Dive into mobile VR

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

Rizwan Ali* Bastian Jonas† Jonathan Wendt‡ Florian Wehling§

Figure 1: *a) ingame menu, b) view of an advanced level*

Abstract

”Virtual Workshop” is a virtual reality game for the iPhone. Based on a hand-detection, position- and orientation tracking system you can solve physic-based puzzles in a virtual 3D scene. For maximum immersion-effect, the game is meant to be played with 3D stereoscopic VR glasses.

Keywords: game programming, physic game, virtual reality, augmentend reality, iphone, head mounted display, hand detection, opengl, real time graphic, room tracking, position detection

1 General Information

”Virtual Workshop” is a virtual reality game developed in the practical course ”Dive into mobile VR” at the Computer Graphics and Multimedia chair of the RWTH-Aachen. The game consists of several levels, each representing an unique puzzle. In the first phase, the placement phase, the player has to move given items to the correct spots. Then, in the second phase, the underlying physic-engine takes control over all objects in the scene and simulates what will happen with them. The puzzle is solved, if the target-task is fulfilled. The technical challenges and solutions are described in the following sections.

2 Technical Details

2.1 Drawing the virtual Environment

The whole game is developed for IOS 8.x and therefor the underlying graphic API is OpenGL ES [Ope c]. We choose to use the version 3.0 of OpenGL ES to be able to use some more advanced features. The challenge for our graphic engine is drawing a variety of objects in a way that the player has the feeling of being part of the scene. Furthermore, the graphic system should be one of the first working parts in the development process and thus has to be as flexible as possible for upcoming design decisions (e.g graphic materials or effects). Because of that we decided to implement a script-driven material and effect system which allows us to quickly introduce new visual effects and enhances the flexibility of the entire rendering system.
As we are targeting a cartoonish style (2.5) with the textures of our game, we not only implemented per-fragment-lighting and basic texturing techniques, but also added a few advanced rendering techniques like shadow mapping, particle systems and transparency to support it. To reduce aliasing and to show the power of our effect system, a simple ”Fast Approximate Antialiasing” (FXAA) is part of our rendering engine as well.

Figure 2: *Move in the game world with dive glasses and green gloves*

*ali.rizwan@rwth-aachen.de
†bastian.jonas@rwth-aachen.de
‡jonathan.wendt@rwth-aachen.de
§florian.wehling@rwth-aachen.de

2.2 Moving in the Game World

To solve the problem of moving in the game world without access to the phone's touchscreen, we chose to make use of the tracking library supplied by the chair. It allows us to locate ourselves within a room by finding reference points within the camera image and estimating the camera pose from them. For this to work we first require a point cloud reconstruction with descriptors for each point which has to be generated from imagery of the location.
In theory we can achieve a fairly exact localisation using this technique, however we face inaccuracies due to the limited computation power of the system. To overcome them, we chose to implement a Kalman filter [Kalman 1960]. This filter is capable of not only filling in gaps in the data or pauses between measurements, but also predict a small time into the future, so that the movement is smoothed fairly well. Still, a small amount of jitter is left as further smoothing would make the tracking irresponsive.

2.3 Physics and Game Logic

As the whole game is physic driven, we choose to use the real-time physic simulation engine Bullet-Physics [BUL] to simulate our virtual environment.
To create an accurate simulation, every object within the game is represented by a rough body in the physics engine. The interaction is then handled by the engine and updates objects positions each game tick. This way we can correctly simulate gravity, bounciness and friction of objects. Additionally we are able to implement fixed connections between objects as well as hinges between objects by restricting possible movements.
Many interactions in the game world were not realisable by the Bullet engine, therefore we designed algorithms to calculate gear interaction and the transported force, e.g., using conveyor belts and also the transport of heat (e.g. lighting a cannon fuse or powering a solar panel) and electricity (e.g. switches, motors, ...). Any resulting forces are given to the physics engine to be applied to the respective objects. This way we can simulate firing a cannon or exploding a bomb triggered by heat transport.
Additionally, we implemented the possibility to have "hanging" cables which are calculated physically correct in real-time based on hyperbolic cosine curves. The whole game can be controlled (e.g., changing levels or starting and resetting the physics) with a menu (Figure 1 a)) that is integrated in the room model which can be controlled with our unique interaction system explained in the following section.

2.4 Hand-Tracking

An essential and very critical part is the hand-tracking. The whole game is based on the interaction of the players hands with the virtual environment, as such the developed hand tracking system has to recognise the position of the hand and some gestures the player performs.
The hand tracking system has to solve several problems: As the iPhone has only one backside camera, calculating a 3D position from only a single image is hard. Also it has to differentiate between one or two hands and the background which is computationally expensive. Finally it has also to detect gestures for closing and opening the hand. The developed hand tracker uses the Open Source Computer Vision (OpenCV) Framework to process the camera image to determine the hands position and gesture. The algorithm uses colour distinction to separate hands from other objects in the filmed environment. It then detects contours and defects to count the opened fingers. Finally the relative hand position is determined by the contours bounding box. To fill any gaps in the detected data and generally smooth it, we use the same filter that we already used for the position estimation.
This allows us to track multiple hands at once as long as they have a distinct colour difference to the background. For this reason we chose to wear green gloves while playing the game.

2.5 Asset Creation and Level Design

We designed the room, objects and levels to have a cartoonish character. The texture are essential to this style and have been chosen to specifically support it. This way we were able to design assets with a small number of polygons without them looking out of place. We created three levels with increasing difficulty to allow the user to easily understand and learn the game and then apply this knowledge later on. They focus mainly on introducing game mechanics and are not too hard, but in the future one could design challenging puzzles for users to solve.
The level files are directly exported from 3DS Max [3DS] to a custom xml-based file format and loaded into the game using TinyXML [Tin].

3 External Libraries and Tools

The game is implemented with the help of several external libraries and tools:

1. OpenGL ES 3.0 is the underlying graphic API. [Ope c]
2. Bullet simulates our physics. [BUL]
3. OpenAL brings the 3D Sound effects. [Ope b]
4. OpenCV helps to detect the players hands. [Ope a]
5. Eigen and CML simplifies the math behind the game. [Eig], [CML]
6. Artemis manages the zoo of entities. [Art]
7. TinyXML makes our xml-files readable. [Tin]
8. FreeImage brings textures to our objects. [Fre]
9. 3DS Max brings the ideas to life. [3DS]

References

Autodesk 3ds max.

Artemis c++ - entity system.

Bullet physics library.

Configurable math library.

Eigen math library.

Freeimage.

KALMAN, R. E. 1960. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering 82*, 1, 35–45.

Open source compute vision.

Openal - cross platform 3d audio.

Opengl es 3.0. https://www.khronos.org/opengles/.

Tinyxml 2.