Developing Shattered Star

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

David Erler* Joachim Herber† Eugen Seljutin‡ Leon Staab§

Figure 1: *A view of the main level, as seen while playing.*

Abstract

With the current trend in stereoscopic games, our task was to create a 3D-game for the Apple iPhone in combination with a Dive VR device. We chose to develop a game which, unlike other stereoscopic games which try to shock the player with horror elements or quick action loaded rides, relaxes him. Warm colors and a special low-poly look, combined with lovely level design within a natural scene, provide the ideal conditions to take a break from a hard working day.

Keywords: game programming, dive into mobile vr/ar games

1 Gameplay, Design and Setting

Since the target group of our game is not based on experienced players but on people who want to try out virtual reality, our gameplay is focused on simple mechanics. The most important interaction with the user is camera control by moving the head. This natural mapping does not require teaching. The other mechanic is actively looking at objects. But even without it, the game is already consumable and gives the player a simple VR experience without the pressure to fulfill objectives and learn mechanics. The graphic style is simple and symbolic and not cluttered with details that would draw away the players attention. There are no text elements at all and no HUD elements are floating in mid air.

*david.erler@rwth-aachen.de
†joachim.herber@rwth-aachen.de
‡eugen.seljutin@rwth-aachen.de
§leon.staab@rwth-aachen.de

2 IOS Development

Since no member of our group had previous experience with the target platform and we didn't always have access to a device, we chose to introduce an abstraction layer for the initialization process and player input. The usage of apple-specific APIs was generally avoided. The game was developed on multiple platforms, including OSX, Windows, Linux, with frequent intermediate iPhone builds to check compatibility.

3 Content Pipeline

As our main tool for content creation we chose Blender. A level can be modeled, textured and designed all inside a single Blender scene. Due to the static nature of the levels we chose to use the Blender renderer to bake ambient occlusion maps to add definition to the geometry. Colors are solely set using vertex colors. For a robust transfer of the content data into our engine, we decided to include the Assimp library as an importer. The importer would transfer the scene graph, meshes and materials to our engine, preserving the relations relevant to our systems. Using reserved identifiers, our level loading mechanism would tag certain nodes as being objects using a component of a specific system (e.g. navigation nodes, light spheres). With this method we skipped the need for a dedicated level editor and a custom level file format, promoting rapid development.

4 Entity Component Architecture

Our very first prototype used an ad-hoc architecture for the purpose of a quick working concept. We quickly realized an isolated set of systems would offer better flexibility and simplicity of development. As such, a simple entity component system was implemented. Once per frame, all systems are updated and allowed to interact on their components. Every component system could be

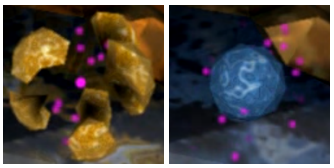

Figure 2: *A light-sphere before (left) and after (right) it has been collected.*

developed in an isolated and testable environment.

4.1 Event System

The event system is used for communication between systems. It provides event-channels for different types of events. The event-channels store events in a front and a back buffer, which are switched once per frame. This results in a delay of one frame for delivering an event, with isn't noticeable in our game, due to its slow pacing. The benefit of this approach however is, that it avoids blocking the game due to infinite event chains.

4.2 Selection System

The selection system notifies related systems about targets, which were selected based on the direction of the viewer. While the user is looking at an object, its selection score increases. The incrementation rate is controlled by the angle between the relative position of the object and the viewing direction. This prevents accidental selections.

4.3 Navigation System

The navigation system operates on a tree structure of nodes. Upon initialization, bézier curves are generated, which define the path of the camera. During navigation, the system updates the position of the camera and detects branching points. When approaching a branching point, the movement is slowed down up to a stop, or the user selects one of the possible branches.

5 Graphics

We decided to use a deferred pipeline, because of the high number of light-spheres, which act as point lights. Our deferred shader writes all information required to combine the resulting image in a G-buffer, which includes the color, normal, ambient occlusion, world position and depth of each pixel. This information is used to draw the ambient lit world in the first step in a full screen pass, and later adding lights additively in a second pass.

5.1 Light Spheres

The rendering of the light-spheres is done in two steps. In a first step the lighting of the scene geometry, due to the light-spheres, is calculated and the geometry of the light-spheres is then rendered in a second step.
The progress of collecting the light-sphere determines the color, through the use of a 2D-Texture. The pattern in the color of the light-spheres, as seen in figure 2, is generated based on a noise-value. The noise-value for one point is generated based on a 3D-Noise-Texture, its direction from center of the light-sphere and the current time.
An other way that we use to indicate the progress in collecting a light-sphere is the displacement of the light-sphere pieces. This is done by assigning a displacement vector to each vertex of the light-sphere model, which is scaled based on the current progress in collecting it.

5.2 Particle System

The particle system is used to highlight gates and give the user feedback of the currently selected entity. It is implemented via two alternating buffers used for transform feedback.The updating of particle positions in done entirely on the GPU, for better performance. Each emitter has an origin, a direction, a velocity, a lifetime value and a buffer used to store information on its particles. The emitted particles are initialized with the values of their emitter, provided with some random bias to provide a nice looking effect. The cone angle in which particles are emitted, depending on the direction, can also be adjusted.

5.3 Transparent Rainbow Path Rendering

The geometry of the rainbow path is generated on the fly when a level is loaded. This allows for a dynamic creation of the path, as the rainbow path is calculated based on a hierarchy of bézier curves. To acquire a continuous rainbow color gradient, even over the borders of single curves, a global distance to the origin is calculated for each (sub-)path and used for drawing.
To achieve correct transparency, all road segments have to be sorted. The sorting is done based on the middle points of the road segments. For correct transparency, the rainbow road has to be drawn at the end of the rendering pipeline since the lighting is deferred, prior to the post-processing.

6 Tools and Libraries

For the development of our game, we used the following external tools and libraries.

Assimp Library used for the import of Collada files.

GLM Mathematical library

LodePNG Library for cross-platform image reading.

Blender Tool for mesh modeling, AO-baking and level design.

XCode IDE used for iPhone development.

CMake Cross-platform build system management.

7 Conclusion

The intended low entry barrier to our game allows casual users great experiences. The use of warm colors and a peaceful natural environment proved to be positively received. A working animation system was implemented, but is currently not being used due to time restrictions of the level development. Our use of collada files as levels opens possibilities for simple development of further levels. The full potential of our engine could not be used in our levels, but many of our design decisions lead to positive results.

References

2015. Open asset import library. http://assimp.sourceforge.net/.

2015. Blender. http://www.blender.org.