Developing a Mini Racing Game

Results of a practical course at the Chair for Computer Graphics and Multimedia
(RWTH Aachen University, Germany)

Eric Béguet* Andreas Heuvels† Christian Janßen‡ Markus Schamberg§ Paul Voigtlaender¶

Figure 1: *Picture of a typical game scene showing cars in the front and meteorite impacts in the background. Also visible: the user interface.*

Abstract

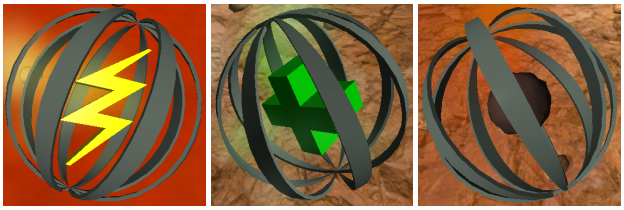
Our idea was to create a racing game with a Mars theme, because of the landing of the Mars rover "Curiosity". We wanted a competitive racing game, that can be played against others and is also visually appealing. The game is settled in the future where humans construct the first buildings on Mars. In a race against others, the player controls a Mars rover, and has to avoid meteorites which are falling on the track.
To create our game we implemented techniques like instancing, motion blur, soft particles and shadow mapping. Additionally we support a two-player mode and AI challengers for a competitive gameplay.

Keywords: game programming, racing game, Mars

1 Gameplay

As in any classical car racing game the player drives the rover on a track and tries to win against his opponents. Because the track has no borders, the player is forced to drive trough checkpoints. If the rover gets stuck or crashes and can not move anymore, it respawns at the last visited checkpoint.

The track consists of areas where powerups can be found. Those powerups give the player certain advantages: a speed boost, a manual respawn without delay and the ability to trigger a meteorite which will hit one of the other rovers.



(a) *speed boost* **(b)** *instant respawn* **(c)** *meteorite*

Figure 2: *The powerups used in the game*

Additionally, a meteorite spawns every 20 seconds without being triggered by a player. The meteorites' targets are chosen so that the players at the tail have a greater chance to catch up. The challengers in the game consist of vehicles controlled by basic artificial intelligence and optionally by another human player on the same computer. Even though the AI is based on a very simple algorithm, it is still a challenge for most players.

2 Workflow

As basic architecture for our software, we kept the Model-View-Controller approach which was provided in the template. In addition we used Qt's signal and slot mechanism to communicate loosely coupled between the logical part of the game and the graphics.

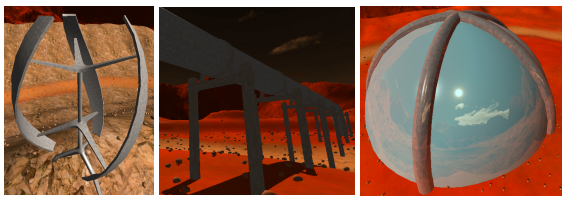
We split the workload into three parts. Two major groups for the graphics and physics each consisting of two developers and one for detailed content creation.
In our game the scenery consists of a large terrain with a detailed ground and only a few objects, therefore we did not focus

*eric.beguet@epfl.ch
†andreas.heuvels@rwth-aachen.de
‡chrisitan.janssen1@rwth-aachen.de
§markus.schamberg@rwth-aachen.de
¶paul.voigtlaender@rwth-aachen.de

on an elaborate method to create and export the whole scene from Blender. The time saved on this process justifyed the distribution of the developing group members and gave us the possibility to mainly focus on the graphical and physical parts.
To organize a consistently running project, weekly meetings were held allowing us to coordinate the development between the groups and increasing the efficiency.

3 Graphics

For the basic atmosphere we used a skybox representing a Mars scenery filled with mountains. The lumpy terrain on which the race takes place is rendered using a uniform triangle grid, whose height-values are sampled from a heightmap-texture in the vertex shader. We use multiple tileable textures to achieve a high resolution rendering of the ground. A lot of rocks decorate the ground, which at first caused performance problems. To solve this, we implemented instancing and multiple levels of detail, i.e. the amount of rocks decreases with the distance to the car. The scenery is enriched by several objects like pipelines, wind turbines and greenhouses (cf. figure 3). All those objects and the heightmap were created using Blender.



(a) *wind turbine* **(b)** *pipeline* **(c)** *greenhouse*

Figure 3: *An example of objects used in the game*

To enhance the visual appeareance of the greenhouses and the cars, we added cubemap reflection. The scene is illuminated by a directional light source which models the sun using phong shading. Additionally we implemented shadow mapping using an orthogonal projection. The shadow is only visible near the player's car as the scene is too large to be covered entirely by a single texture. In practice, however, this effect is hardly noticeable. Giving the game the feeling of high velocity, we implemented adaptive motion blur by averaging the color values of the last few frames, which are stored in textures. The scene is observed by a third-person camera which follows the car in a smooth way, i.e. not with a fixed angle.

Figure 4: *a meteorite with particles*

Particle effects are also widely used in the game to intensify the level of details of the overall scenery. They are used in several places. The most basic effect is the fire effect which is used behind the falling meteorites. Furthermore it is used when the rock hits the ground to create a kind of explosion effect. After that explosion the so-called smoke effect comes into play a few seconds at the impact point. Particles are also employed to create the dust effect made by the rover rolling on the land. But it is also used to create the permanent storm effect in front of the player's camera. Furthermore particle effects are attached to a variety of objects in the scenery (powerups, cooling tower..). Because the particles wouldn't mix up correctly with the landscape the Soft particles technique is used. It makes particles which are near the ground more transparent allowing a nice blending effect between the particles and the soil.

Our split screen implementation enables the hot seat mode. We added an UI which displays information like the current rank and speed of the player and a minimap.

4 Physics

The physics of our game is based on the Bullet Physics Engine. As a physical surface we use a heightmap, which is already supported by bullet. This gives us an efficent option to deform the ground in realtime, which is necessary for the implementation of craters, appearing after the impact of meteorites.
Meteorites spawn during the race, deforming the map and after multiple impacts the track becomes impassable. To eliminate this effect, the heightmap rejuvinates over time resulting in only a few impact locations on the map. To provide data for the heightmap, we use a png-image, where the height-value of the vertices are represented by the colour-value of the coresponding pixel. This png-file is an orthogonal high-angle shot of the map which was designed in Blender, where it was possible to work precisely on the result.
Additionaly we use another png-file, which stores the spawnspoints for random powerups and legal impact locations for the meteorites. All impact locations depend on the rovers position, its velocity, and its next reachable checkpoints.

Creating the car and a realistic driving behaviour was difficult in bullet. Due to the bumpy terrain the default raycast vehicle provided by bullet had a lot of unrelistic collisions. We solved this with additional raycasts for the wheels and reducing the size of the physical representation of the lower side of the car. Furthermore, to increase the driving stability we added adaptive forces depending on the velocity of the car and its ground contact. At higher velocities the car is pushed down on the track by a manually created force. Moreover to prevent somersaulting we also added a force around the z-axis.

5 Conclusion

This racing game was the first large project for all our team members. The lack of experience in working on a project of this scale led to some, because of the limited time, not feasible ideas. After discussing those with our supervisor, they were broken down to a realistic and viable concept.
In process of developing the game we gathered a lot of experience in creating, modeling and managing a project with team members. For the implementation we used various graphic techniques and tried to generate a suitable atmosphere for our Mars theme. We managed to set up an enjoyable driving experience through our modifications of the vehicle provided by bullet.

Despite the first problems we created an appealing game, that provides a unique challange, since the player has to react to the changing terrain which is caused by the imapct of the meteorites. In addition we were also able to implement features which were not initally intended like splitscreen mode, a minimap and AI-competitors.