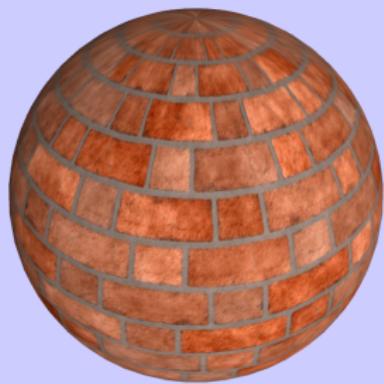


# Компьютерная графика

## Практика 10: Normal mapping, environment mapping

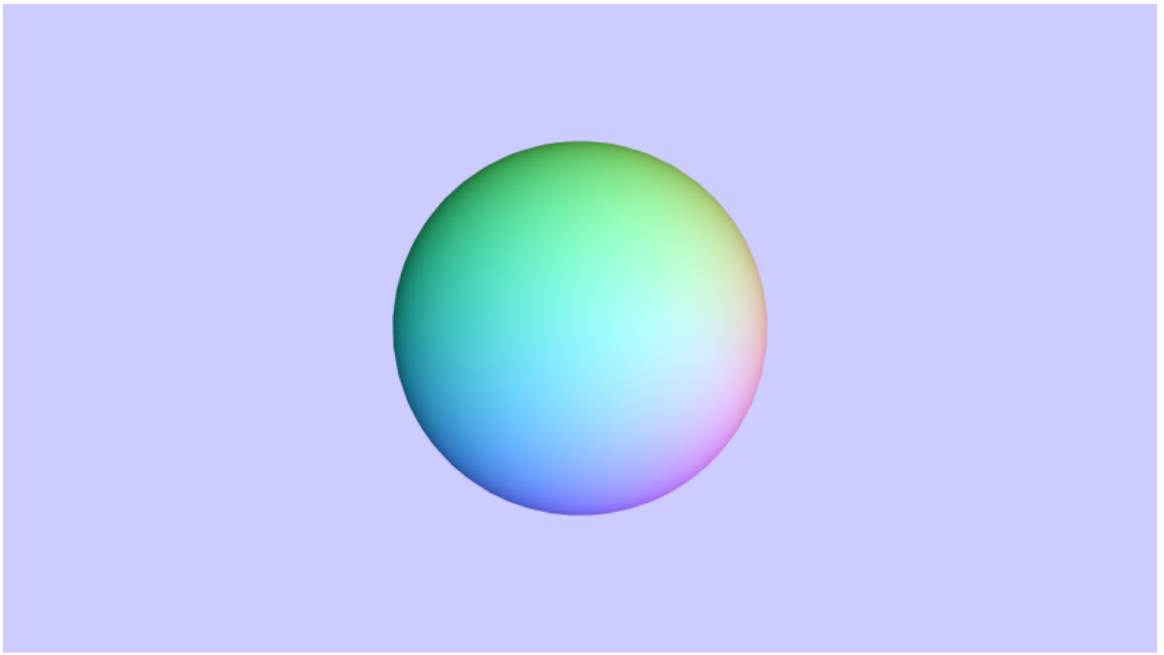
2022



## Задание 1

Готовимся к normal mapping

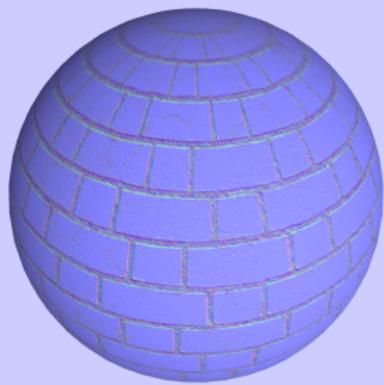
- ▶ Выводим нормаль к поверхности вместо цвета:  
 $\text{albedo} = \text{normal} * 0.5 + \text{vec3}(0.5)$



## Задание 2

Читаем normal map

- ▶ Загружаем текстуру textures/brick\_normal.jpg (в коде есть функция load\_texture)
- ▶ Заводим во фрагментном шейдере uniform-переменную sampler2D normal\_texture, перед рисованием устанавливаем её значение в 1 (аналогично albedo\_texture)
- ▶ Перед рисованием делаем эту текстуру текущей для первого texture unit'a (GL\_TEXTURE1)
- ▶ Во фрагментном шейдере читаем эту текстуру в качестве значения альбено  
`albedo = texture(normal_texture, texcoord).rgb`



## Задание 3

Вычисляем настоящие нормали

- ▶ Во фрагментном шейдере вычисляем bitangent вектор:  
`vec3 bitangent = cross(tangent, normal)`

- ▶ Вычисляем матрицу преобразования из normal map в мировые координаты:

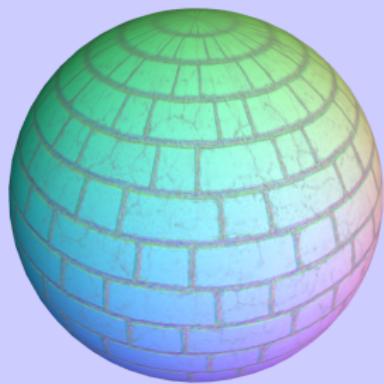
```
mat3 tbn = mat3(tangent, bitangent, normal)
```

- ▶ Прочитанное из normal map значение переводим из [0, 1] в [-1, 1] и применяем матрицу:

```
vec3 real_normal = tbn * (texture(...).xyz * 2.0  
- vec3(1.0))
```

- ▶ В качестве цвета используем новую нормаль:

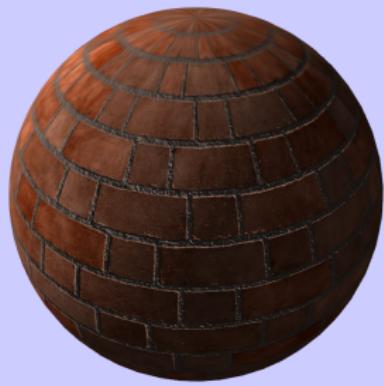
```
albedo = real_normal * 0.5 + vec3(0.5)
```



## Задание 4

Используем настоящие нормали для освещения

- ▶ Во фрагментном шейдере возвращаем старое albedo  
(бралось из albedo\_texture)
- ▶ В вычислении освещения используем `real_normal` вместо `normal`



## Задание 5

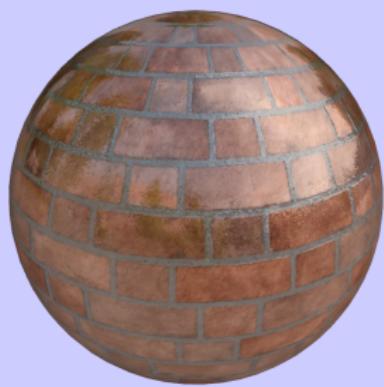
### Добавляем reflection mapping

- ▶ Читаем текстуру `textures/environment_map.jpg`, добавляем для неё uniform-переменную, устанавливаем её в значение 2, текстуру делаем текущей для 2ого texture unit'a (`GL_TEXTURE2`)
- ▶ Во фрагментном шейдере по положению камеры и положению пикселя (`position`) вычисляем направление на камеру
- ▶ С помощью нормали (`real_normal`) вычисляем направление отражённого луча
- ▶ Вычисляем текстурную координату для environment map:  
 $x = \text{atan}(\text{dir.z}, \text{dir.x}) / \text{PI} * 0.5 + 0.5;$   
 $y = -\text{atan}(\text{dir.y}, \text{length}(\text{dir.xz})) / \text{PI} + 0.5;$
- ▶ В качестве значения итогового цвета берём среднее между старым цветом (`albedo * lightness`) и значением из environment map

## Задание 5

Добавляем reflection mapping

- ▶ N.B.: если отражения слишком сильно искаются картой нормалей, можно слегка подвинуть нормаль:  
`real_normal = normalize(mix(normal, real_normal, 0.5))`



## Задание 6\*

### Добавляем environment mapping

- ▶ Добавляем новую шейдерную программу, в которой захардкожен полноэкранный прямоугольник (как в первой практике), и фиктивный VAO
- ▶ В вершинном шейдере вычисляется соответствующая координата в пространстве и передаётся во фрагментный шейдер:

```
vec4 ndc = vec4(vertex, 0.0, 1.0);
vec4 clip_space = view_projection_inverse * ndc;
position = clip_space.xyz / clip_space.w;
```

- ▶ `view_projection_inverse` – матрица  
`inverse(projection * view)`
- ▶ Во фрагментном шейдере вычисляется направление из камеры в этот пиксель
- ▶ Это направление используется для чтения из `environment map` (так же, как в предыдущем задании)
- ▶ N.B. нужно подумать, какие тут нужны `uniform`-переменные и текстуры, и правильно передать их

