4th Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2007)
J. Dingliana, F. Ganovelli (Editors)

Refraction of Water Surface Intersecting Objects in Interactive Environments

H. Cords[†]

Institute of Computer Science, University of Rostock, Germany

Abstract

This paper presents a rapid method to render dynamic water surfaces with penetrating obstacles in real-time. Taking the surface boundary into account, our method allows the rendering of single reflections and single refractions of objects even intersecting the water surface, including a physically approximative perspective refraction mapping. Thereby, water surfaces are represented as 2.5D height fields and obstacles as polygonal objects. In principle, we determine approximating virtual reflection and refraction eye coordinates. With respect to the water surface, the reflected and refracted objects and parts of objects are projected onto the surface from separate, virtual eye coordinates. Since we are using per-pixel reflection and refraction mapping, our multi-pass, image-based technique is suitable for GPU-based implementations. Moreover, we demonstrate the interactive application of the method for height field based data sets extracted from interactive 3D Smoothed Particle Hydrodynamics (SPH) simulations in real-time. Thereby, the presented approach achieves high frame rates and plausible results.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

Real-time rendering of dynamic water surfaces requires simplifications of complex physical effects like reflections, refractions and caustics. Refraction is a fundamental visual feature if a refracting material can be seen at varying angles, e.g. a glass of water. Therefore, refractions should be handled by an accurate approximation in the rendering process in order to achieve convincing results.

This paper addresses the problem of polygonal *objects intersecting (or lying under) the water surface*. We focus on real-time applications and approximated reflections and refractions of these polygonal objects, since realistic and rapid reflections and refractions at strongly curved height field based surfaces are still a practically unsolved problem: Fast standard environment mapping techniques cannot handle curved surfaces with intersecting objects. Hence, we present a rapid, image-based approximation of single reflections and refractions at strongly curved, dynamic water surfaces. Due to the excellent performance of a GPU-based implementation, our method results in high framerates on consumer graphics hardware. Moreover we applied the presented technique to a physically based, interactive and rapid real-time fluid solver – based on a 3D SPH simulation, but the surface is reconstructed as a height field. Thus, a virtual pool of water can be handled interactively like e.g. a glass of water in reality (cf. fig. 12c). Beside SPH, our approach can be adapted to any other height field based water surface generation technique easily. In practise, our approach can be used to reflect and refract e.g. a pontoon, an avatar, fish, rocks in a river, or a boat intersecting water surface. Fast approximating techniques simulating those effects are favored by current video games.

The basic idea of our visualization approach is to approximate the curved surface surrounding an object by a plane. This plane is necessary for calculating the virtual reflected and refracted eye positions to generate approximated images of the refl./refr. object (Sec. 2.1). For each object, this procedure is repeated separately. Afterwards, the generated images are mapped onto the surface according to physical

† e-mail: hilko.cords@uni-rostock.de

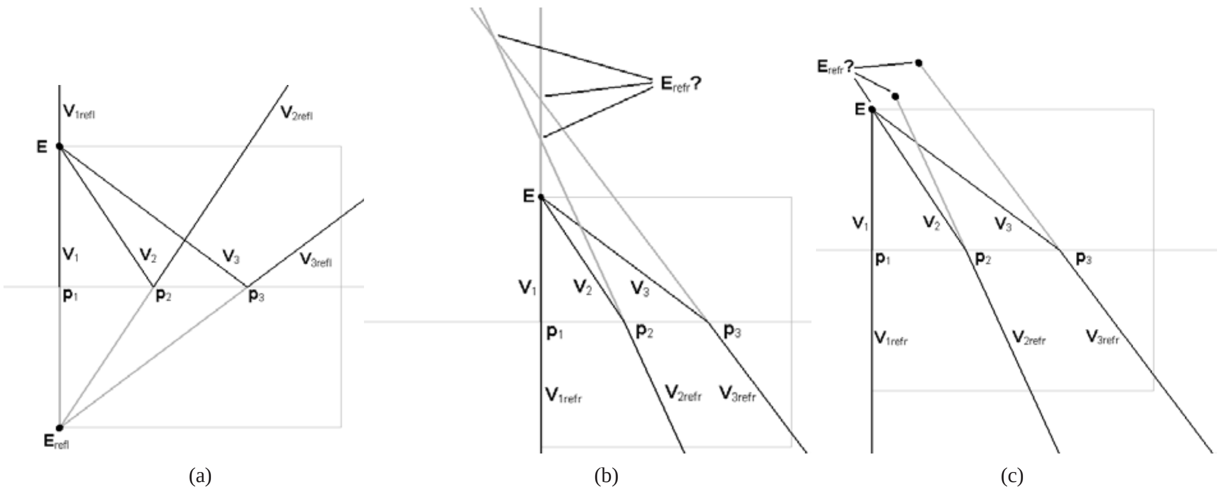


Figure 1: *Principle of planar reflections and refractions: A planar, 100% specular reflecting mirror is seen from eye coordinates* $\mathbf{E}$ *(a). Three example viewing rays* $\mathbf{v}_1$, $\mathbf{v}_2$ *and* $\mathbf{v}_3$ *are reflected. All reflected rays focus at the virtual reflected eye position* $\mathbf{E}_{refl}$*. Using a refractive plane (example b,c: refraction ratio of water), no focus occurs (b). In theory, the virtual refracted camera vectors can be determined for each point seen on the plane* $\mathbf{p}_1$, $\mathbf{p}_2$ *and* $\mathbf{p}_3$*. Therefore, the distances between* $\mathbf{E}$ *and* $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ *are equated to the distances between* $\mathbf{E}_{refr}$ *and* $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ *(c).*

refl./refr. of the real, curved surface (Sec. 2.2). Hence, refraction properties like virtual angle shift or virtual scaling of objects intersecting the water surface can be simulated. Those phenomena can not be approximated accurately at high framerates with the techniques used e.g. in today's computer games. Particular attention is paid to objects which are intersecting the water surface, with the ambition to minimize artifacts at the visual transition at the surface-layer between object parts lying above and under water surface. Our approach achieves convincing results at reasonable framerates for curved surfaces, as shown for surfaces resulting from real-time SPH simulations (Sec. 3 and 4).

1.1. Related Work

Due to the complexity of interaction between light and water, an exact simulation of these physical effects (e.g. raytracing) is not affordable for real-time applications. In fact, the simulation and rendering time for even approximated photo-realistic water lies beyond real-time [CMT04] [HK05]. Thus, a lot of work has been done in the field of computer graphics simplifying or approximating the physical behavior of liquids (e.g. [Sta99], [MCG03], [Har05]), as well as the physically based visual appearance.

Specular reflections or single refractions can be approximated by using static or dynamic environment maps. The scene is rendered into the environment map from the view of the reflecting and refracting object. The map generated that way is accessed according to view direction and surface orientation during the final rendering pass, usually GPU-driven [NC02] [RKL*06]. This technique works well for small objects, but for large objects with flat or close to flat surface areas (e.g. water surfaces) the environment map approach suffers. This problem can be solved for specular *reflections* with an algorithm determining projected reflection vertices [RH06] [EMDT06], which allows accurate reflections even for intersecting objects. *Planar single reflections* can be realized with a second rendering pass: The scene is rendered from the mirrored viewpoint into the reflecting plane [BMG*99]. *Planar refractions* can be approximated in a similar way, but are more complex in practise. The scene in refraction space is distorted into image space. Thus a transition function has to be calculated, which transforms the refracted scene into the image space [DB94]. Image-based double refraction (meaning refraction through two surface borders) using an environment map can be realized with a two pass rendering approach [Wym05]: The refractive object is split into back and front faces. Restricted to static environment maps (lying at infinite distance), the approach achieves reasonable results at high frame rates.

Approaches towards the real-time rendering of *water surfaces* including reflections and refractions are usually either based on approximating raytracing techniques or environment mapping techniques. The real-time raytracing approaches include e.g. a simplified raytracing algorithm [BD06] using two height fields: The water surface and the ground surface. The intersection of refracted and reflected rays with the ground surface is calculated on the GPU for reasonable frame rates. This technique achieves pleasing results for height field based grounds, including height field humps penetrating the surface. Other GPU-based approaches towards real-time raytracing built upon some major simplifications have recently been proposed [SKALP05] [PMDS06]. However, the raytracing approaches achieve good results, but the lack of high framerates does not allow the use e.g. in actual computer games.

The environment mapping approaches include a fast technique used in today's computer games: The scene parts are drawn into an environment map from eye coordinates, which is accessed during the surface rendering pass with an offset depending on the surface height and surface normal to simulate the typical wobbling behavior of reflecting and refracting water surfaces [Sou05]. This empiric approach achieves good visual results and performances but cannot represent the typical refraction effects like angle shifts or object scaling. Another approach generates the environment maps depending on a global planar approximation of the water surface [Bel03], visualizing lakes and oceans: The refraction map is generated from camera view. This approach achieves good results for slightly curved surfaces, but cannot handle the perspective shift of objects lying under water or the occurrence of strongly curved waves. In [SW01] a GPU-based approach is presented, using a noise function to generate the surface. This method includes standard environment mapping for approximating sky reflections and ground refractions, but cannot handle objects intersecting the surface.

2. Our Approach

In this work, we solve the following problems of currently used techniques (see previous section) for rapid rendering of reflections and, particularly, refractions of 2.5D water surfaces: Highly curved surfaces, objects intersecting the surface, objects lying underwater. Except for special scenarios (e.g. splashing, breaking waves), most water surfaces are 2.5D. Thus, choosing height fields for the surface representation is an intuitive approach. Moreover, we observe that except for small wavelengths, the surface of water surrounding an object can be approximated by a plane (Fig. 6 top). Hence, this plane is used to determine reflective and refractive views of this object. We demonstrate that these planar approximations can be mapped neatly onto the surface with respect to the laws of reflection and refraction, resulting in plausible optical properties. The basic steps of our method are the following:

1. A surface approximative plane around the handled object is generated.
2. This plane is used to determine the refl./refr. virtual eye coordinates.
3. These virtual eye coordinates are used to generate the refl./refr. environment map.
4. During surface rendering, this env. map is accessed using refl./refr. vectors of the slightly modified surface.

The virtual eye coordinates are used to generate an approximating perspective refl./refr. map. In detail, these vectors are introduced in section 2.1. The final intensity ratio of reflection and refraction is calculated by the Fresnel equations.

It must be pointed out that each refl./refr. object is handled separately. Objects intersecting the surface should be split into two parts: One part lying above surface and one part lying under surface. We assume that reflected and refracted rays do not intersect the surface again. Hence, single reflection and refraction is sufficient. In the majority of cases, this simplification is valid. Thus, just the object parts lying above surface can be reflected and just the object parts lying under surface can be refracted.

In detail, we describe the basic *planar* refl./refr. in section 2.1 and the extension to *curved water surfaces* in section 2.2. Furthermore, we demonstrate an approximation technique, to reduce the visual artifacts occurring at surface penetrating objects and multiple objects fairly (Sec. 2.3), resulting in a good tradeoff between performance and realism. A simple and rough, but fast caustic approach is presented in section 2.4. The final composition is described in section 2.5.

2.1. Basic Approach

We start with *planar reflections*. For planar refractions, we introduce our idea afterwards:

Single planar reflections are simple, due to the virtual focus of reflected view rays (Fig. 1a): Using the plane normal $\mathbf{n}_{plane}$, a point $\mathbf{p}_{plane}$ lying on the plane (e.g. $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$), the camera position $\mathbf{E}$ and the scalar product $\langle \mathbf{a}, \mathbf{b} \rangle$, the *virtual reflected eye position* $\mathbf{E}_{refl}$ can be determined:

$$\mathbf{E}_{refl} = \mathbf{E} - 2 \cdot \langle \mathbf{E} - \mathbf{p}_{plane}, \mathbf{n}_{plane} \rangle\, \mathbf{n}_{plane}. \qquad (1)$$

Hence, the scene lying above the plane can easily be rendered from virtual eye vector $\mathbf{E}_{refl}$ and mapped onto the plane, according to a physically correct planar mirror. As you may see, the distance between eye vector $\mathbf{E}$ and reflection position $\mathbf{p}_n$ ($n = 1, 2, 3, \ldots$) equals the distance between virtual eye vector $\mathbf{E}_{refl}$ and the corresponding reflection position $\mathbf{p}_n$. These equal distances are important for correct perspective mapping and the correct distance between eye vector and any virtual reflected point.

The difficulty of *planar refractions* depends on one major difference with respect to reflections, making the real-time simulation of refractions more difficult: A focus of refracted rays does not exist as shown for reflection rays. Fig. 1b shows the three *virtual refracted eye vectors* $\mathbf{E}_{refr}$, resulting from intersection of refracted viewing rays $\mathbf{v}_{1refr}$, $\mathbf{v}_{2refr}$ and $\mathbf{v}_{3refr}$. Principally, the refracted rays do not focus. Furthermore, figure 1c exemplifies the calculated virtual camera vectors for different refracted viewing rays – with equal distance between eye vector $\mathbf{E}$ and refraction position $\mathbf{p}_n$

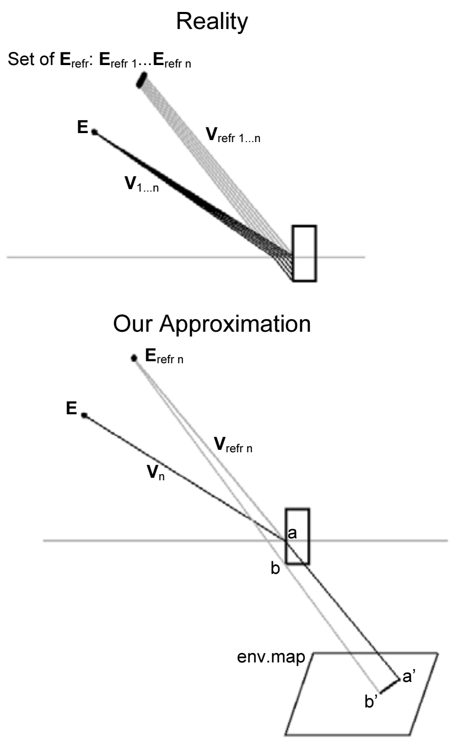


Figure 2: *The basic idea of planar refractions: The set of virtual refracted eye vectors* $\mathbf{E}_{refr\ 1...n}$ *(top) is reduced to just one vector* $\mathbf{E}_{refr\ n}$ *(bottom). At the best, today's computer games uses* **E** *for a rough approximation of refraction (also see figure 10).*

($n = 1,2,3,\ldots$) and virtual eye vectors $\mathbf{E}_{refr}$ and the same refraction position $\mathbf{p}_n$.

The basic idea of our planar refraction method compared to reality or e.g. raytracing is shown in figure 2: The set of virtual eye coordinates per object $\mathbf{E}_{refr\ 1...n}$ (Fig. 2 top) in reality is simplified to one virtual eye position $\mathbf{E}_{refr\ n}$ per object (Fig. 2 bottom). $\mathbf{E}_{refr\ n}$ is the virtual eye position, calculated with the refraction vector $\mathbf{v}_{refr\ n}$ of $\mathbf{v}_n$ refracted at **a**:

$$\mathbf{E}_{refr\ n} = \mathbf{a} - \|\mathbf{E}-\mathbf{a}\| \frac{\mathbf{v}_{refr\ n}}{\|\mathbf{v}_{refr\ n}\|}. \qquad (2)$$

In 3D, **a** is the closest point to **E**, lying on the plane and on the object. Consequently, according to **a**, $\mathbf{E}_{refr\ n}$ matches the reality. In practise, **a** is determined by the cut of an object's bounding box and the straight line between **E** projected on the plane and the cut of the object's principal axis and the plane. $\mathbf{v}_{refr\ n}$ is determined by Snellius' law with respect to the ratio of refraction indices η and view direction **I**:

$$k = 1 - \eta^2(1 - \langle \mathbf{I},\mathbf{n}\rangle^2)$$
$$\mathbf{R}_{refr} = \begin{cases} \eta\mathbf{I} - (\eta\langle\mathbf{I},\mathbf{n}\rangle + \sqrt{k})\,\mathbf{n} & : \quad k \geq 0 \\ 0 & : \quad k < 0. \end{cases} \qquad (3)$$

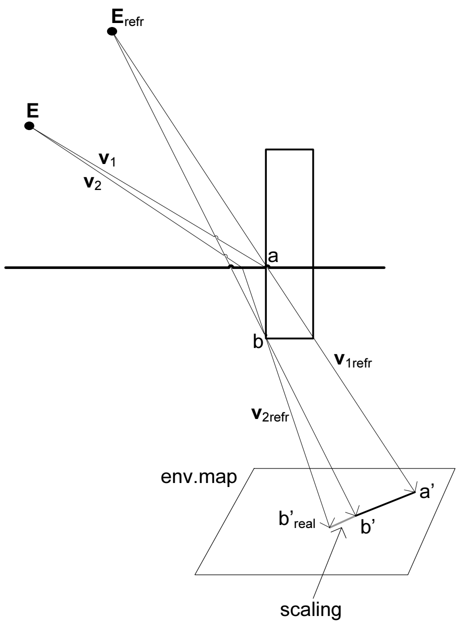


Figure 3: *Fixing the scaling problem.*

Objects intersecting the surface have to be split into two parts to prevent refraction of object parts lying above the water surface: A multi-pass per-pixel culling (stencil- and depth-buffer based) results in less artifacts but is also reducing the framerate. Thus, we use the approximating planes described above for hardware accelerated single-pass plane culling on a per-object basis. Usually, the resulting artifacts are minor. The object parts lying under water surface are rendered into an environment map, using this eye position $\mathbf{E}_{refr\ n}$. This environment map is accessed during rendering pass by a shader program: According to surface position and normal, the correct per pixel reflection and refraction vectors are calculated and used for environment map look-up. Hence, the point **a** is seen at the same position as in reality, if the environment map is accessed at **a** with the associated refraction vector $\mathbf{v}_{refr\ n}$. However, the described procedure leads to an error in the virtual size of refracted object parts (Fig. 3): Physically, viewing rays $\mathbf{v}_1$ and $\mathbf{v}_2$ result in refraction rays $\mathbf{v}_{1refr}$ and $\mathbf{v}_{2refr}$. Hence, object points a and b would be seen at a' and b'_{real}. Our method projects a correctly onto a', but b would be projected onto b'. This would result in an inadequate size of refracted object parts. A scaling of the rendered underwater object parts adjusts the object size according to reality. Therefore, we suggest a linear scaling around a', such as b' is scaled onto b'_{real}. Technically, we determine the scale factor of the main axis of the bounding box and interpolate linearly. Thus, the underwater parts of objects get about the same virtual length as refracted in reality.

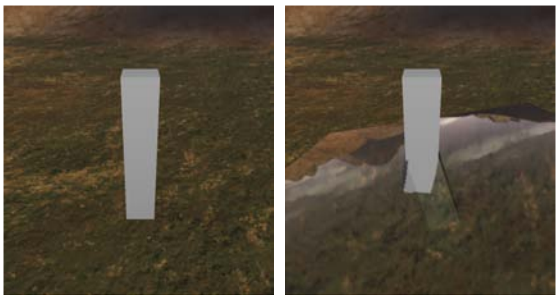


Figure 4: *An obstacle is reflected and refracted on a planar surface. Take notice of the perspective change of view for reflected and refracted object parts: The left and right side of the cuboid can be seen in the refracted and reflected images, although these polygons are directly invisible from actual eye coordinates.*

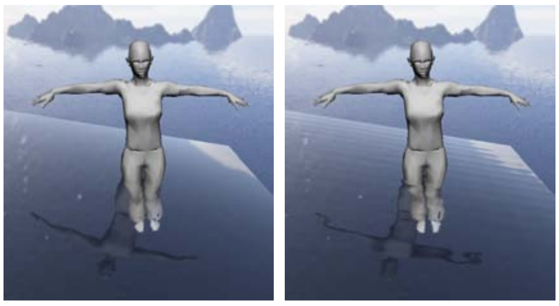


Figure 5: *A woman (8690 polygons) is reflected and refracted by a plane (a) and an artificial surface (b). The surface is generated by superposition of dynamic wave functions.*

Summarizing our refractive plane approximation b is seen in reality from view position $\mathbf{E}_{refr\ 1}$ – with our approach from view position $\mathbf{E}_{refr\ n}$. The corresponding error occurs for object parts between a and b: In reality, the perspective would change. Our approach interpolates this area linearly, seen from viewpoint $\mathbf{E}_{refr\ n}$. However, this imprecision can be accepted, because it is usually just small. An example of the presented planar reflections and refractions is given in figure 4 and 5a.

2.2. Extensions

The described algorithm for planar refl./refr. can be expanded to refl./refr. of curved water surfaces. We assume a height field, describing the surface and a polygonal object intersecting the surface. We approximate the surface surrounding area of an object with a plane (Fig. 6 top). The plane is

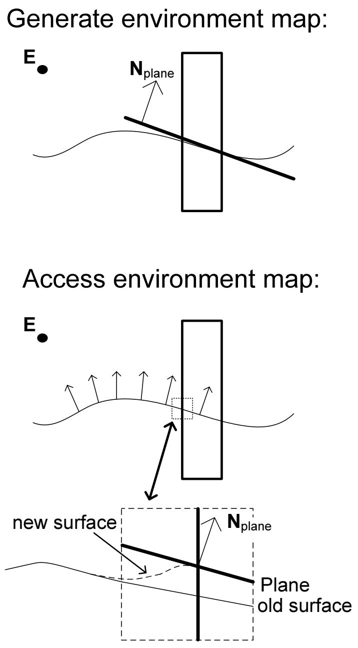


Figure 6: *Generation of and access to environment maps.*

determined by the average height information and normals of the surrounding local surface. This plane is used to generate a dynamic environment map according to Sec. 2.1, using the virtual refl./refr. eye vectors.

Later, this map is accessed within a shader, using the real surface positions and normals (Fig. 6 bottom). These can be calculated according to the law of reflection and Snellius' law (Eq. 3), using the eye position $\mathbf{E}$, surface positions and normals. In general, a water surface is highly dynamic – the environment map has to be recalculated every frame. A static one can only be used in special scenarios with sparse wave dynamics.

To guarantee an appropriate visual transition at the surface for objects lying partly above and partly under water, we introduce a linear interpolation method using normals and surface positions (Fig. 6 bottom): The visual transition depends on the accuracy of the planar approximation of the surface. Therefore, the surface reflection and refraction rays (eye coordinates) have to fit the reflection and refraction rays determined by the plane. Thus, the surface close to the object is adapted linearly, in a way that it approximates the plane in the surroundings of the object. The normals in this region are also interpolated linearly, to be congruent with the normal $\mathbf{N}_{plane}$. Hence, the visual transition of reflected and refracted objects seen on and through the water, respectively, to the parts lying above the water surface is guaranteed – even for highly curved surfaces (Fig. 10).

Recapitulating the idea, the algorithm outline is the following:

```
E ← eye position

// environment map generation
for each object i:
    if (obj. i intersects surface)
        split obj. i according to surf.
    $\mathbf{p}_i$ ← planar approx. of surface (surrounding the object)
    $\mathbf{E}_{refl\ i}$ ← determine virtual refl. eye coords ($\mathbf{E}$, $\mathbf{p}_i$)
    $\mathbf{E}_{refr\ i}$ ← determine virtual refr. eye coords ($\mathbf{E}$, $\mathbf{p}_i$)
    env. map ← render obj./obj. parts above water (view $\mathbf{E}_{refl\ i}$)
    env. map ← render obj./obj. parts under water (view $\mathbf{E}_{refr\ i}$)

// scene rendering
modify surface (surrounding each obj. i) according to planes $\mathbf{p}_i$
render water surface with refl./refr. shader (view $\mathbf{E}$)
                    ↓
            - per surface pixel:
                determine refl. ray $\mathbf{v}_{refl}$
                determine refr. ray $\mathbf{v}_{refr}$
            - final color = fresnel (env.map($\mathbf{v}_{refl}$),
                                     env.map($\mathbf{v}_{refr}$))
```

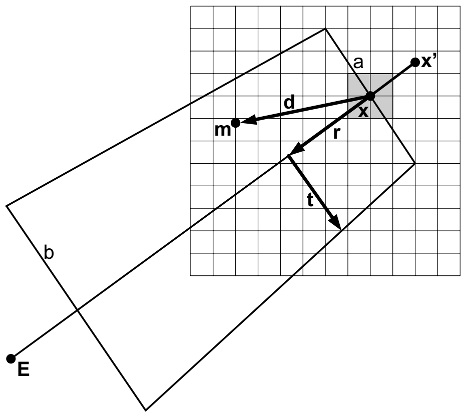


Figure 8: *Calculation of the trapezium, to determine which environment map to use. The trapezium is laid around an obstacle (grey area) reverse to the view direction.*

2.3. Multiple Objects

In principle, the described method can be used directly to reflect and refract several polygonal objects (Fig. 7). All objects are rendered successively into the environment map, using different virtual refraction eye coordinates. As a result, one environment map stores the refraction information for all objects – with their own refraction perspective.

Due to the image-based approach, in case of high wave amplitudes or objects nearby, visibility mismatches can occur. The reason are strong normal-gradients, which lead to major reflection and refraction vectors changes. E.g. an object in the environment map can hide another one, because of different refraction directions. We therefore suggest to combine close objects for refractions. In general, these effects occur just in highly curved and dynamic situations. During animation, these unwanted artifacts are hardly noticeable.

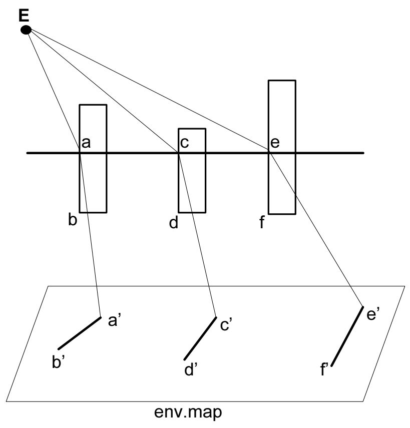


Figure 7: *Principle of environment map generation for several refracted objects.*

However, in the following we present a method to reduce at least visibility mismatches behind or next to objects. E.g. a wave behind an object can lead to an environment map access, returning refraction data of the same object. In general, this is physically not valid. Therefore, we use two different environment maps. A static map contains the full static environment data, but no data of refracted, polygonal objects. A dynamic map is used in the inverse areas, similar to the static one, but including the dynamic refracted objects. This map is generated with the approach described in Sec. 2.1.

The main area of object refractions obviously lies between the object and the view point. We use a trapezium to blend between both environment maps (Fig. 8). The parameters a and b can be used by the animator for fine tuning.

The trapezium is aligned from object center to viewpoint $\mathbf{E}$ projected on the surface. The decision which environment map to use depends on the position $\mathbf{m}$ relative to an object with position $\mathbf{x}$ (Fig. 8).

An environment map blending depending on the distance between $\mathbf{m}$ and the centerline $\mathbf{x}+s\mathbf{r}$ ($s>0$) of the trapezium solves the problem of hard transitions between both environment maps. Hence, the inaccurate reflections and refractions behind or next to objects are, if not completely removed, at least reduced to an visual inconsiderable or acceptable amount. Moreover, using $\mathbf{x}'$, positioned in view direction behind the object center $\mathbf{x}$ solves the problem of strong artifacts behind and close to the collision object. Figures 12a and 12c exemplify the reflection and refraction of several objects with just one environment map – note that the table-legs (Fig. 12c) are handled as single objects.

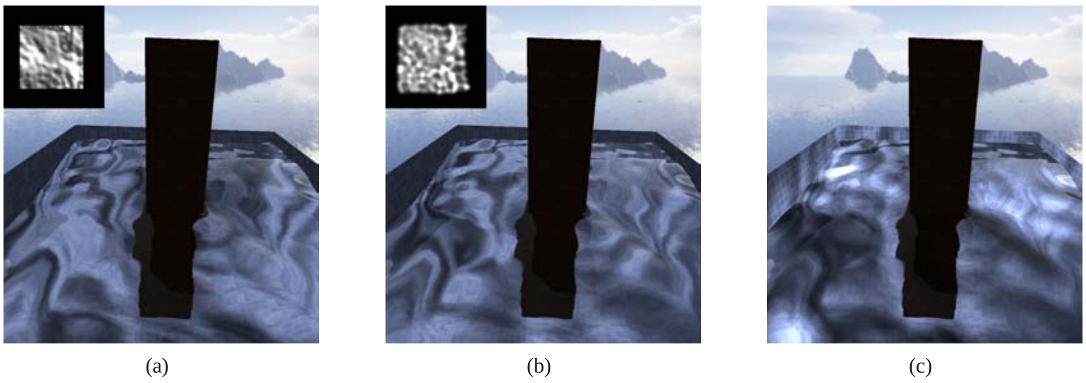


Figure 9: *A dynamic liquid flow around an obstacle. The translated and scaled surface normal field or height field can be used to approximate the intensive physical caustics (a). Physically based caustics, resulting from a light source on a planar ground (b). A superposition of both increases contrast and realism and mapping caustics onto the pool above water surface simulates light reflections (c).*

2.4. Caustics

Caustics arise from light that is focused by reflective or refractive objects, producing remarkable light effects. A water rendering system should include some caustics to enhance realism. In fact, many approaches simulating caustics at interactive rates have been proposed for special cases. For physical accurate caustics, a backward raytracing approach is recommended. Due to the complex and fast-changing composition of caustics, depending on water motion, -height and -surface, light and ground structures, the accuracy of caustics can normally not be validated by the human eye, which usually promotes strong simplification in caustics algorithms.

By taking advantage of the usual translated affinity according to the light source between the caustics on a planar ground and the water surface height and gradient field, an interpolation between the height and gradient field can approximate the caustics (Fig. 9a): The height field z-values and corresponding normals $\mathbf{n} = (n_x, n_y, n_z)$ are mapped into a caustic-representing intensity value I:

$$I = a \cdot \frac{z - z_{min}}{z_{max} - z_{min}} + b \cdot \|n_x + n_y\| + c. \tag{4}$$

The constants a,b and c are used to adjust the intensity of caustics. However, some caustics-specific light interferences are lost – in particular, the high frequencies of interferences. Being inaccurate, but with a shader based implementation virtually free of charge (Table 1), this approach can reach convincing and plausible results. In the following, we reference to this intuitive approach as "simple caustics". For comparison, we also use a second, standard texture-based approach to simulate caustics: A planar light map is generated via backward raytracing using Eq. 3, which is projected onto the scene from light view. In the following, we reference to this approach as "physically based caustics" (Fig. 9b). A superposition of both caustics approaches is shown in figure 9c.

2.5. Final composition

The intensity of reflections and refractions are determined according to the Fresnel equations, which can be stored in a texture or calculated with one of the existing approximation formulas. The composition of the intensities and their corresponding color values is very suitable to be performed on the GPU – as well as the handling of caustics, the handling of absorption or e.g. the handling of dispersion.

3. Simulation and Surface Extraction

We use a speed-optimized 3D SPH-Algorithm comparable to [MCG03] for solving the Navier-Stokes equations, using the smoothing kernels described there. The system includes collision handling as well as a fast 2.5D surface generation algorithm. With it, the simulation of 3000 particles on a single processor machine excluding refractions and caustics at frame rates above 100 Hz is possible. Another benefit of 3D SPH is the possibility of full user-interaction with a water volume. As mentioned before, the virtual water can be handled interactively like a glass of water. However, the use of a height field based rendering technique limits the rendering to curved surfaces – complex 3D water effects (e.g. splashes) cannot be visualized.

The surface is created by means of an implicit function, which is defined by the n particle positions $\mathbf{x}_i$ $(i = 1 \ldots n)$ and spherical potentials (h: iso-radius):

$$\phi(\mathbf{x}) = \sum_{i=1}^{n} \sqrt{1 - \frac{r_i^2}{h^2}} \tag{5}$$

$$r_i = \|\mathbf{x} - \mathbf{x}_i\|. \tag{6}$$

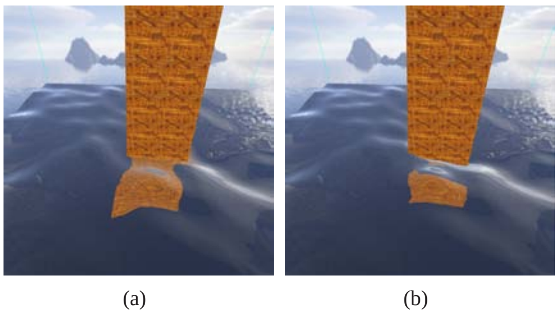


Figure 10: *Visual transition of refraction at the surface border: Our method provides an accurate transition, even at strongly curved surfaces (a). Actual techniques use* $\mathbf{E}$ *instead of* $\mathbf{E}_{refr\ n}$ *to generate the environment map (see figure 2) – the visual transition can be disrupted (b).*

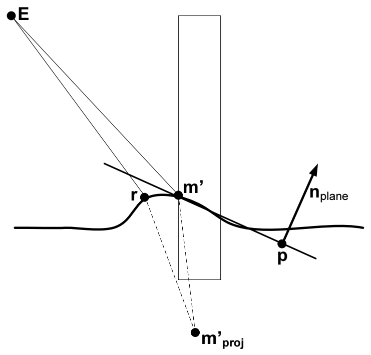


Figure 11: *Artifacts may occur at high wave amplitudes. The approximating plane refracts* $\mathbf{m'}$ *to* $\mathbf{m'}_{proj}$ *on the environment map, such as* $\mathbf{m'}$ *can also be seen at* $\mathbf{r}$*.*

The square root (Eq. 5) can be approximated to increase performance. In principle, this approach equals a volume reconstructing marching cubes algorithm, but instead of 3 dimensions, 2.5 dimensions are used: Just a height field is determined by the potential. Hence, the performance is increased significantly, compared to a full 3D surface extraction. The surface information necessary for the rendering step are the height field and the extracted normal field.

4. Implementation and Results

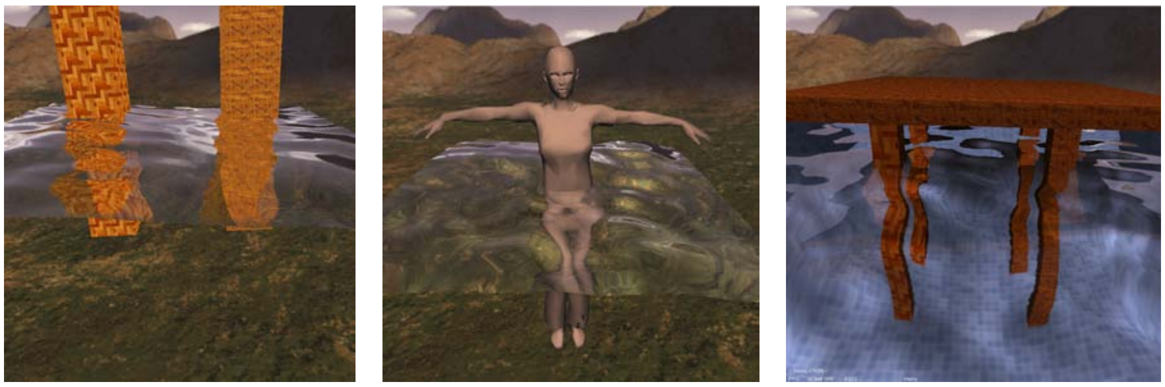
We implemented the proposed rendering algorithm as well as the simulating SPH method in C++, using OpenGL 2.0 and the related shading language GLSL. The presented examples were performed on a dual-core desktop PC with a 2,6 GHz AMD Athlon 64 CPU, 2GBs of RAM and a graphics card based on an ATI Radeon x1900 GPU. We use a parallel implementation with one CPU core simulating the physics with SPH and one CPU core extracting the surface from the particle system, creating the textures required for reflections, refractions, caustics and rendering. A simulation time step of $0.003s$ and a maximum of 3 time steps per frame were used. We use cubemaps as environment maps, due to the intuitive access with reflection and refraction vectors, but a neat use of 2D textures is possible.

The measured frame rates for the given examples are shown in table 1 (Resolution: 950×950 pixel). The most performance expensive parts are the simulation and surface extraction, due to their numerical complexity. The reflection and refraction calculations only need little CPU power. Hence, the major advantage of our algorithm is the high frame rate, being suitable for real-time environments such as VR or computer game applications. The benefit of our method lies in the application to highly curved surfaces. Even in those scenarios, the perspective accurate refraction of objects intersecting the surface is possible at high framerates. Moreover, our method provides an continuous transition at the surface border for objects intersecting the water surface (Fig. 10). These effects cannot be achieved with the existing rapid refraction techniques (Sec. 1.1) or basic environment mapping techniques. However, our algorithm is not a substitution for physical accurate off-line approaches, e.g. raytracing, even though our rough method achieves plausible and visual pleasing results in scenes with strongly curved surfaces (Fig. 12, Fig. 13).

However, the amount of simulated water is still small (usually < 5000 particles), due to the high computational costs of simulation and surface generation. But the presented rendering technique can be added to any 2.5D surface generating system, not only those ones resulting from physically based simulations. In these cases, a real-time visualization of a large amount of water is possible.

The presented refraction method approximates reality for planar and slightly curved surfaces. However, the wave structure of curved surfaces affects the physical accuracy. For example, artifacts may occur if the wavelength with a high amplitude is small compared to the width of an object (Fig. 11). Furthermore, the width of the refracted object is proportional to the physical inaccuracy: the thinner the object, the higher the accuracy. For objects with large width, this problem can be solved by the calculation of the refracted position for every point of the bounding box of an object and their corresponding projection. Artifacts may also occur due to overlapping of close objects in the environment map, or due to intensive surface variations. Despite these facts, the approximation of reality is visually convincing and in general, the artifacts for objects intersecting the surface during animation are minor.

Additionally, we demonstrated the high performance of height field based rendering of 3D SPH simulations, resulting in a rapid and strongly interactive liquid simulation (see accompanying video) - including reflections and re-



(a) (b) (c)

Figure 12: *Examples of the proposed rendering technique.*

Example	1. Refl./Refr. (FPS)	2. Simple Caustics (FPS)	3. Phys. Caust. (FPS)	4. Simulation	5. No. of CPUs
Fig. 4	91	91	-	artificial	1
Fig. 5(b)	51	51	-	artificial	1
Fig. 9	102	102	77	SPH (2000P)	2
Fig. 9	84	84	71	SPH (3000P)	2
Fig. 12(a)	75	64	64	SPH (2000P)	2
Fig. 12(b)	55	55	45	SPH (2000P)	2
Fig. 12(b)	52	52	41	SPH (3000P)	2
Fig. 12(c)	51	51	44	SPH (2000P)	2

Table 1: *Performance measurements of the presented algorithm for different scenarios. Due to the shader based approach, simple caustics are virtually free of charge (column 1 and 2). The artificial surface (line 1 and 2) is generated by simple superposition of dynamic wave functions. The framerates shown for figure 12a-c include the SPH-based liquid simulation, 2.5D surface extraction and rendering.*

fractions of collision objects. Such interactivity cannot be achieved with just surface-waves simulating techniques (e.g. [Gom00]). The presented method is one of the first fully interactive liquid simulation achieving such high framerates on a desktop PC including an approximating refraction method – however, due to the height field based rendering approach, no complex 3D liquid effects can be visualized.

5. Conclusion and Future Work

The motivation for the presented approach was to create a rapid, real-time rendering system for water surfaces resulting from interactive Computational Fluid Dynamics (CFD)-simulation, including reflections and refractions. Our new, image-based approach approximates these optical effects for polygonal objects intersecting the water surface. This is an important effect to visualize e.g. a pontoon, an avatar, or rocks in water.

Due to the high performance, the presented approach can be used in applications in the fields of VRs and computer games. We demonstrated the interactive application within a physically-based real-time SPH-Simulation on a desktop PC: To increase performance immensely, the simulated 3D water volume is rendered as a height field. Although artifacts may occur in the surroundings of surface penetrating objects, the illusion of water during animation is convincing. Our approach does not claim to be universally valid, but successfully targets the rapid reflections, refractions and caustics of highly dynamic height field based water surfaces including obstacles.

Currently, we are working on accurate refractions for objects with large widths and the automatic detection of splashing areas (with the need for a 3D representation), to solve the mentioned problems of height field based rendering. Our future investigation includes the development of rendering methods for reducing artifacts that occur at strongly curved surfaces, as well as extensions using several environment maps to tackle the problem of overlapping objects. Also, we would like to integrate further optimizations, including a fully shader-based approach to our method and the use of LoD models.

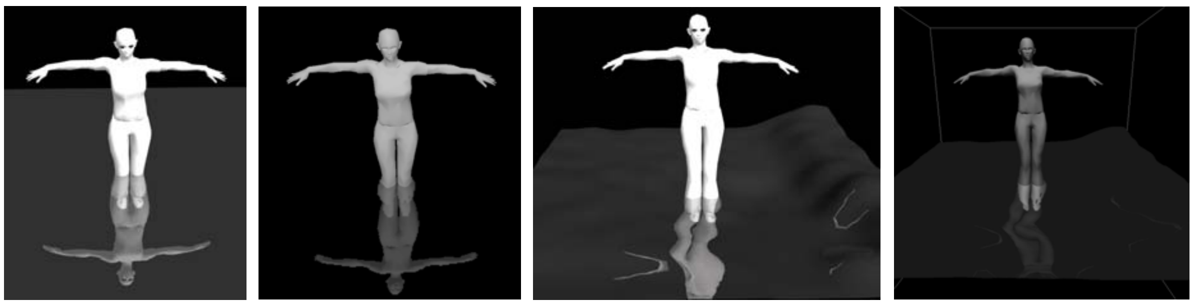


Figure 13: *Comparison of raytracing (a,c) and the presented real-time approach (b,d) (slightly different field of view). A planar water surface (left) and a curved surface (right) is shown. The presented real-time approach (b,d) is not as accurate as a raytracing approach. However, render times: Raytracing (Autodesk 3ds Max)* $\sim 4s$*, Real-Time approach:* $0,025s = 41FPS$.

References

[BD06] BABOUD L., DECORET X.: Realistic water volumes in real-time. In *Eurographics Workshop on Natural Phenomena* (2006), Eurographics.

[Bel03] BELYAEV V.: Real-time simulation of water surface. In *GraphiCon-2003* (2003), OOO "MAX Press", pp. 131–138.

[BMG*99] BLYTHE D., MCREYNOLD T., GRANTHAM B., KILGARD M., SCOTT R.: Programming with opengl: Advanced rendering. In *Computer Graphics (Siggraph-99) Conf. proceedings* (1999), Siggraph.

[CMT04] CARLSON M., MUCHA P., TURK G.: Rigid fluid: Animating the interplay between rigid bodies and fluid. In *ACM Transactions on Graphics* (2004), vol. 23, pp. 377–384.

[DB94] DIEFENBACH P. J., BADLER N. I.: Pipeline Rendering: Interactive Refractions, Reflections, and Shadows. *Displays (Special Issue on Interactive Computer Graphics) 15*, 3 (1994), 173–180.

[EMDT06] ESTALELLA P., MARTIN I., DRETTAKIS G., TOST D.: A GPU-driven Algorithm for Accurate Interactive Reflections on Curved Objects. In *Proceedings of Eurographics Symposium on Rendering* (2006), Eurographics Association, pp. 313–318.

[Gom00] GOMEZ M.: Interactive simulation of water surfaces. In *Game Programming GEMS* (2000), Charles River Media Inc., pp. 187–194.

[Har05] HARRIS M.: Fast fluid dynamics simulation on the gpu. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses* (2005), ACM Press.

[HK05] HONG J.-M., KIM C.-H.: Discontinuous fluids. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (2005), ACM Press, pp. 915–920.

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2003), Eurographics Association, pp. 154–159.

[NC02] NIELSEN K. H., CHRISTENSEN N. J.: Real-time recursive specular reflections on planar and curved surfaces using graphics hardware. In *Journal of WSCG* (2002), vol. 10, University of West Bohemia, pp. 91–98.

[PMDS06] POPESCU V., MEI C., DAUBLE J., SACKS E.: Reflected-scene impostors for realistic reflections at interactive rates. *Computer Graphics Forum 25*, 3 (Sept. 2006), 313–322.

[RH06] ROGER D., HOLZSCHUCH N.: Accurate specular reflections in real-time. *Computer Graphics Forum (Proceedings of Eurographics 2006) 25*, 3 (Sept. 2006).

[RKL*06] ROST R., KESSENICH J., LICHTENBELT B., MALAN H., WEIBLEIN M.: *OpenGL Shading Language, Second Edition*. Addison-Wesley, 2006.

[SKALP05] SZIRMAY-KALOS L., ASZÓDI B., LAZÁNYI I., PREMECZ M.: Approximate ray-tracing on the gpu with distance impostors. In *Computer Graphics Forum (Proc. of Eurographics 2005)* (2005), vol. 24.

[Sou05] SOUSA T.: Generic refraction simulation. In *GPU Gems 2* (2005), Addison-Wesley, pp. 295–305.

[Sta99] STAM J.: Stable fluids. In *Siggraph 1999, Computer Graphics Proceedings* (1999), Addison Wesley Longman, pp. 121–128.

[SW01] SCHNEIDER J., WESTERMANN R.: Towards real-time visual simulation of water surfaces. *Proceedings of the Vision Modelling and Visualization Conference* (2001), 211–218.

[Wym05] WYMAN C.: An approximate image-space approach for interactive refraction. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (2005), ACM Press, pp. 1050–1053.