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

SpeedThugs

Interim Report

Jens Puwein

Pascal Rota

13.05.2007

Table of Contents

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.

Interim Report II -23.04.2007

Physics Engine – Pascal

It turned out that the “Bullet Physics Engine” port to C# is not as useable as expected. The main drawback is that on the Xbox 360 it can sustain only about 10 moving colliding objects without dropping frame rate under 30.

Therefore we decided to begin a new game engine, which does not base on rigid body or on real world physics. It will be a pseudo physical engine completely tailored to our game and supporting different gravity directions and possibly also jumps.

Track – Pascal/Jens

A first, very simple track model was finished by Jens and it will be implemented into the game during following week.

HUD – Jens

We implemented a HUD which shows an energy bar for shield and nitro, the time for the current lap, the available/selected weapons and the speed.

Interim Report II -30.04.2007

Pseudo Physics Engine – Pascal

The new game engine is going well. It is now possible to race on a track which has different gravity direction, but it still needs a bit of tuning.

Game Foundation – Pascal

I implemented some game foundation, e.g. class structure for different game modes and also a vehicle and track selection screen. Nothing interesting, but someone has to do it.

Shaders – Jens

To cast shadows we use a shadow map technique. At the moment a point light source is moving along with the car to simulate the shadow of directional lighting. This has the drawback that also the track boundary may cast a shadow if it is inside the light region. We intend to place point light sources along the track. It is also possible to cast lights with texture, i.e. for example a light that projects “START” onto the track and the vehicles.

The motion blur/tunnel vision effect is pretty simple. To achieve motion blur we simply display a weighted sum of the previous and the current frame. Tunnel vision is simulated by darkening pixels far

from the image center. Parameters are passed to the shader to control the amount of blurring and tunnel vision.

Particle Effects – Jens

Getting started.

Pre-Alpha Release – Pascal / Jens

Over the weekend we combined our work resulting in a pre-Alpha release. We could play one track, which has two checkpoints where time is stopped. So we did some race against the clock and Jens won.

Interim Report III -13.05.2007

Sound – Pascal

I was browsing around the web and stumble over the "Racing Game" example on the XNA Creators Club web page. Good for us, there is some racing sound with the packing, which we included in our Game.

To include Sound in our game, I used the "Microsoft Cross-Platform Audio Creation Tool (XACT)" as explained in the tutorial. The only problem here was that XACT doesn't work on Windows Vista, no problem I simply took my old XP.

For the engine I found an F/A 18 engine sound, sampled it, and modified the pitch, etc. with Audacity. Then included it in XACT, included some "RPC Presets" which are bind to a variable called "Speed" and now it is possible to control the Pitch and Volume of the Jet sound during game play.

Split screen – Pascal

Split screen was not a real issue only something that has to be done. To avoid too much code duplication some classes has to be split up and created. Nothing exiting at all.

Car-Car collision is now introduced and works as expected. We used the XNA Sphere Bounding Boxes to achieve this in a simple and efficient manner. The collision is physically like an inelastic impact.

Particle Effects – Jens

I just followed the tutorial that was posted on the twiki (http://jsedlak.org/node/177) and adapted it to our needs. Right now we use our particle emitter to produce smoke.

Unfortunately we had/have some performance issues. After several experiments we came to the conclusion that the pixel fillrate might be the problem since drawing e.g. 2000 sprites of 96x96 pixels means a lot of pixels (~100*100*2000*20 (frames) = 400 Megapixel...) and reducing the sprite size increased the performance. But once we reduced the size of the sprites we got performance problems with the managing of the particles (updates etc). So we thought we could handle the particles on the gpu. After a while we came up with a suboptimal solution: one texture for the particles positions and one for their velocities (e.g. 300*300 pixels). In a pseudo drawing pass where we render a quad that occludes the whole screen (and therefore all pixels in the pixel stage are accessed) we update the positions and velocities. To render the particles, we pass the appropriate number of points to the graphics card (e.g. 90'000) and read their positions by fetching the positions texture in the vertex stage. This is done by assigning each point a unique texture coordinate referencing one pixel in the positions

texture. Unfortunately this didn't work completely since some artifacts remained that we were not able to resolve.

There has to be an easier, more elegant way to do the particle handling on the gpu but we could not find it. But since the results we get with the cpu look ok we will probably stick with the cpu implementation.

We will probably use more particle effects if our schedule allows it (e.g. fire like effects etc).

Shader debugging – Pascal/Jens

We had some problems with the rendering. It turned out that one of the problems was the orientation of the faces (clockwise vs. counterclockwise) which is not chosen in the same way by different exports from different programs. A problem which remained is the following: if we use *.fbx files which include materials then those parts of the mesh using materials and no textures are rendered black if we use our custom shader.

Track

Now that we know that we are able to drive the vehicle on non-horizontal tracks we may start modeling a more sophisticated, probably final track. It should be about ten times as big as our current track (which has a simple, oval form) and take between 90 and 120 seconds to complete a lap. We are not sure yet whether we will include jumps/are able to include jumps.

Overall Progress

Level 1

Complete: Simple track, collision detection, Nitro

Incomplete: Powerups

Level 2

Complete: Cel shading, track and vehicle look acceptable; track is a bit too simple though.

Incomplete: Energy gun and powerups (Big mistake in our proposal: weapon before having enemies (be it 2-player mode or AI)...

Level 3

Complete: Split screen, sound (needs tweaking), motion blur

Incomplete: Rocket launcher, AI

Level 4/5

Complete: Smoke particle effects, but not yet in use

TODO List

- Weapons and powerups (?)
- AI (Pascal)
- New track (Jens)