Developing a Pinball Game

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

Philip Trettner[*] Philipp Bartels[†] Theo Dreßen[‡] Mateusz Buglowski[§] Michael Herwig[¶]

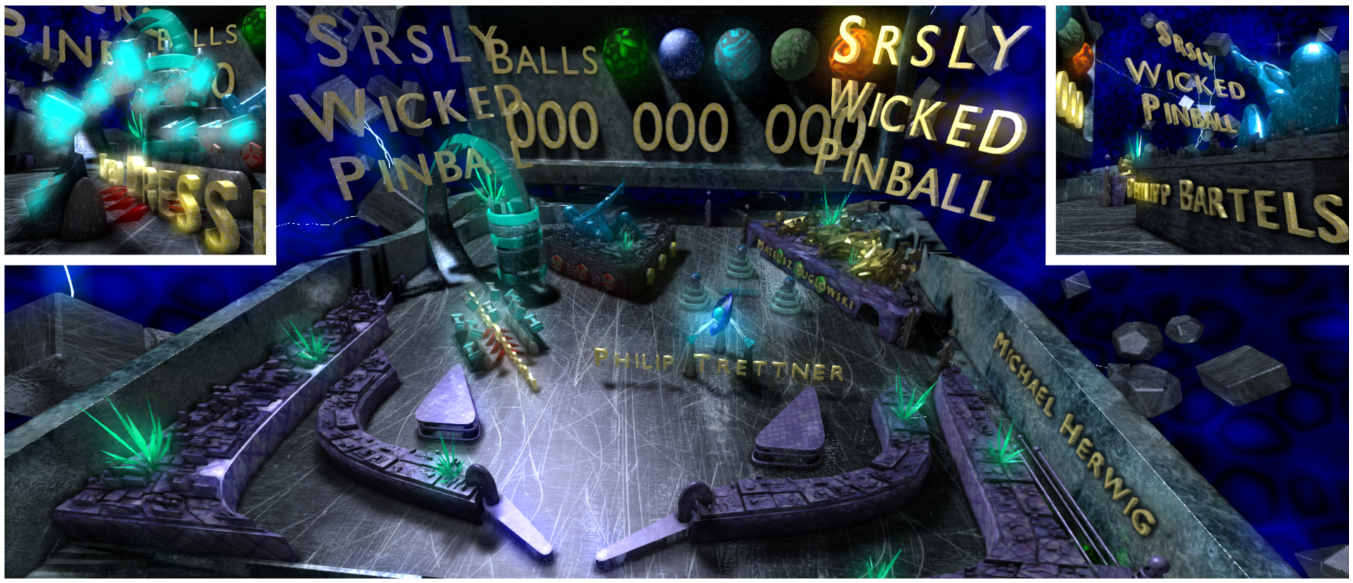

Figure 1: *Our pinball table is of high geometrical complexity and is rendered with advanced lighting techniques and modern graphics effects. It is stored in a large XML file with embedded Lua-scripts which is designed to be human-editable.*

Abstract

In this practical course, a pinball game was to be developed with a focus on the graphics and technical detail rather than the gameplay. We implemented a variety of modern 3D effects and used the Bullet engine for complex rigid body physics. The pinball table is stored in a huge XML-file that contains the composition and material configuration of all 3D object as well as Lua-scripts that are responsible for a majority of gameplay elements.

Our game is set in a futuristic and surreal world consisting of metallic and crystal-like materials. Mysterious energies power the pinball table as well as a maelstrom of background objects.

Keywords: game programming, pinball game, advanced graphics

1 Graphics

As our setting was metallic-looking and surreal, we employed several advanced rendering and lighting techniques in order to realize a graphically formidable and plausible game.

Instead of normal Phong lighting, which almost always looks like plastic, we used the Cook-Torrance reflectance model [Cook and Torrance 1982] for metallic looking surfaces. In addition to Phong shading, we utilized normal mapping to achieve high surface detail even on objects with low polygon count. Glow is used to enhance exceptionally bright spots and simulate scattering in air and lenses (cf. top left of figure 1). The scene is lit by one point light with exponential soft shadows [Annen et al. 2008]. In order to greatly enhance the metallic look of the scene, we render the complete (dynamic) background into a cubemap that is used for reflection. When updating the cubemap, the previous one is used for calculating reflections yielding the illusion of multiple reflections.

Our material system allows different textures and lighting parameters for every object. Figure 2 shows a variety of materials for the

[*]philip.trettner@rwth-aachen.de
[†]philipp.bartels@rwth-aachen.de
[‡]theo.dressen@rwth-aachen.de
[§]mateusz.buglowski@rwth-aachen.de
[¶]michael.herwig@rwth-aachen.de

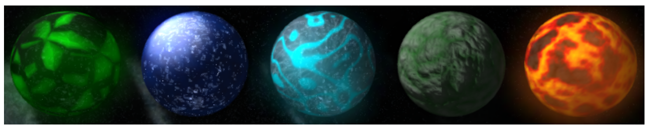

Figure 2: *5 different ball materials with different textures, reflection, specular, glow and roughness parameter.*

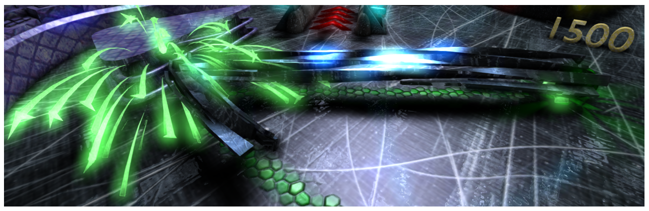

Figure 3: *Fast balls leave metal trails, cause scratches on the ground and throw out sparks on collision.*

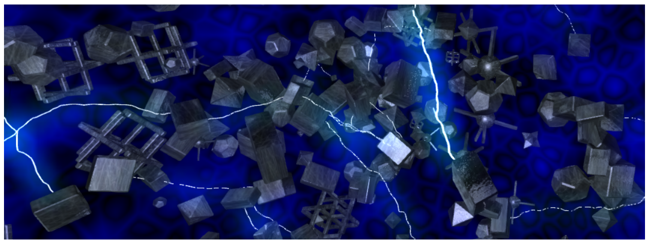

Figure 4: *Thousands of background objects, moving, rotating, connected by lightning. Objects are drawn with instancing, lightning with geometry shaders.*

balls. These balls have several means to visually interact with the table (cf. figure 3). Scratches caused by a ball are rendered into the ground texture and alter color, glow, normal and lighting parameter. Sparks and metal trails are created by a geometry shader and are able to cast shadows. Some balls project a light pattern onto their vicinity.

The background of the pinball table consists of thousands of moving metal objects with lightning that bounces between them (cf. figure 4). As this background is drawn six times for updating the cubemap and once for the actual scene, the rendering needs to be exceptionally fast, which is achieved by instanced objects and geometry shader generated lightning. Other minor graphics features include FXAA for anti-aliasing and fully integrated 3D text for menus and ingame messages.

2 Physics

The actual gameplay heavily relies on the Bullet physics engine [Bul Sep., 2012] as the complete table is part of the simulation in which the balls are launched, collide, repel, accelerate, score and finally vanish.

One particularly challenging problem is the high speed of the balls. Typical physics engines only support rudimentary continuous collision detection which is necessary to prevent high velocity balls to pass through other objects because the collision is not recognized properly. This problem is even further enhanced if the objects are triangle meshes without sound definitions of inside and outside in which case balls can be trapped within objects. To circumvent this problem especially for the flippers, we approximated the meshes by a combination of primitives known by Bullet which greatly enhances the simulation stability.

3 Gameplay

The most basic gameplay is provided by Bullets rigid body simulation. Additionally, our engine is notified in case the ball collides with certain objects or enters and leaves predefined areas. These events, together with some additional features like timers and applying impulses and forces are made available to a Lua scripting engine. Using scripts to implement high-level gameplay like missions or quests is not uncommon in games and allow for fast development of complex behavior.

In our case, several parts of the game are controlled by Lua. The accelerators (cf. figure 5(b) and (c)) can accelerate the balls based on their materials or if previous missions are fulfilled. Quests are presented to the player in regular intervals and demand fulfillment of several criteria like pressing one button of each color. The scoring mechanism is also controlled by scripts.

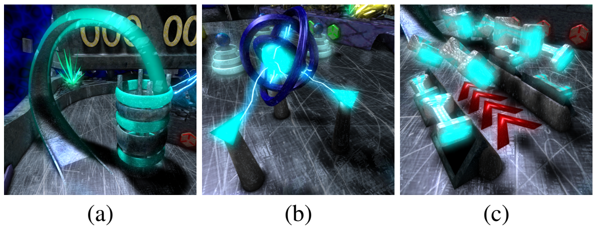

Figure 5: *Several detailed game objects: (a) looping, (b) triple accelerator, (c) railgun accelerator.*

We designed the interface of the scripting engine to be as general as possible such that the creativity of our level designer is limited by a minimum.

4 Assets

Our assets came from several inhomogeneous sources like *3ds Max*, *Blender*, *Photoshop* and *GIMP*. As a unified and easily extendable baseline we chose *.obj* files for models and *.png* (and sometimes *.jpg*) for textures.

At first, we were under the impression that we have to write a level editor and designed a feature rich XML-based file format for our table. After this misunderstanding was resolved we simplified our file format in order to make it easily understandable and modifiable by humans. This XML file additionally contains Lua scripts such that the complete description of the scene ("which objects are where and with what material parameter") together with the gameplay (the scripts) are in one file. While this approach leads to huge files (our level consists of about 3000 lines), editing, modifying and extending the level becomes an easy task for an adept level designer.

We designed the file format with relative paths to resources such that it is in theory possible to create a folder consisting of the level XML and all assets (textures and models) that can be bundled and shipped as "one level". A small exception is that the shaders are treated in a special way and cannot be included in the level folder.

5 Conclusion

Pinball games communicate a lot of its gaming experience by flashy graphics and fast gameplay. We developed a modern 3D game with rich graphics and consistent style to achieve a visually appealing game. A rigid body physics simulation allows for plausible and complex gameplay, while assets that are rich in detail further enhance the singular atmosphere that we ached to create. Our level file format permit extensive level design and bundles "level folders". Even though we started with ambitious goals, we were only forced to abandon few of them and through good teamwork we were able to maintain a pleasurable working atmosphere.

References

ANNEN, T., MERTENS, T., SEIDEL, H.-P., FLERACKERS, E., AND KAUTZ, J. 2008. Exponential shadow maps. In *graphics interface 2008*, Canadian Information Processing Society, GI '08, 155–161.

Sep., 2012. Bullet physics engine. bulletphysics.org/wordpress/.

COOK, R. L., AND TORRANCE, K. E. 1982. A reflectance model for computer graphics. *ACM Trans. Graph. 1*, 1 (Jan.).