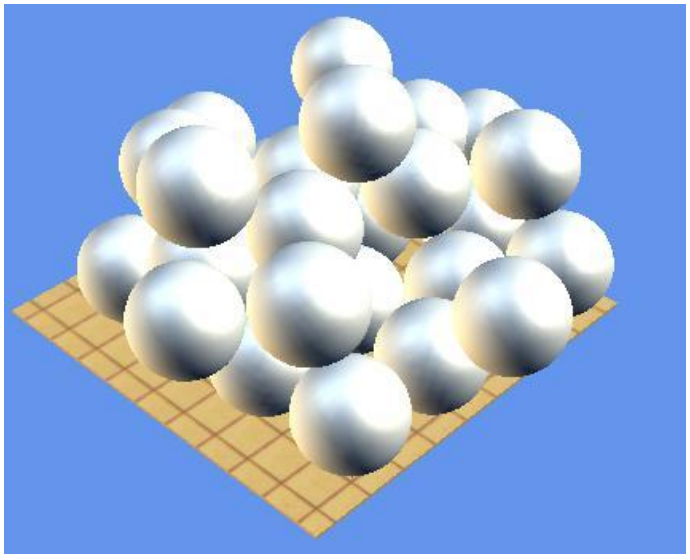
Some part of layer 1 and 2 are already done and we are in time with our schedule.

Pictures

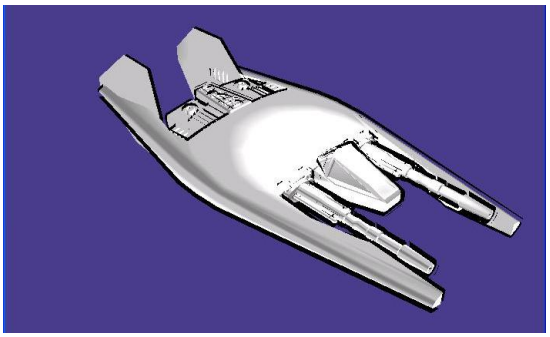
Physics Engine (own)



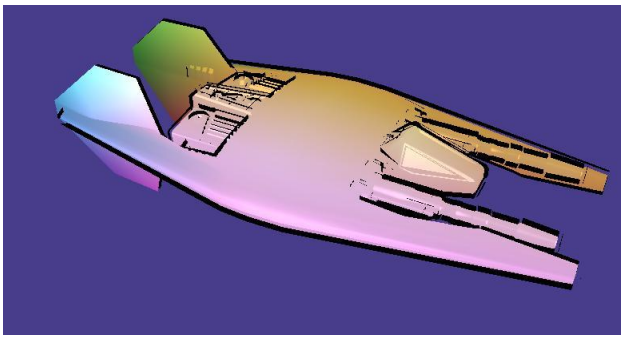
Physics Engine (Bullet)



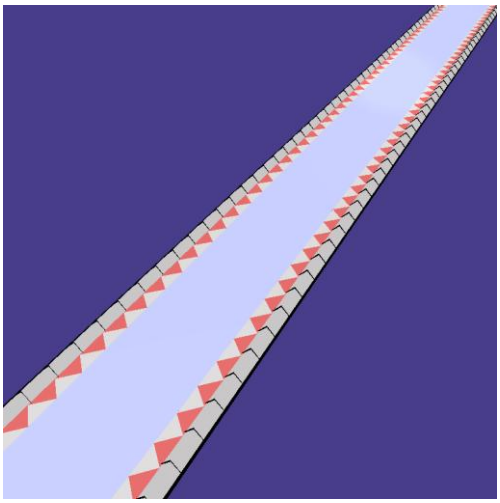
Cel-Shading (1st test)



Cel-Shading (Colored)



Track



Track (start)