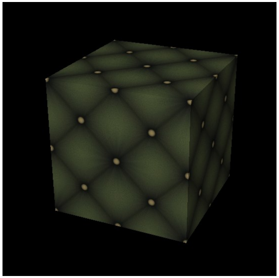
Lesson 13
Parallax Occlusion Mapping

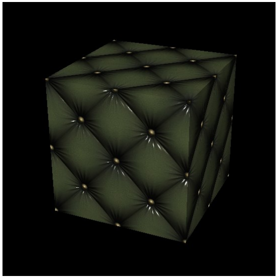
PV227 – GPU Rendering

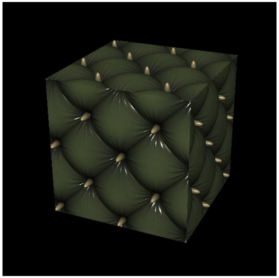
Jiří Chmelík, Jan Čejka
Fakulta informatiky Masarykovy univerzity

12. 12. 2016

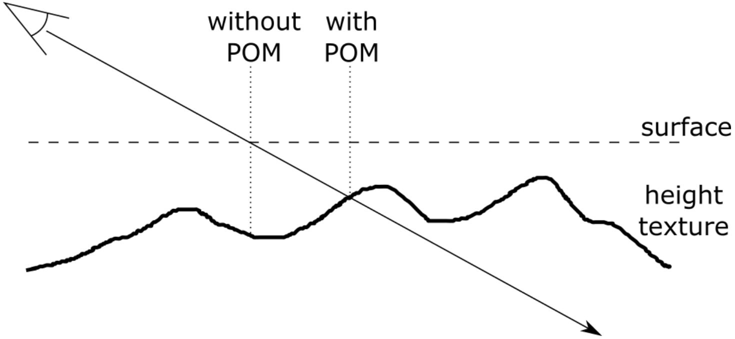
Parallax Occlusion Mapping


Nothing


Normal mapping


Parallax Occlusion Mapping

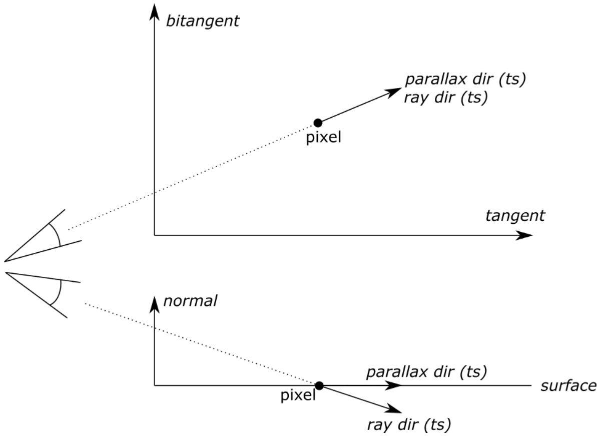
Basic principle



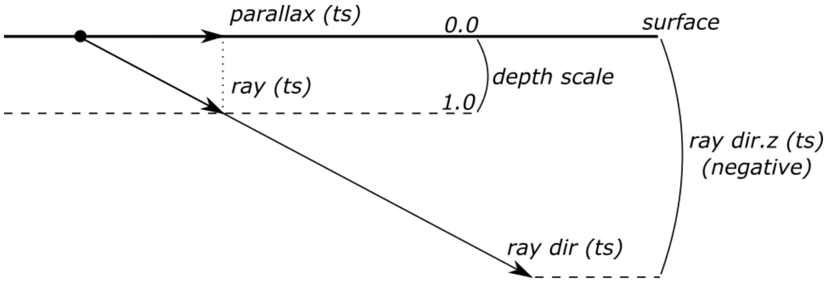
Basic principle

Parallax direction

- We work in the tangent space (ts) (on the surface of the object)
- $\textit{parallax dir}\,(ts) = \textit{ray dir}.xy\,(ts)$



Maximal parallax



$$\frac{ray\,(ts)}{depth\,scale} = \frac{ray\,dir\,(ts)}{-ray\,dir.z\,(ts)}$$

$$\frac{ray\,(ws)}{depth\,scale} = \frac{ray\,dir\,(ws)}{-ray\,dir.z\,(ts)}$$

$$parallax\,(ts) = ray.xy\,(ts)$$

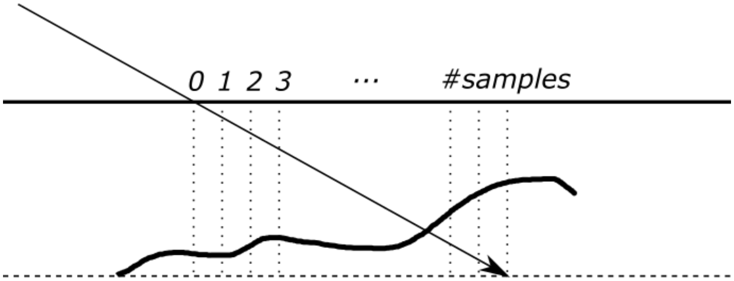
Texture space

- Our space with texture coordinates is a little different from the tangent space
- Directions of tangents and bitangents are the same as directions of *s* and *t* coordinates
- The sizes are different

$$parallax\left(texs\right) = ts\_to\_texs \cdot parallax\left(ts\right)$$

Sampling

- Sample the height texture to find the first intersectoion



Algorithm

```
for sample $i$ do
    $percentage \leftarrow i/\#samples$
    $sample\_tex\_coord \leftarrow tex\_coord_0 + parallax\_texs \cdot percentage$
    $tex\_depth \leftarrow one\_minus\_sample(height\_tex, sample\_tex\_coord)$
    $ray\_depth \leftarrow percentage$
    if $ray\_depth > tex\_depth$ then
        break
    end
end
```

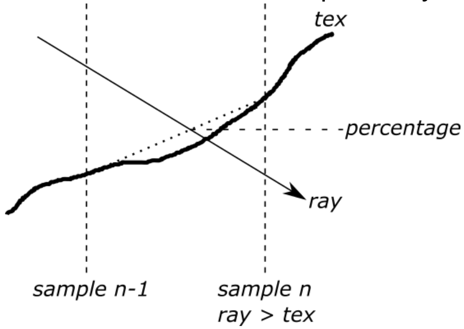
Use last *percentage* to compute the final texture coordinate and position.

Task: Implement POM

- **Task 1:** Implement this algorithm

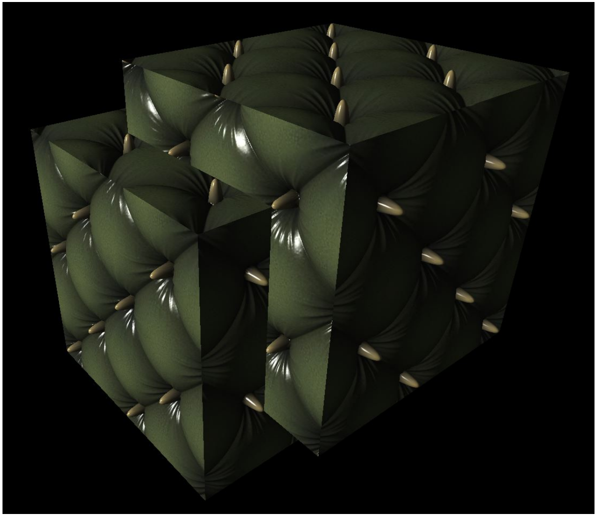
Task: Improve POM

- **Task 2:** Compute the intersection more precisely



- Result: Better result with less samples

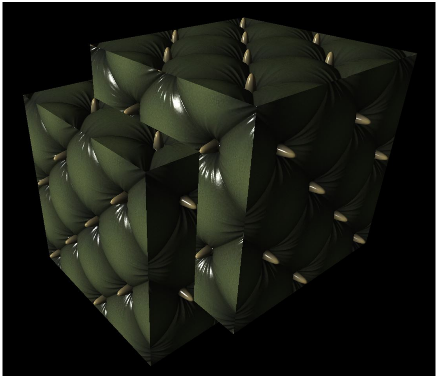
Interaction with other objects

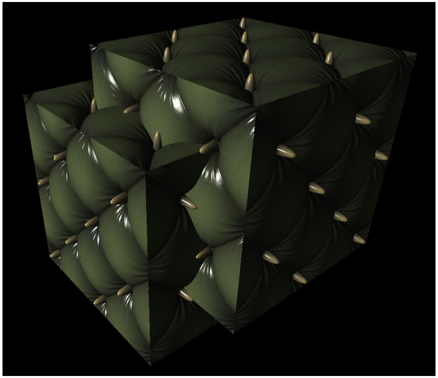


Task: Improve interaction with other objects

- **Task 3:** Adjust fragment's depth
  - Transform offseted position into clip space (transform in with *view* and *projection* matrices)
  - Transform it into normalized device coordinates (divide it with its *w*)
  - Transform it from $[-1, 1]$ to $[0, 1]$
  - Store its *z* into *gl_FragDepth*

Task: Improve interaction with other objects

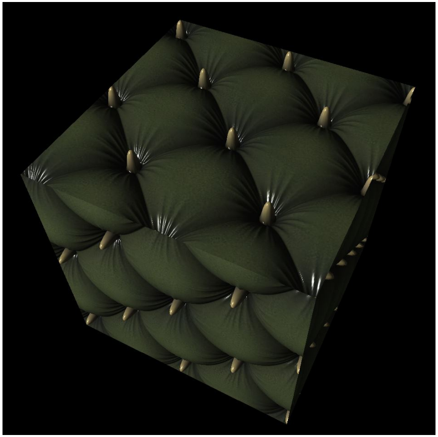

Without depth adjustment


With depth adjustment

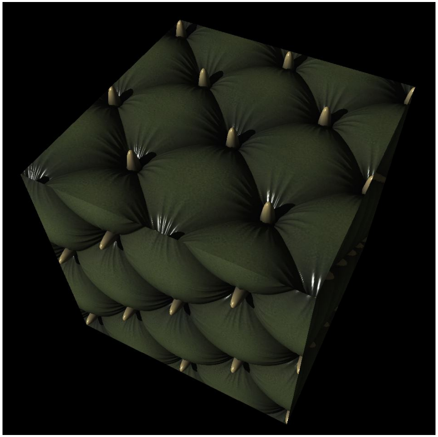
Task: Self-shadowing

- **Task 4:** Implement self-shadowing
  - Cast another ray from the offseted position to the light
  - Check whether there is an obstacle in the height map

Task: Self-shadowing



Without self-shadows



With self-shadows