

### Задание №3

**Целью работы является расчёт и визуализация динамических теней от объектов, отбрасываемых на плоскость (по желанию, также на другие объекты).**

Требуется разработать программу, демонстрирующую тот же набор из трех предметов, который был выбран для вашего варианта задания №2, с сохранением их свойств материалов. Эти предметы следует расположить на плоскости или над плоскостью. Дополнительно сами объекты и плоскость можно текстурировать.

В сцене должен присутствовать один источник освещения, меняющий местоположение, при этом требуется рассчитывать и отображать тени, которые объекты отбрасывают на плоскость, на/над которой они находятся.

Рассчитывать тени можно разными способами, однако в работе требуется реализовать алгоритм с использованием карты теней (shadow mapping).

Первый шаг алгоритма - рендеринг сцены с точки зрения источника света в текстуру, называемую картой глубины или теневой картой (Depth Map). Эта текстура глубины (карта теней) записывает информацию о глубине (расстоянии от источника света) каждого видимого объекта в сцене. Текстура глубины должна быть привязана к Framebuffer Object (FBO), который создается перед первым проходом алгоритма. Далее производится сам первый проход - отрисовка (рендеринг) сцены с использованием ортогографической или перспективной матрицы проекции, исходящей от источника света.

Затем осуществляется рендеринг основной сцены с точки зрения камеры, используя ранее созданную карту глубины для определения теневых областей. При рендеринге сцены для каждого фрагмента (пикселя) вычисляется его глубина с точки зрения камеры. Затем эта глубина сравнивается с соответствующим значением глубины из карты глубины (преобразованной в пространство камеры). Если глубина фрагмента больше, чем значение в карте глубины, фрагмент находится в тени.

В этом способе отрисовки теней также используются шейдеры: в вершинном шейдере нужно передавать позиции вершин как в пространство камеры, так и в пространство света. Фрагментный шейдер затем использует эту информацию для определения, находится ли фрагмент в тени. Дополнительно можно реализовать сглаживание, чтобы тени не выглядели резкими

Описание подхода:

<https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping>  
[Tutorial 16 : Shadow mapping \(opengl-tutorial.org\)](#)

**Пример.** В сцене заданы непрозрачная сфера и куб, на грань которого сфера будет отбрасывать тень.

Алгоритм создания карты теней (shadow\_map) и рендеринга.

**1. Расчёт вектора света:**

```
light_vx = camera[0] + 30 * np.cos(camera[3]) - light_camera[0]
```

```
light_vy = -light_camera[1]
```

```
light_vz = camera[2] + 30 * np.sin(camera[3]) - light_camera[2]
```

```
light_vector = np.asarray([light_vx, light_vy, light_vz])
```

Здесь рассчитывается направление вектора света от камеры к источнику света.

## 2. Проекция точек:

```
for model in Model._registry:
```

```
    project_points(model.shadow_points, light_camera, shadow_mod=0.01 * lenght)
```

В этом фрагменте проецируются точки модели в 2D пространство с учётом положения камеры света.

## 3. Рендеринг карты теней:

```
for model in Model._registry:
```

```
    render_shadow_map(
```

```
        model.shadow_points, model.triangles, light_camera, shadow_map
```

```
    )
```

Здесь вызывается функция **render\_shadow\_map**, которая рендерит карту теней.

## 4. Рендеринг сцены с учётом теней:

```
for model in Model._registry:
```

```
    draw_model(
```

```
        frame,
```

```
        model.points,
```

```
        model.triangles,
```

```
        camera,
```

```
        light_camera,
```

```
        z_buffer,
```

```
        model.textured,
```

```
        model.texture_uv,
```

```
        model.texture_map,
```

```
        model.texture,
```

```
        model.shadow_points,
```

```
        shadow_map,
```

```
    )
```

В функции **draw\_model** происходит рендеринг всей сцены, с учётом карты теней для определения, находится ли точка модели в тени.

## 5. Проверка теней и освещение:

```
if point[2] > shadow_map[lx][ly]:
```

```
    shade2 = min(
```

```
        0.9,
```

```
        2.5 / np.sum(
```

```
            point[2]
```

```
            > shadow_map[lx - 1 : lx + 1, ly - 1 : ly + 1]
```

```
        ),
```

```
    )
```

Этот фрагмент проверяет, находится ли точка в тени, сравнивая её глубину с соответствующим значением в карте теней.

Результаты: тень, отбрасываемая сферой на грань куба:

