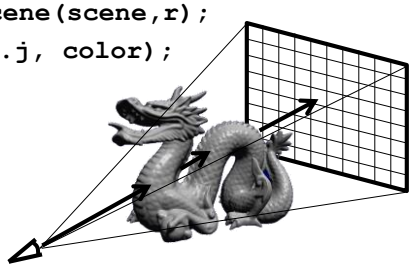
Pràctica 4: Ray tracing

Informàtica gràfica
Curs 2018-19

Gonzalo Besuievsky
IMAE - UdG

Basic Algorithm

```
action RayCasting(scene, camera)
  for each Pixel px in camera do
     r=defineRay(e,px,camera);
     color=intersectScene(scene,r);
     setPixel(px.i, px.j, color);
  end for
end action
```



See theory notes

Implementation

- Compose scene and specify camera
  - Design structure (see example)
- Compute ray generation
  - Code example given
- Implement intersection algorithms
  - TO DO
- Compute illumination
  - TO DO
- Store image
  - Use canvas2D (see example)

Structuring the Scene

- Could be specified in a file (like OBJ)
- Inserted in JS in the code
- Use objects
  - Screen: the canvas
  - Camera: center, UP, FOV, …
  - Shapes: Array of geometries
  - Sphere, Plane, Triangle, …
  - Material: Rambient, Rspecular, Rdiffuse
  - Light: position, color

Exemple

```
var Screen = {
        width     : 0,
        height    : 0,
        canvas    : null,
        context : null,
        buffer    : null,
};
```

```
var Scene = {
        Fons: [0, 0, 0],
        Shapes: [
        {
                id: "pla_groc",
                tipus       : "pla",
                normal      : [0,1,0],
                color       : [0.5,0.5,0],
                reflex      : 0.6,
                specular    : 50,
        },
        {
                id: "esfera_blava",
                tipus       : "esfera",
                radi        : 1.5,
                centre      : [-1.5,1.3,1],
                color       : [0,0.7,0.7],
                reflex      : 0.6,
                specular    : 50,
        },
        ...
        ]
```

Exemple (continue)

```
// Part of the scene
Camera: {
        position: [3,3.5,5],  // posicio camera
        up        : [0,1,0],          // vector amunt
        centre  : [-1,0.5,0],  // centre escena
        fov       : 60,                     // field of view
        X         : vec3.create(),
        Z         : vec3.create(),
        Y         : vec3.create(),
},
Lights: [
{
        position: vec3.create(), // S'emplena segons els valors entrats
        color   : vec3.create(), // S'emplena segons els valors entrats
}, ...
```

See raytracing code example

Defining a Screen

- We need somewhere to represent the imatge
  - Store the image in a file (png, tga, ppm ...)
  - Directly paint on the screen
- We are going to use the “canvas” as 2d context
  - Acces as object of HTML5

Basic object Screen and init()

```
var Screen = {
        width    : 0,
        height   : 0,
        canvas   : null,
        context  : null,
        buffer   : null,
};
```

```
// in initialize ...
Screen.canvas = document.getElementById("glcanvas");
if (Screen.canvas == null)          {
                    alert("Invalid element: " + id);
                    return;
}
Screen.context = Screen.canvas.getContext("2d");
if(Screen.context == null){
                    alert("Could not get context");
                    return;
}
Screen.width = Screen.canvas.width;
Screen.height = Screen.canvas.height;
Screen.buffer = Screen.context.createImageData(Screen.width,Screen.height);
```

Plot (x, y, color)

```
function plot(x,y,color){
        var index = (x+y*Screen.buffer.width)*4;
        Screen.buffer.data[index+0] = color[0] * 255;
        Screen.buffer.data[index+1] = color[1] * 255;
        Screen.buffer.data[index+2] = color[2] * 255;
        Screen.buffer.data[index+3] = 255;
        return index;
}
```

Geometry utilities

- Useful for computing directions, normal, reflections, etc …
- Typical vector and matrix operations
- Use gl-matrix library
  - Vec2
  - Vec3
  - Vec4
  - Mat3
  - Mat4

TODO: Intersection routines

- intersectScene (Scene, rdir, …)
  - return color of the scene
- ComputeIntersection (Scene, rdir, …)
  - Return first intersection with *id* information
- Intersect primitives
  - intersectSphere
  - intesectPlane
  - IntersectTriangle
  - Return *t* value

TODO: Illumination routines

- ComputeIllumination (…)
  - Return the color of a point
- pointInShadow (Scene, point, lightdirection, …)
  - Return TRUE or FALSE whether a point is visible or not