Alpha Report

Game Programming Lab 2010

Thomas Siegrist (ts)
David Gerhard (dg)
Philipp Keller (ph)
Jonas Hauenstein (jh)

1 Contents

2 Development state

2.1 Progress

Since the interim presentation, the time went by pretty fast. There was also a dense workload from other courses so the time we spent on the game itself wasn't as much as in the last few weeks.

Nevertheless, we are making still remarkable progress. We have completely achieved the targets of layer 3 (desirable targets) and are working now on the fulfilment of the fourth layer (high targets).

3 Details on some targets

3.1 The Game Logic

The gameplay logic is in place by now. This implies the pickup / delivery process as well as the busted events.

For every delivery, the player is rewarded with a certain amount of fuel. This amount depends on the distance between the player's starting point, the next pickup and the next delivery point. The amount of fuel is not yet perfectly fine tuned. We expect the balance to be more clearly defined after playtesting.

3.2 Police Attendance System

Also related to the game logic is the attendance system of the AI police cars. For now, we have implemented this the following way:

If the player is driving without any barrels on the truck, the police don't care about him. This also insures that the time used driving to the next delivery point is entirely based on the players driving skills.

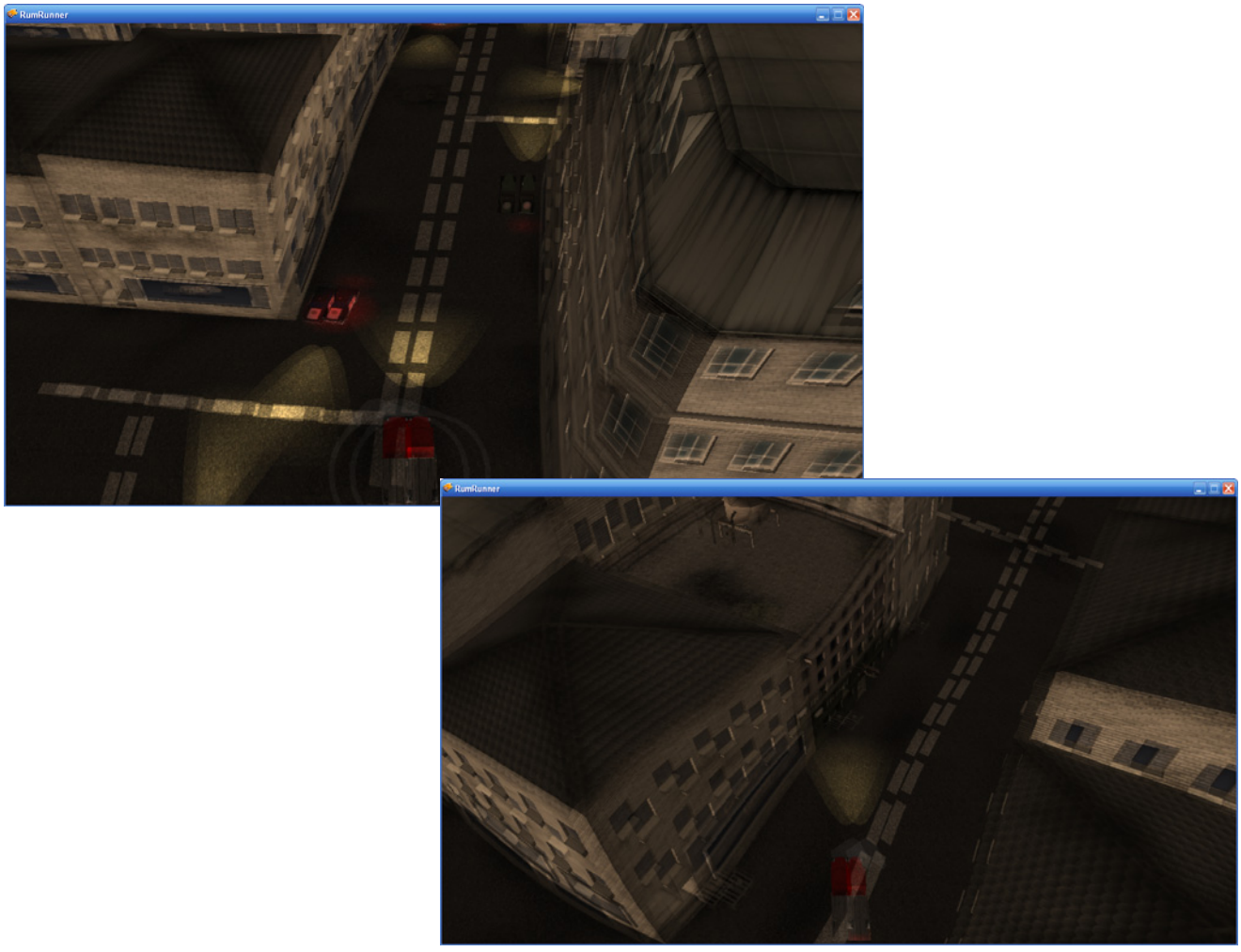
If the player has some cargo on the truck, the police will follow him as soon as his truck is spotted – meaning the truck is within a certain range around the police car. As soon as a police car spots the player, a "radio broadcast" is sent to all other police cars in a certain range. The police cars within this range will now also follow the player. This "Calling all Cars" radius will be increased based on the amount of successful deliveries the player managed to accomplish.

3.3 Drunk'O'Meter

We have implemented the drunk'o'meter as follows:

Upon delivery, the player takes a drink with the owner of the pub. This increases the drunkenness of the player – which results in an "appropriate" graphic effect (double vision and wiggle of the complete image) and a very small impact on the steering. The drunkenness decreases on a steady rate.

The idea behind this is – apart from being a funny feature of course – that a fast pickup-delivery time will be in some sort punished. For example, if the player is lucky on the randomly chosen next points, he will have the handicap of the drunkenness effect. The drive to the next pickup point (and depending on the drunkenness perhaps also to the next delivery point) will be a bit harder.



3.4 Sound

The game sound development progress is currently somewhere between desirable and high target. At the moment there are sound effects for all car engines, car brakes, car crashes and police sirens. As at the interim report stage, during gameplay, there is some background music.

All the sounds are set up inside an XACT project. The effect sounds in a standard wave bank and the music in a streaming wave bank.

The car engine sounds, siren sounds and crash sounds have RPC Presets for distance and Doppler Effect which are used by the XNA 3D sound engine.

To implement 3D sound, we first tried use an extended version of the example project from the XNA Creators Club (http://creators.xna.com/en-US/sample/3daudio). To limit the possible cues playing at the same time, they introduce a cue pool with a fixed size. For each sound event, a cue instance is popped from the pool and stored in a list during the time it plays. After it finishes playing, it is returned to the pool.

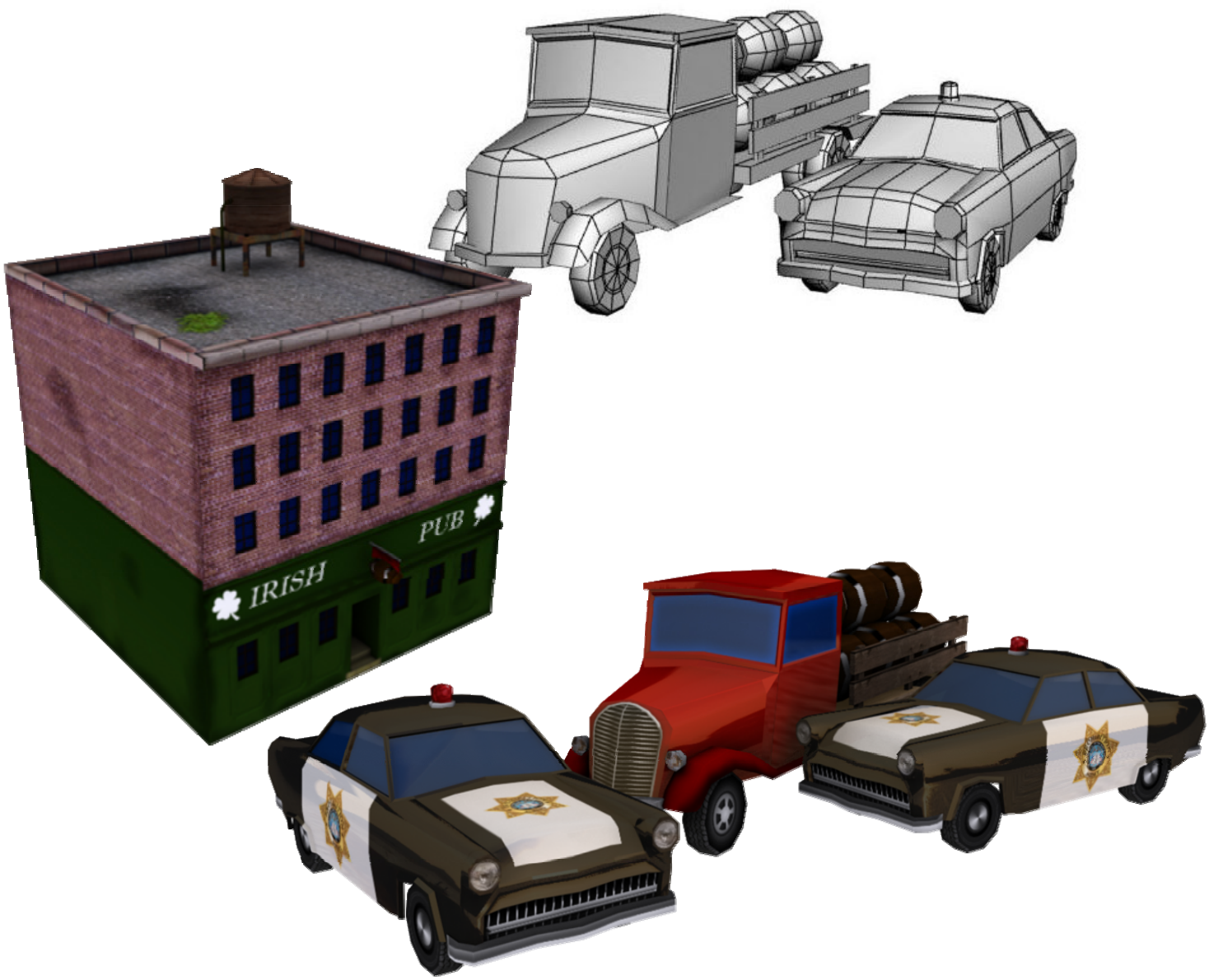
We tried to build the sound engine this way and we introduced pools for each type of sound (crashes, engines...) to be able to control the amount of simultaneously playing sounds of each type. But unfortunately, the XNA 3D sound engine didn't want to cooperate. It started to throw System.ArgumentExceptions from the call of the Apply3D method of the cues.

We tried to find a solution for this problem for quite some time, but finally had to give up and write the sound engine again from scratch.

The current approach creates at initialization time a dictionary with an entry for each car. The keys are references to the car objects, and the values are references to small helper objects containing four slots for sound cues. This way, the update method can simply iterate over the whole dictionary and external event calls (like for crash sounds) can access the data by key. Like this, each car could play 0 to 4 sounds simultaneously, which would be pretty much overall. Therefore, there are also a number of counter variables that decrease each time a cue is requested from the sound bank and increases after the cue gets disposed. This replaces the pools from the previous approach to limit the amount of simultaneously played sounds.

3.5 Models

We added some more building models and the truck and police car models have also been radically enhanced. The plan is to have some more building (with different textures) and perhaps one or two additional car models. But we are not sure, how much we can get done until the final presentation.



3.6 Optimizations

We have done the following to optimize the performance:

- Modified instancing: The number of instances have been reduced to the absolute minimum.
- Shader instancing instead of hardware instancing: Shader instancing seems to be faster.
- Modularization of the drawing process.
- The AI is now running on a separate thread.