

**Semestrální práce z předmětu KMA/MM
ZS 2013/2014**

Modelování osvětlení scény metodou sledování paprsku

Michal Žák, A13B0079P

1. Úvod

Ray-tracing (česky sledování paprsku) je offline metoda renderingu, tedy zpravidla nepracuje v reálném čase. Slouží k modelování osvětlení scény a výsledky, které poskytuje, působí zpravidla značně realisticky. Zaměříme se na algoritmy s metodou spojené a její neduhy.

Dále se zajímáme o algoritmy, které základní raytracer obohacují o působivé efekty, jakými jsou například rozmazané zrcadlení či lom světla.

2. Algoritmus ray-tracingu pro neprůhledná tělesa

Celý výpočet se v základní podobě dá shrnout do několika kroků:

1. Pro každý pixel vrhni příslušný paprsek do scény.
2. Nalezni nejbližší průsečík paprsku a objektů scény.
3. Spočti lokální model osvětlení v získaném průsečíku, tj. intenzitu I_1 .
4. Nalezni odražený paprsek a nalezni intenzitu I_2 pokračováním algoritmu v bodu 2.
5. Vrať vážený součet I_1 a I_2 .

Významnou vlastností algoritmu je postup od průmětny směrem ke světlu, což odporuje fyzikálním zákonům. Tělesa tedy nemohou „ozařovat“ jiná – nebo obecněji, manipulovat s paprsky, které nekončí na průmětně.

2.1. Vržení paprsku do scény, reprezentace kamery

Pro reprezentaci kamery jsem zvolil popis pomocí tří bodů projekční roviny (označíme je P_1 , P_2 a P_3) a bodu Q reprezentujícího pozorovatele. Dopočítáme bod P_4 :

$$P_4 = P_1 + (P_2 - P_1) + (P_3 - P_1)$$

Body Q , P_1 , P_2 , P_3 a P_4 jednoznačně určují pohledovou pyramidu. Bilineární interpolací vektorů QP_1 , QP_2 , QP_3 , QP_4 získáváme směr paprsku u , který vrháme do scény. Jako zdroj paprsku můžeme použít interpolovaný bod na rovině nebo bod Q . Získáváme parametrické vyjádření polopřímky:

$$x(t) = Q + ut$$

2.2. Nalezení průsečíku

Uvažujeme neprůhledná jednoduchá tělesa definovaná implicitní funkcí $F_i(x) = 0$, kde i značí index typu tělesa. Řešíme soustavu rovnic:

$$\begin{aligned} x(t) &= Q + ut \\ F(x(t)) &= 0 \end{aligned}$$

$$\text{podmínka: } t > 0$$

Minimum ze všech výsledků t je výsledným parametrem, který po dosazení do rovnice

$$x(t) = Q + ut$$

umožní získat souřadnice průsečíku. Je-li množina dílčích výsledků prázdná, pak průsečík s paprskem neexistuje.

2.3. Výpočet intenzity osvětlení a nalezení odraženého paprsku

Pro každé světlo spočteme dílčí kroky a za výsledek budeme považovat jejich součet.

Zkontrolujeme, jestli mezi získaným bodem a zdrojem světla není žádná překážka. Tato část výpočtu v podstatě odpovídá kroku 3.2. Uvažuje se geometrická, nikoliv optická dráha paprsku. Je-li nalezen nový průsečík, bod je v zákrytu a přírůstek intenzity světla nulový.

V opačném případě aplikujeme lokální (např. Phongův) model osvětlení:

$$I = K_D \hat{\mathbf{n}} \cdot \hat{\mathbf{L}} + K_S (\hat{\mathbf{R}} \cdot \hat{\mathbf{L}})^{\text{shininess}}$$
$$\mathbf{R} = \mathbf{v} - 2(\mathbf{v} \cdot \hat{\mathbf{n}}) \hat{\mathbf{n}}$$

K_D ... intenzita difusní složky

$\hat{\mathbf{n}}$ normalizovaný normálový vektor

$\hat{\mathbf{L}}$ normalizovaný vektor směrem ke světlu

$\hat{\mathbf{v}}$ vektor směru vyslaného paprsku

$\hat{\mathbf{R}}$ normalizovaný odražený vektor

Nyní se zaměříme na odražený paprsek. Jeho směr již máme spočtený – použijeme vektor \mathbf{R} nalezený v lokálním modelu osvětlení. Jako pozici nelze bez dalších opatření použít přímo aktuální průsečík, neboť by kvůli omezené přesnosti aritmetiky mohl být v příštím kroce opět nalezen.

3. Modifikace algoritmu pro průhledná tělesa

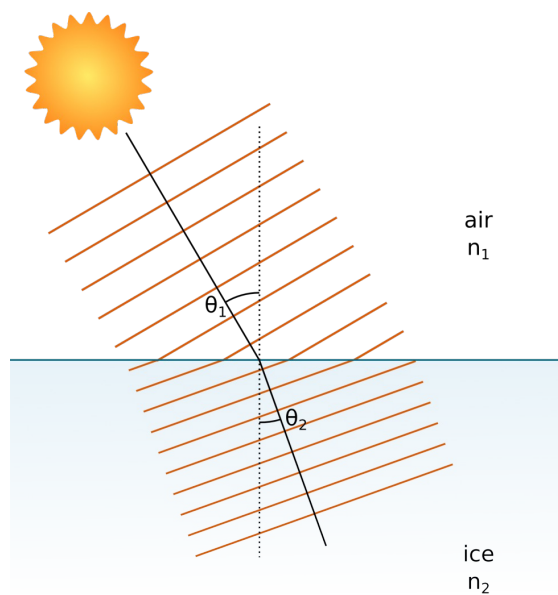
Modifikovaný algoritmus se liší pouze přidáním jednoho kroku (při abstraktním pohledu).

1. Pro každý pixel vrhni příslušný paprsek do scény.
2. Nalezni nejbližší průsečík paprsku a objektů scény.
3. Spočti lokální model osvětlení v získaném průsečíku, tj. intenzitu I_1 .
4. Nalezni odražený paprsek a nalezni intenzitu I_2 pokračováním algoritmu v bodu 2.
5. Nalezni zalomený paprsek, pokud existuje, spočti intenzitu I_3 jím získanou pokračováním algoritmu v bodu 2.
6. Vrať vážený součet I_1 , I_2 a I_3 .

3.1. Výpočet zalomeného paprsku

Zalomený paprsek pokračuje ve směru, který je fyzikálně dán následujícími okolnostmi:

Elektromagnetické vlnění přechází do prostředí s jiným indexem lomu, tedy jinou rychlostí šíření světla. Body rozhraní se stávají podle Huygensova principu novými zdroji vlnění. Jejich interferencí nalezneme vlnění o stejné frekvenci, které můžeme reprezentovat zalomeným paprskem.



Obrázek 1: Lom paprsku na rozhraní dvou prostředí. Kolmice k paprsku znázorňují vlnoplochy.

Zdroj: http://turningmirrors.com/22-halo/halo-snells_law/

Snellův v základním tvaru (se značením shodným jako na obrázku 1) má podobu:

$$\frac{\eta_1}{\eta_2} = \frac{\sin \theta_1}{\sin \theta_2}$$

Pro účely raytraceru potřebujeme vyjádření ve vektorové formě [ros04]:

$$\mathbf{R} = \eta \mathbf{I} - (\eta(\hat{\mathbf{N}} \cdot \mathbf{I}) + \sqrt{k}) \cdot \hat{\mathbf{N}}$$

kde \mathbf{R} je směr zalomený paprsek,
 $\hat{\mathbf{N}}$ je normalizovaná normála rozhraní,
 \mathbf{I} je směr incidentního paprsku,
 η je relativní index lomu.

Proměnnou k vypočteme následovně:

$$k = 1 - \eta^2 (1 - (\hat{\mathbf{N}} \cdot \mathbf{I})^2)$$

Relativní index lomu představuje podíl indexů lomu opouštěného a budoucího prostředí paprsku.

V případě, že k je menší než nula, daný vztah neposkytuje řešení v E^3 . Z fyzikálního hlediska se jedná o tzv. absolutní odraz při přechodu paprsku do opticky řidšího prostředí (např. voda – vzduch), pokračujeme tedy odraženým paprskem.

3.2. Vliv na výpočetní složitost

V průsečíku paprsku s poloprůhledným tělesem nyní vznikají dva nové paprsky – odražený a zalomený. Vezmeme-li v potaz nejhorší možnou variantu, tedy opětovný dopad na průhledné těleso u obou paprsků, pak je počet paprsků v každém dalším kroku zdvojnásoben. Tato skutečnost způsobuje exponenciální výpočetní složitost ray-traceru.

3.3. Výpočet zastínění

Výpočet lokálního osvětlení v daném bodě se s přidáním poloprůhledných těles komplikuje. Zaprvé, musíme zjistit útlum intenzity, nestačí pouze zjistit, je-li zdroj světla v zákrytu. Zadruhé, přichází na řadu zásadní odklonění od reality – z důvodu výkonu volíme pouze geometrickou trasu paprsku (tj. spojnice zdroje světla s místem výpočtu osvětlení, neuvažujíc lom paprsku), nikoliv trasu optickou (takovou, kterou by reálně světlo od zdroje podniklo). Realistické osvětlení scény včetně spekulárních odlesků řeší metoda zvaná photon mapping [jen01], která je v dnešní době používána např. pro rendering filmů v profesionálních studiích.

Nejprve nalezneme průsečíky těles se spojnicí zdroje světla a bodu výpočtu intenzity lokálního modelu. Poté postupujeme ve směru od světla a aktuální intenzitu složek RGB násobíme příslušnými koeficienty propustnosti. Těleso se chová způsobem přirovnatelným k fotografickému filtru.

Alespoň částečně se napodobit skutečné chování paprsku, vypočteme-li útlum v závislosti na odchylce od normály povrchu – s přibývajícím odchýlením více tlumíme paprsek. Ačkoliv tato metoda nemá s realitou nic společného, poskytuje lépe vypadající stíny u zakřivených poloprůhledných těles, například čoček.

4. Matný odraz

S matným odrazem se setkáváme u těles se značnou povrchovou nerovností, která může být i mikroskopického charakteru. Paprsky se zde s danou pravděpodobností odchylují od jinak přesného směru u lesklých povrchů.

4.1. Pravděpodobností rozložení odchýlení

Odchýlení jsem se rozhodl Gaussovským normálním rozložením, které je rotačně invariantní.

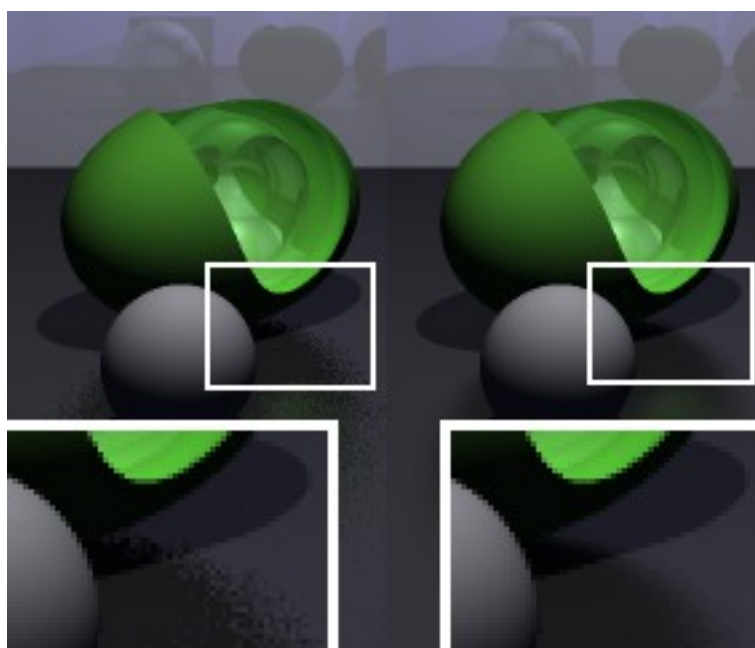
Jako středovou hodnotu μ jsem zvolil nulu, směrodatná odchylka σ je dána materiálovými vlastnostmi. Čím vyšší odchylka, tím více se chová odražený paprsek chaoticky.

Gaussovské rozložení je nejprve vygenerováno součtem rozložení rovnoměrných a následně normováno (úprava, aby $\mu = 0$, $\sigma = 1$). Předgenerujeme si řádově stovky hodnot, které v run-time pouze upravujeme přenásobením na požadovanou odchylku.

Odražený paprsek vzniká sečtením přesného odrazu a ze získaného odchýlení. Právě zde využíváme rotační invariantnost rozdělení. Je možné navzorkovat více odražených paprsků a dosáhnout tak lepší kvality výsledku.

4.2. Vliv počtu vzorků

Obrázek 2 ukazuje vliv počtu vzorků na kvalitu výsledného obrazu. Použijeme-li pouze jeden odrazový paprsek, výsledek je značně zašuměný a není-li hnízdo náhodných čísel voleno obezřetně, obraz může během animace zrnět.



Obrázek 2: Vliv počtu průměrovaných paprsků na kvalitu matného odrazu. Levá polovina používá pouze jeden vzorek, pravá jich používá třicet.

4.3. Výkonnostní dopad

V nejhorším případě výpočet matných odrazů přináší exponenciální složitost o základu rovnajícímu se počtu vzorků. Tento nemilosrdný fakt by případně bylo možné omezit adaptivním vzorkováním, které v semestrální práci realizováno nebylo.

5. Reprezentace těles

Tato část se zabývá vyjádřením jednotlivých těles a hledání jejich průsečíků s paprsky.

5.1. Koule v obecné poloze

- Rovnice $F(x) = 0$ pro kouli: $(x - s_x)^2 + (y - s_y)^2 + (z - s_z)^2 - r^2 = 0$
- Kouli lze vyjádřit rovněž pomocí matice kvadratické formy:

$$\mathbf{M} = \mathbf{T}^T \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -r^2 \end{pmatrix} \cdot \mathbf{T}$$

kde \mathbf{T} je matice posunutí o vektor $-\mathbf{s}$.

- Koule je jedním z příkladů využití kvadrik, ostatní příslušící tělesa (elipsoid, válec...) si dovolím vynechat, neboť z pohledu raytraceru jsou řešena stejně.

5.2. Rovina

- Rovnice $F(x) = 0$ pro rovinu: $ax + by + cz + d = 0$
- Pozice průsečíku se hledá řešením vztahu v projektivním prostoru:

$$\begin{aligned} \mathbf{a}^T \cdot \mathbf{x} &= 0 \\ \text{přičemž} \\ \mathbf{x} &= \mathbf{x}_A + \mathbf{s}t \end{aligned}$$

\mathbf{x}_A ... počáteční bod paprsku

\mathbf{s} ... směr paprsku

t ... hledaná proměnná

- Normála je dána vektorem: $(a, b, c)^T$

5.3. Poloprostor

- Nerovnice pro poloprostor: $ax + by + cz + d \geq 0$
- Řešení je stejné jako u roviny, řešíme navíc pouze to, jestli se počáteční \mathbf{x}_A nachází uvnitř poloprostoru (splnění nerovnice pro nulový vektor \mathbf{s}).

5.4. Kvadratická plocha

- Rovnice pro kvadriku (respektive kvadratickou plochu):

$$\mathbf{x}^T \mathbf{M} \mathbf{x} = 0, \text{ kde } \mathbf{M} \text{ značí matici formy o velikosti } 4 \times 4$$

- Dosazením za $\mathbf{x} = \mathbf{x}_A + \mathbf{s}t$ řešíme kvadratickou rovnici o neznámé t , jejíž koeficienty mají následující hodnoty:

$$\begin{aligned} a &= \mathbf{s}^T \cdot \mathbf{M} \cdot \mathbf{s} \\ b &= \mathbf{s}^T \cdot \mathbf{M} \cdot \mathbf{x}_A + \mathbf{x}_A^T \cdot \mathbf{M} \cdot \mathbf{s} \\ c &= \mathbf{x}_A^T \cdot \mathbf{M} \cdot \mathbf{x}_A \end{aligned}$$

- Normála je dána vztahem: $\text{grad}(\mathbf{x}^T \mathbf{M} \mathbf{x}) = 0$
- Odvozením z předchozího vztahu lze ukázat, že i -tá složka normály se vypočte jako:

$$n_i = \sum_{j=1}^4 M_{ij} + \sum_{j=1}^4 M_{ji}$$

6. Množinové operace

Ray-tracer implementuje základní operace nad tělesy, jako je sjednocení, průnik a množinový rozdíl. Nad základními primitivy je vystavěn CSG strom (constructive solid geometry tree), jehož vnitřní uzly obsahují zmíněné operace a listy naopak primitiva.

Operace se provádí za pomoci intervalové aritmetiky při hledání průsečíku těles s paprskem. Algoritmus nijak nemodifikuje geometrii scény, pouze vkládá průsečíky získané z podstromů (těles) A a B na základě podmínek daných vybranou operací.

Následující sekce rozebírají princip jednotlivých operací.

6.1. Sjednocení ($C = A \cup B$)

V první řadě odstraníme překryv těles A a B v jejich průniku zpřesněním operace sjednocení:

$$C = A \cup (B \setminus A)$$

Pokud provádíme sjednocení, přijímáme všechny průsečíky s tělesem A . Průsečíky s tělesem B přijímáme pouze tehdy, nacházejí-li se mimo těleso A .

6.2. Průnik ($C = A \cap B$)

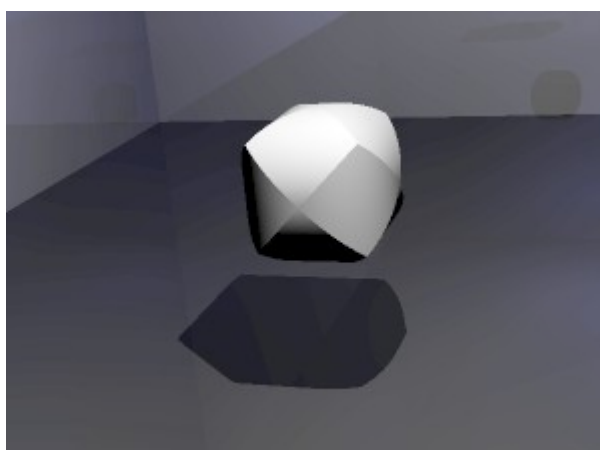
Průsečíky tělesa A přijímáme, pouze pokud se nacházejí uvnitř tělesa B . Obdobně, průsečíky s tělesem B přijímáme, pouze pokud se nacházejí uvnitř tělesa A .

6.3. Množinový rozdíl ($C = A \setminus B$)

Tuto operaci lze přepsat na průnik tělesa A a množinového doplňku B :

$$C = A \cap B'$$

Průsečíky tělesa A přijímáme, pokud se nenacházejí uvnitř B . Průsečíky s tělesem B přijímáme, pokud se nacházejí uvnitř A , a otáčíme jejich normálu (to je způsobeno použitím množinového doplňku tělesa B).



Obrázek 3: Průnik tří válců zarovnaných s osou x , y a z .

7. Různá vylepšení

7.1. Supersampling

Pro vyhlazení výsledného obrazu je vržen větší počet paprsků pro jeden pixel. Jako výsledná hodnota se použije obecně vážený součet dílčích hodnot. Pozice subpixelů může být zvolena odlišnou strategií, např. mřížka, Poissonovo rozdělení, orotovaná mřížka a jiné [gpu05].

V semestrální práci je použita základní mřížka 2x2, která se vyznačuje zbytkovým aliasingem u hran téměř horizontálních či vertikálních.

7.2. Paralelizace

Ray-tracer pracuje paralelně na CPU s více jádry, využívá návrhový vzor Farmer-Worker. Před započítáním renderingu se obraz rozdělí na bloky. Každé pracující vlákno si vyzvedne svůj blok, který následně vykreslí. Po dokončení dílčí práce se proces opakuje. Pixely obrazu jsou na sobě zcela nezávislé, nicméně sběr statistik není – po dokončení pixelu musíme přistupovat ke sdíleným zdrojům, abychom statistiky sečetli. To má za důsledek, že urychlení není přímo rovno počtu jader.

Testování výkonu (na CPU Intel® Core™ i5-3350P @ 3.10Ghz) se scénou obsahující 630 neprůhledných neodrazivých těles a jeden zdroj světla:

Počet jader (vláken)	Doba výpočtu [s]	Urychlení
1	47,83	1
2	24,06	1,99
3	16,49	2,9
4	13,23	3,61

8. Příklad programu, ovládání

Aplikace vyžaduje pro překlad prostředí Microsoft Visual Studio 2010. Hlavním souborem projektu je *src/raytracer.sln*. Přeložený program je závislý na dynamicky linkované knihovně *libpng12.dll* a *zlib.lib*.

Program je ovládán ze svého hlavního menu. Načtení scény a export obrázku se nacházejí pod položkou „File“. Vykreslení obrazu je možné vynutit položkou „Render“ (např. pro opakování procesu).

9. Závěr

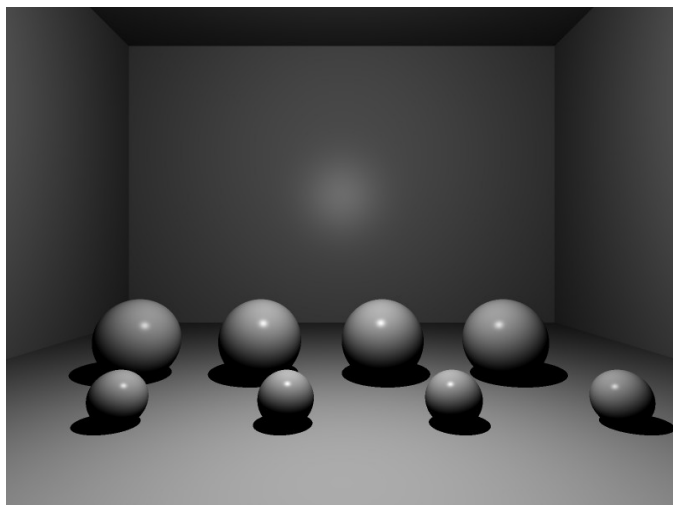
Implementoval jsem základní ray-tracer podporující množinové operace, neostře zrcadlení a lom světla na rozhraních poloprůhledných objektů. Stabilitu ray-traceru dokazuje i to, že s ním bylo vyrenderováno bezchybně 250 snímků animace výpočetně náročné scény. Výsledek lze zhlédnout na adrese http://youtu.be/gOagX4Jeb_A.

Vzniklý ray-tracer bude nejspíše dále využit pro výpočet ambient occlusion map (textur s informací o lokálním zastínění) pro rendering realistické grafiky v reálném čase.

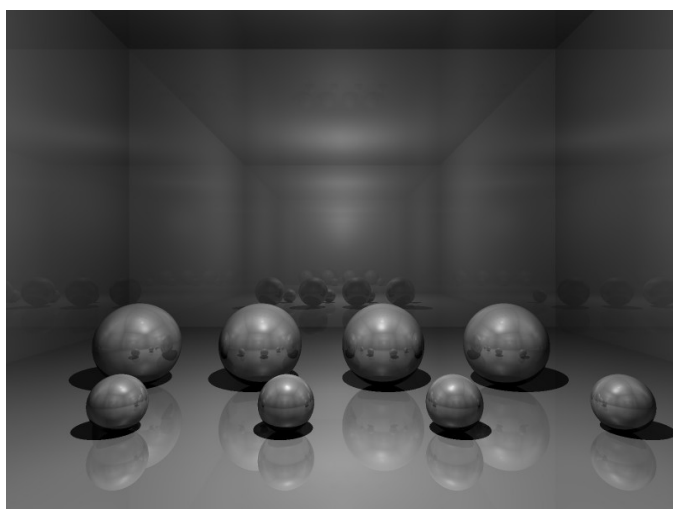
10. Literatura

- [gpu05] GPU gems 2: programming techniques for high-performance graphics and general-purpose computation. 2005. ISBN 0-321-33559-7.
- [jen01] Jennsen W. H. Realistic Image Synthesis Using Photon Mapping. 2001. AK Peters. ISBN 978-1568811470.
- [ros04] Rost R. J.: OpenGL Shading Language. 2004. ISBN 0-321-19789-5.
- [ska93] Skala V. Světlo, barvy a barevné systémy v počítačové grafice. Academia. 1993.

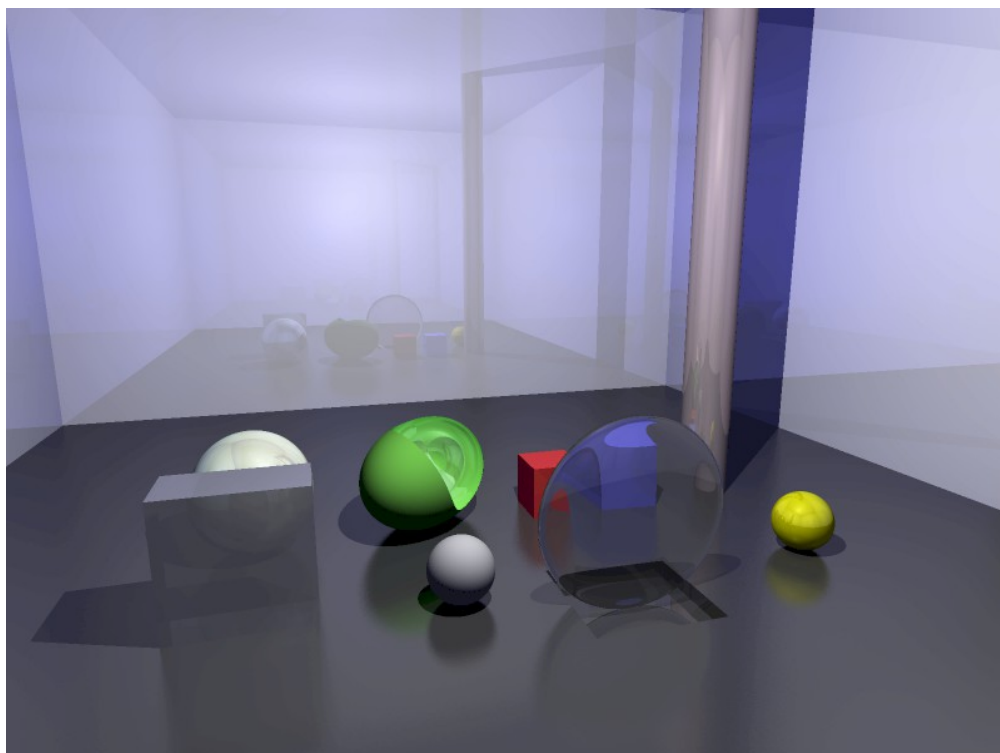
Příloha 1: Obrázky vyrenderované v ray-traceru



Ray-tracer s výpočtem do úrovně 1, tedy bez odrazů.



Ray-tracer s výpočtem do úrovně 5.



Jeden ze snímků vyrenderované animace. Výpočet probíhal do úrovně šesti odrazů, průměrný čas kreslení jednoho snímku odpovídal třiceti minutám (časová náročnost je dána neostrým zrcadlením na podlaze).