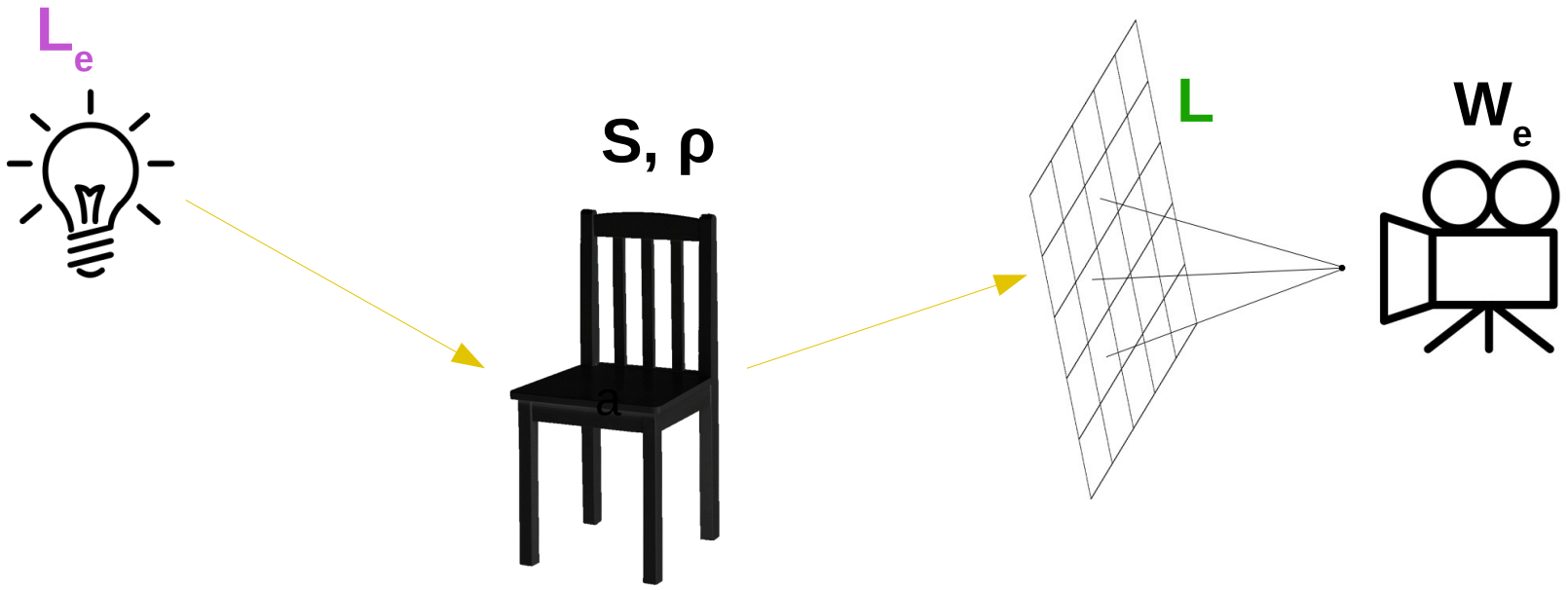
Modelovanie vstupov do renderovacej rovnice s dôrazom na svetelné zdroje

Študent: **Rastislav Kučera**
Školiteľ: **doc. Mgr. Tomáš Plachetka, Dr**

Renderovací problém

- Vstup: L_e, S, ρ, W_e
- Výstup: obrázok – hodnoty funkcie L na kamere



$$L(x,\omega) = L_e(x,\omega) + \int_\Omega \rho(x,\omega',\omega)\, L(RT(x,-\omega'),\omega')\cos\theta'\, d\omega'$$

Riešenie rendrovacieho problému

Náhodné cesty (random walks)

- Priame riešenie renderovacej rovnice, bez aproximácií
- Iteratívna metóda
- Monte-Carlo integrácia

Svetelné zdroje

- Chceme jednoducho popísať svetelné zdroje, vhodné pre priame riešenia renderovacej rovnice
- Žiadne obmedzenia
- Žiadne zjednodušenia

Svetelné zdroje - popis

- Interface schopný generovať začiatok náhodnej cesty zo svetelného zdroja
- Formát pre ukladanie svetelných zdrojov – Java serializácia

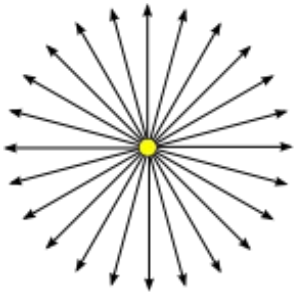
```
interface LightSource extends java.io.Serializable {
    double getPower();
    Beam getNextBeam();
    double getNumberOfBeams();
    double getRadiance(x, omega, E);
}
```

Svetelné zdroje - implementácie

```
public abstract class SpdLightSource implements LightSource {
    protected SPD spd;
    protected double beams = 0;
    protected double power = 0;

    public SpdLightSource(SPD spd, double power){…}

    public double getPower(){…}
    public double getNumberOfBeams(){…}
    public Beam getNextBeam(){}

    public void setPower(double power){…}
    public void setSpd(SPD spd){…}
}
```

Svetelné zdroje – Point light source

```
public class PointLightSource extends SpdLightSource{
  private Vec3 position;
  private Random Ax,Az;


  public PointLightSource(Vec3 pos, SPD spd, double power){
    super(spd, power);
    position = pos;
    Ax = new UniformGenerator();
    Az = new UniformGenerator();
  }


  public Beam getNextBeam(){
    Beam b = new Beam();
    b.lambda = spd.getNextLambda();
    b.origin = position;
    double angleX = Ax.nextDouble() * PI;
    double angleZ = Az.nextDouble() * 2 * PI;
    b.direction = new Vec3(angleX, angleZ);
    beams++;
    return b;
  }
}
```



Svieti **z jedného bodu rovnomerne všade do priestoru**

Svetelné zdroje – implementácie

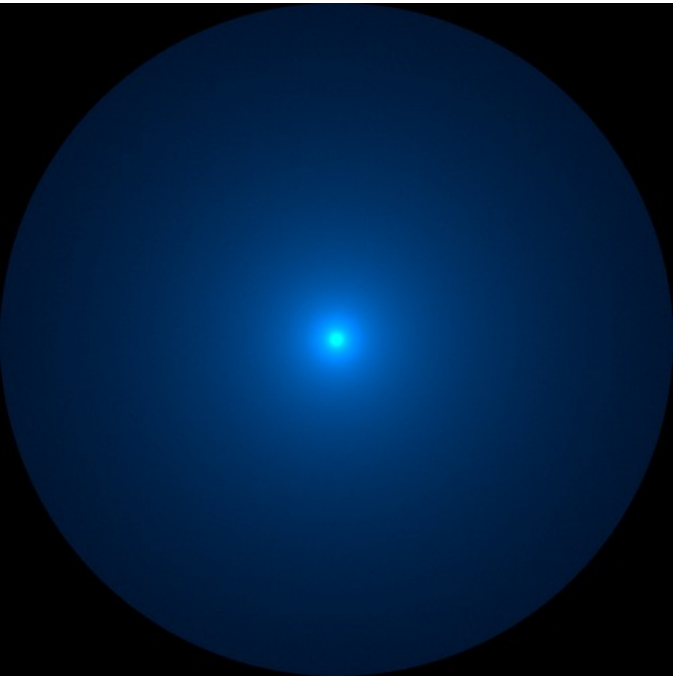
- Meníme konštruktor a funkciu getNextBeam()
- Voláme getNextBeam(), spôsob prístupu je rovnaký pre **ľubovoľný** svetelný zdroj
- Bežné formáty definujú fixnú množinu zdrojov, s každým sa pri renderovaní pracuje inak

```
// Vytvorenie
LightSource ls = new ArbitraryLightSource(…)
...
// Použitie pri renderovaní
ls.getNextBeam()
```

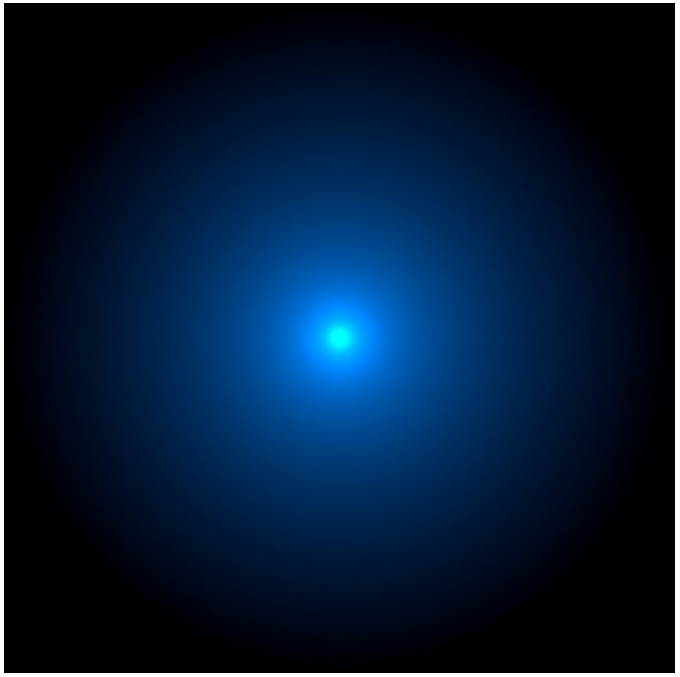
```
// Vytvorenie
PointLightSource pls = new PointLightSource(…)
...
// Použitie pri renderovaní
pls.getColor()
pls.getPosition()


LaserLightSource lls = new LaserLightSource(…)
...
lls.getColor()
lls.getPosition()
lls.GetDirection()
```

Svetelné zdroje - implementácie

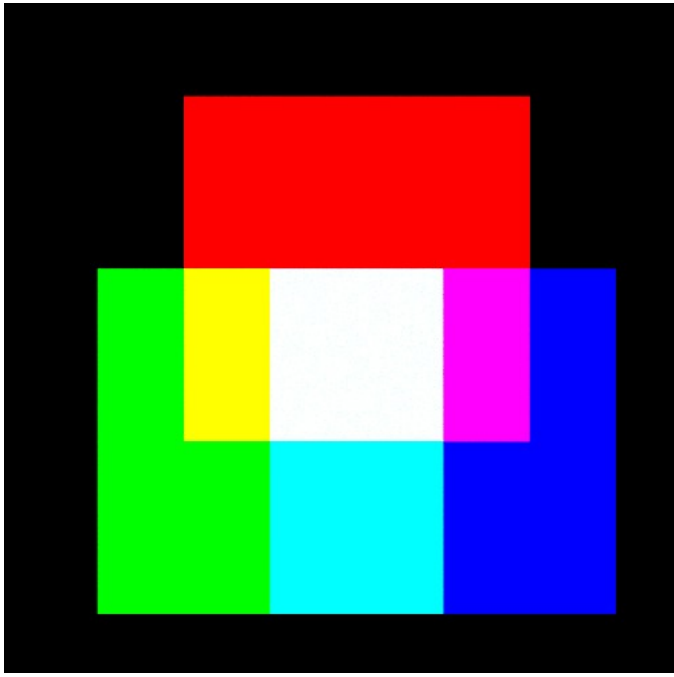
- Implementované: *Point*, *Circle*, *Laser*, *Sky*, *Spot light*, *Fading spot light*, *Composite Light source*



Spot light



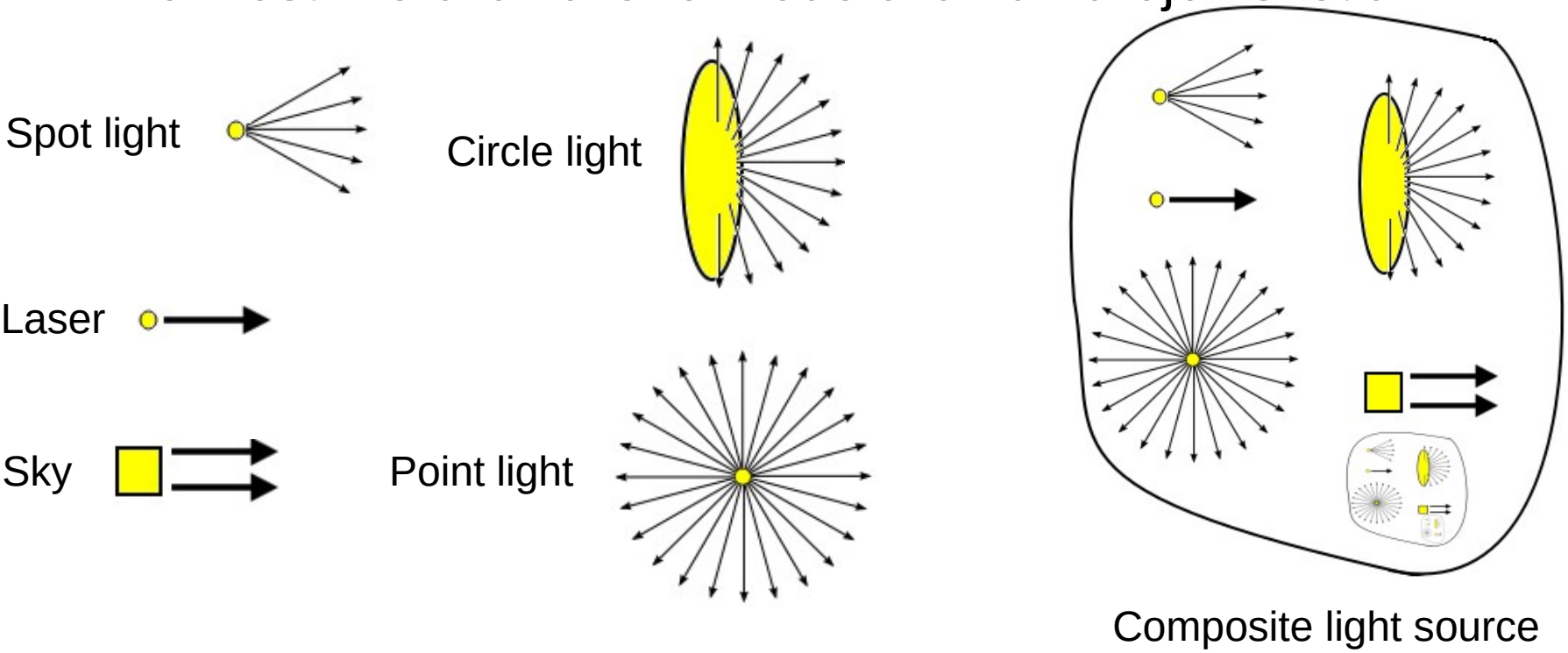
Fading spot light



Composite light source

Svetelné zdroje – Virtuálny svetelný zdroj

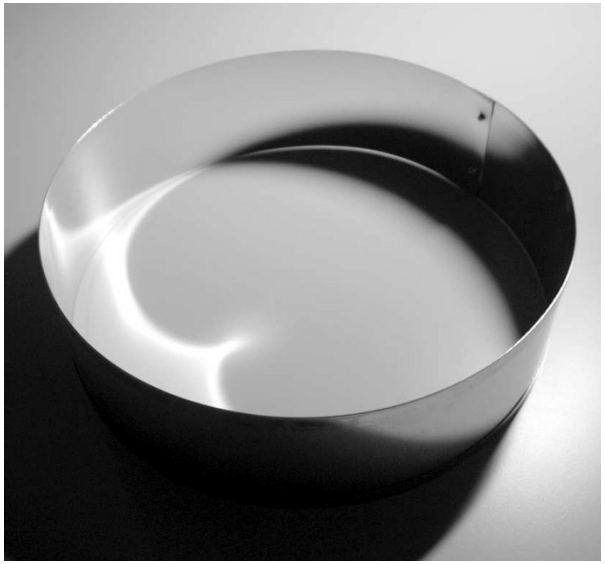
- Kombinuje viacero ľubovoľných svetelných zdrojov
- Všetky zdroje na vstupe reprezentované jedinou funkciou L_e , renderer pristupuje k jedinému objektu
- Možnosť hierarchického modelovania zdrojov svetla

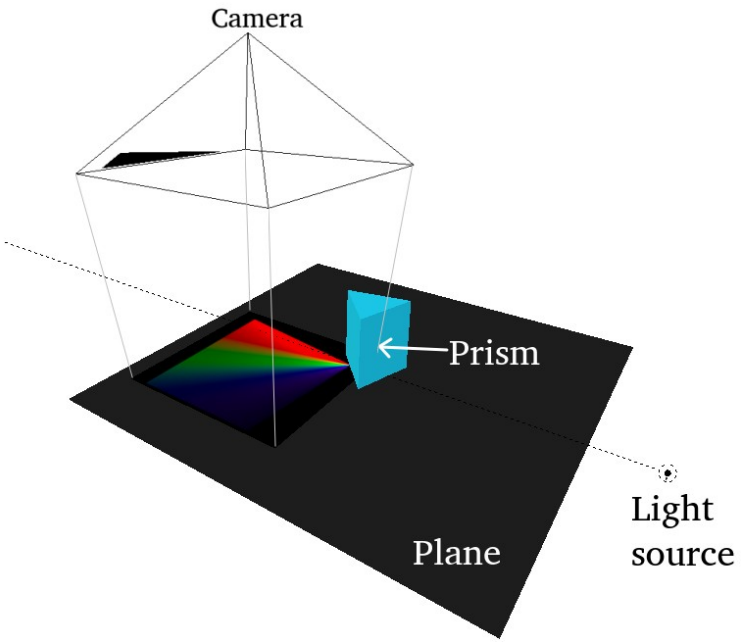


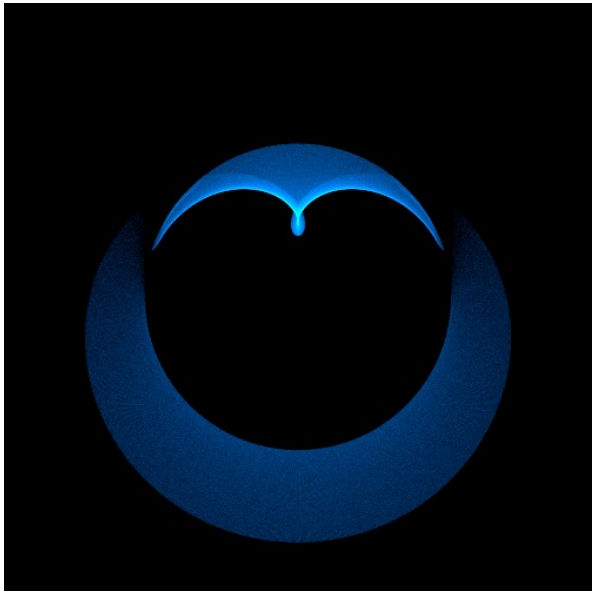
Svetelné zdroje - experimenty

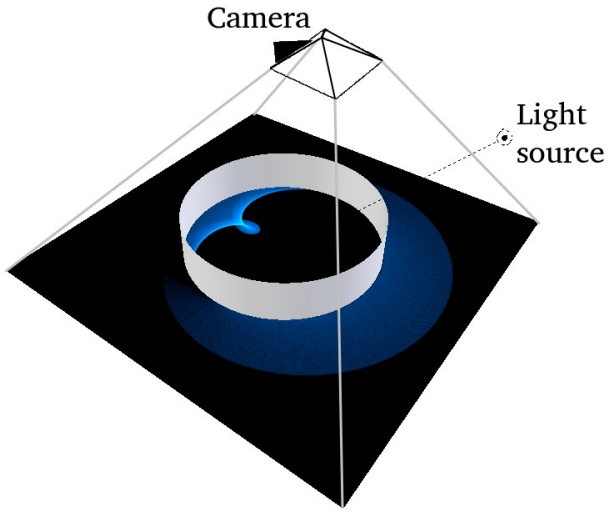
- Zjednodušený renderer – jednoduché materiály a geometria
- Vlastná implementácia, JavaFx pre GUI
- Ukážka funkcionality a všeobecnosti – renderovanie problematických javov : Kaustika, Disperzia

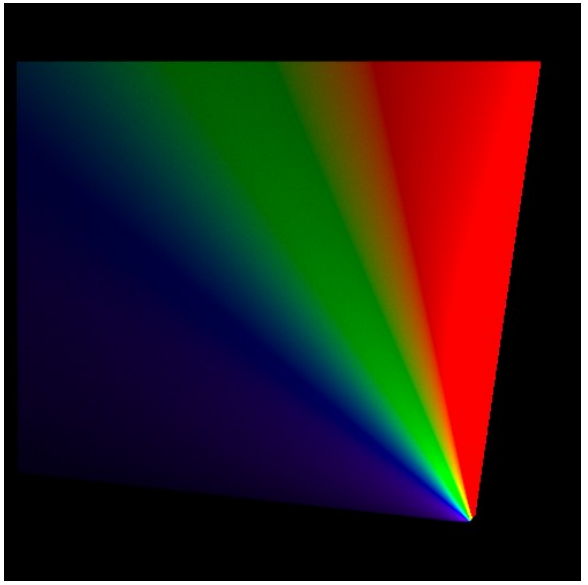
Svetelné zdroje - experimenty











Svetelné zdroje - výsledky

- Kompaktný a všeobecný interface
- Implementácia jednoduchého renderera a niekoľkých parametrizovaných svetelných zdrojov
- Jednoduché renderovanie problematických javov

```
interface LightSource extends java.io.Serializable {
   double getPower();
   beam getNextBeam();
   double getNumberOfBeams();
   double getRadiance(x, omega, E);
}
```