# Задание 2: Симуляция водной поверхности на основе численного решения уравнения мелкой воды

## *Описание задания*

Необходимо написать программу, симулирующую поведение водной поверхности и визуализирующую результат в 3D. Цель задания - применить численное решение уравнения мелкой воды на GPU для создания реалистичной поверхности воды. Задание разрешается выполнять в трёх вариантах:

1) Численное решение реализовано на CPU. простая визуализация на OpenGL1.

**База 10 баллов.**

2) Численное решение реализовано на CPU, визуализация с освещением на шейдерах.

**База 20 баллов.**

3) Численное решение реализовано на GPU, визуализация с освещением на шейдерах.

**База 30 баллов если реализация выполнена на OpenGL3+.**

**База 40 баллов если реализация выполнена на Vulkan.**

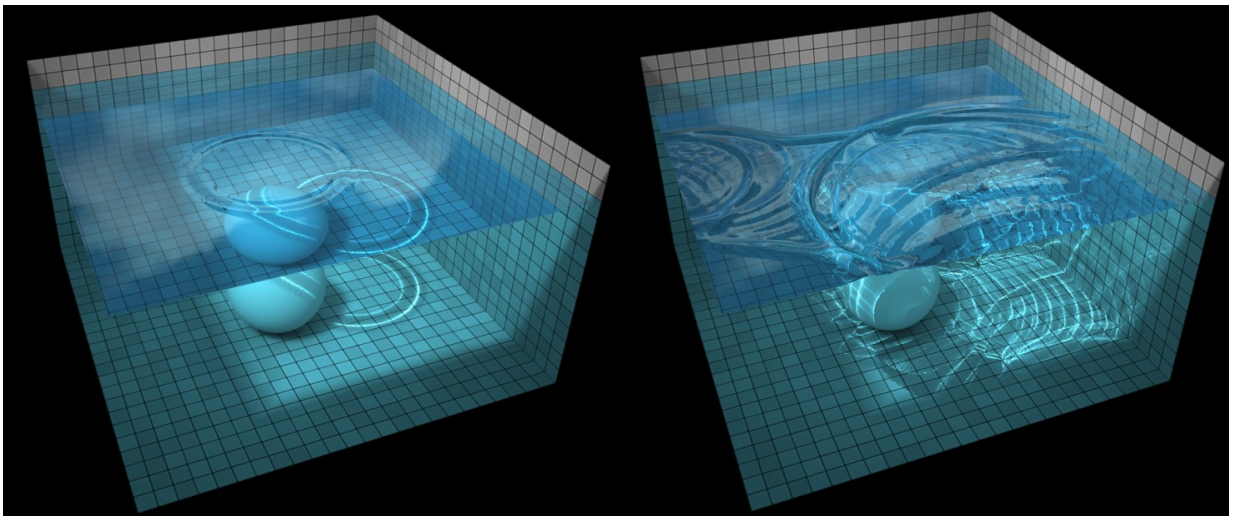

***Рис. 1. Пример выполненного задания.***

## *Обязательная часть задания*

Требуется написать программу, симулирующую и визуализирующую поведение водной поверхности при возникновении на ней малых возмущений (падение капли, движение мелких предметов).

- **По клавише “Q” необходимо включать каркасную визуализацию поверхности воды!**
- Минимум три различных объекта в сцене, под водой.
  - Пол считается за 1 отдельный объект
  - Стенки “бассейна” считаются за 1 отдельный объект
- Минимум один источник света в сцене при реализации освещения.
- Наличие пола.
- Реалистичное освещение водной поверхности (что подразумевает корректное вычисление нормали).
- Вода должна быть прозрачной.
- Управление программой должно быть реализовано как минимум следующим образом:
  - WASD - движение камеры
  - мышка - поворот сцены при зажатой левой кнопки мыши или камера как в шутерах.
  - F1, F2, F3 - смена камер если реализовано несколько фиксированных положений.
  - F5 - пауза симуляции (если реализована реалистичная визуализация на CPU)
  - F6 - реалистичная визуализация (если реализована реалистичная визуализация на CPU)
  - Правая кнопка мыши - добавление возмущающих объектов если реализовано.
  - стрелочки - управление лодкой если реализовано.

Пожалуйста соблюдайте установленные требования, **особенно в части клавиш управления**. Баллы будут снижены если требования не будут выполнены.

## Требования к симуляции

Поскольку разностные схемы предполагают фиксированный шаг по времени, необходимо обеспечить симуляцию с фиксированным шагом по времени, не зависящем от производительности машины на которой программа запущена. Простейший способ (если вы пишете симуляцию и визуализацию в 1 потоке) - включить вертикальную синхронизацию и считать, что программа всегда работает со скоростью 60 кадров в секунду. Вы также должны не забыть включить вертикальную синхронизацию в контрольной панели вашей видеокарты. На машине проверяющих вертикальная синхронизация будет гарантированно включена.

## *Дополнительная часть задания*

- Текстуры на объектах (+1).
- Отражения (при наличии окружения над водой либо при взгляде из под воды): (+2)
- Преломления (от +2 до +4) (от +4 до +8 если реализовано на GPU). Максимальный балл ставится только за корректную реализацию на основе трассировки лучей.
- Тени (от +2 до +6)
- Каустики (До +8):

  Внимание! Каустики должны зависеть от текущей формы поверхности воды. То есть должны быть следствием преломления света через рассчитанную водную поверхность. **За фейковые каустики сделанные в виде простой проективной текстуры ставится не более 1 балла!**

  - Некорректные, но правдоподобные каустики (+1)
  - Корректные каустики на дне (от +2 до +4) Корректные каустики на дне + стенах бассейна (+6)
  - Корректные каустики на произвольных объектах (от +6 до +8). Засчитывается если дно неплоское.
  - Максимальный балл за каждый из пунктов ставится только если реализована фильтрация текстуры, в которую собираются фотоны. Рекомендуется использовать медианный фильтр. Баллы за каустики различных типов не складываются.
- Окружение надводное (от +1 до +4)
  - Максимальный балл ставится только за насыщенное объектами (например ландшафт + деревья + скайбокс) окружение, реализованными с высокой степенью реалистичности.
- Корректное взаимодействие симуляции с рельефом окружающего берега (+2).
  - Волны должны корректно отражаться от всех границ берега (как от стены). Берег должен иметь нетривиальные (непрямоугольные и неугловатые) участки. Если таких участков нет, ставится только +1 балл.
- Водоросли, рыбы, ракушки и другие подводные объекты в виде отдельных моделей (от +1 до +2)
- Визуализация детальной поверхности дна (микрорельеф) - песок, камни, растительность. (от +1 до +3) (от +2 до +6 на GPU при использовании Parallax Occlusion Mapping (POM) или любой другой техники отображения детальных поверхностей - например тесселяции). Про Parallax Occlusion Mapping и трассировку лучей вы можете прочитать в задании предыдущего года. Его же вы сможете использовать для трассировки лучей при реализации преломлений.
- Визуализация возмущающих объектов - капли дождя, лодка, плавающая по воде, и. т.д (от +1 до +4).
- Реалистичность и приятный внешний вид (+1).
- Интерактивность - возможность кликать мышкой, добавляя возмущения или управлять лодкой на воде (от +1 до +2). Уточнение для случая работы с мышкой: максимум баллов ставится только если возмущающие объекты появляются в месте, на которое указывает мышка.
- Управление камерой - (от +1 до +2).

- Несколько камер (как минимум 1 под водой, минимум 3 камеры) - переключение обязательно реализовать по клавишам F1,F2,F3.... (+1).

## *Описание алгоритма симуляции*

Уравнение мелкой воды — система гиперболических дифференциальных уравнений в частных производных, которая описывает потоки под поверхностью жидкости. Уравнения получаются путём интегрирования по глубине уравнений Навье — Стокса при условии, что горизонтальный масштаб много больше вертикального. Уравнение выглядит:

$$\frac{\partial h}{\partial t} + \frac{\partial (uh)}{\partial x} + \frac{\partial (uh)}{\partial y} = 0$$

$$\frac{\partial (uh)}{\partial t} + \frac{\partial (u^2 h + 1/2 g h^2)}{\partial x} + \frac{\partial (uvh)}{\partial y} = 0$$

$$\frac{\partial (vh)}{\partial t} + \frac{\partial (uvh)}{\partial x} + \frac{\partial (v^2 h + 1/2 g h^2)}{\partial y} = 0$$

Где:

- h - значения высоты поверхности воды
- u, v - горизонтальные скорости распространения волны по x и по y
- g - константа ускорения свободного падения

При этом условии из закона неразрывности следует, что вертикальные скорости в жидкости малы, вертикальные градиенты давления близки к нулю, а горизонтальные градиенты вызываются неровностью поверхности жидкости и горизонтальные скорости одинаковы по всей глубине. При интегрировании по вертикали вертикальные скорости уходят из уравнений [wiki]. Поскольку курс компьютерной графики не включает в себя численное моделирование, мы предоставим вам готовую разностную схему которая решает уравнение мелкой воды. Однако интересующиеся могут изучить как именно получается ее вывод по следующим 2 ссылкам [1_shallow.pdf, 2_shallow_water_2010.pdf] а также иной многочисленной литературе по данной тематике.

Конечная разностная схема выглядит так:

$$H_{next}[i,j] = (1-\omega) * H_{prev} + \omega * \frac{H_{curr}[i,j+1] + H_{curr}[i,j-1] + H_{curr}[i+1,j] + H_{curr}[i-1,j]}{4}$$

***Рис 2. Разностная схема для решения уравнения мелкой воды.***

Где:

- $H_{curr}$, $H_{prev}$ и $H_{next}$ - значения высоты поверхности воды соответственно на текущем, предыдущем и следующем шагах.
- i,j - индексы ячеек двумерной регулярной сетки.
- ω - дословно "параметр релаксации". Его значение должно быть больше 1 но меньше 2. Рекомендуем присвоить омега значение стремящееся к двойке слева (например 1.985).

Для численного решения уравнения мелкой воды при помощи разностной схемы вам прежде всего вам необходимо завести 3 массива (если вы пишете симуляцию на CPU то достаточно будет только 2 массивов) - A,B,C. На каждом k-ом шаге симуляции в массиве A будут хранится значения Hprev высоты с шага k-2 симуляции, в массиве B - значения высоты $H_{curr}$ с шага k-1. На каждом шаге необходимо получать новые значения высоты - "C[i,j] = F(A,B,i,j)" при помощи разностной схемы, после чего производить циклическую перестановку ссылок - "rotate(A,B,C)".

Алгоритм симуляции будет выглядеть на псевдокоде следующим образом:

```
procedure Water_Sim() is
  type Array2D is array (positive range <>, positive range <>) of float;
  SIM_SIZE : constant integer := 256;
  A,B,C,T  : access Array2D;
begin

  A := new Array2D(1..SIM_SIZE, 1..SIM_SIZE);
  B := new Array2D(1..SIM_SIZE, 1..SIM_SIZE);
  C := new Array2D(1..SIM_SIZE, 1..SIM_SIZE);
  T := new Array2D(1..SIM_SIZE, 1..SIM_SIZE);

  while true loop

    -- simulation step
    --
    for (i,j) in [0..SIM_SIZE, 0..SIM_SIZE]:
      C[i,j] := F(A,B,i,j);
    end for;

    DrawWaterSurface(C);

    -- rotate(A,B,C)
    --
    T := A;
    A := B;
    B := C;
    C := T;
```

```
  end loop;

  -- free A,B,C,T if needed
  --

end Water_Sim;
```

Функция F должна вычислять значение Hnext высоты на k-ом шаге на основе значений k-1 ого и k-2-ого шагов **по указанной разностной схеме** (рис. 2).

## *Материалы для выполнения задания*

[1] http://google.ru
[2] 1_shallow.pdf
[3] 2_shallow_water_2010.pdf

## *Критерии оценки*

При проверки задания основной критерий качества — результат, не формальные правила. Если в реализации базовой функциональности присутствуют явные ошибки, баллы за базу могут быть снижены. То же самое относится и к реализации дополнительной функциональности. Максимальный бал ставится только если эффект выполнен на высоком уровне. Если сцена выглядит убого, программа работает слишком медленно для своей функциональности, симуляция дергается или обнаруживаются иные бросающиеся в глаза недочеты, баллы также могут быть снижены. Обязательное требование - разбираться в материале. В спорных ситуациях оценка выставляется после личной беседы, выявляющей понимание принципа действия основных алгоритмов.