

HRY

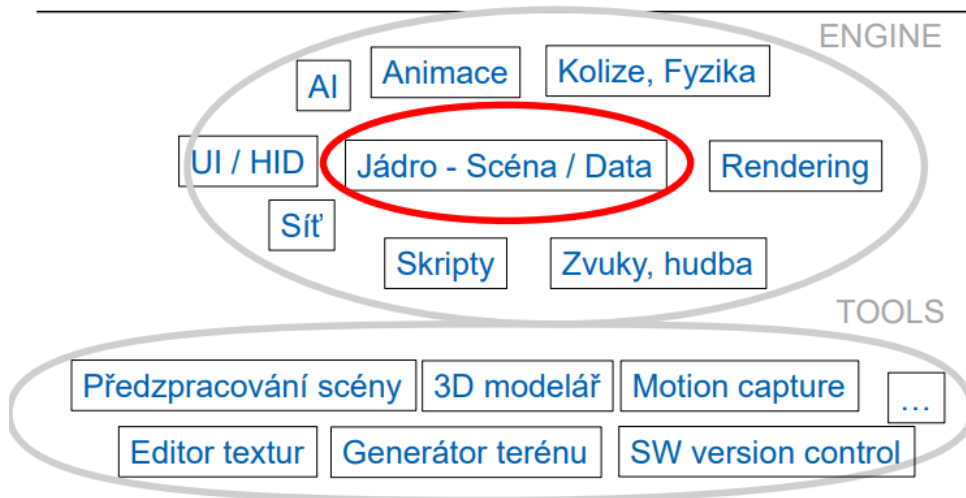
1. Komponenty herního engine, herní smyčka

Přehled komponent herního engine a jejich funkce

Herní engine

- framework pro tvorbu her, předdefinované funkce, do 90. vždy vlastní engine

Komponenty herního engine



Herní smyčka a její varianty, fixní a proměnlivý simulační krok (využití a způsob realizace), souvislost reálného a herního času.

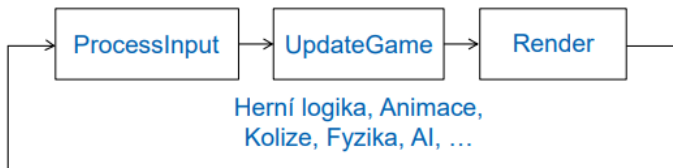
Herní smyčka je jádro hry

Různé časy:

- Skutečný čas - real-time, "čas na hodinách na zdi"
- Herní čas - logický čas ve hře
- CPU / GPU čas - např. doba vykonávání jedné smyčky

Základní herní smyčka

Bez kontroly nad FPS, smyčka poběží, co nejrychleji může



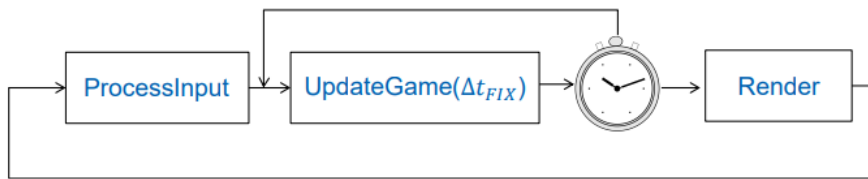
Časovaná herní smyčka

Na rozdílném HW bude velmi rozdílné Δt (delta time, čas od posledního smíčku), což zapříčiní nedeterministické chování hry (různá fyzika, herní logika a animace). Stopky naznačují, že vyčkáme daný čas, aby herní smyčka běžela konstantně.



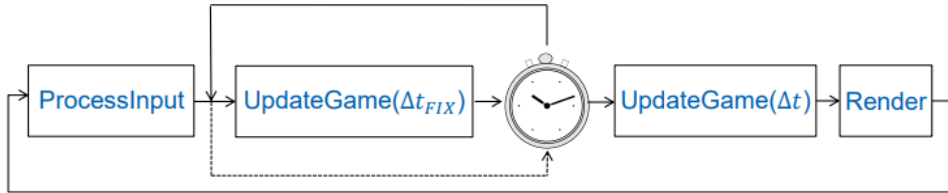
Herní smyčka s opakovanou simulací

Aktualizujeme podle stabilního času Δt_{FIX} . Aktualizujeme scénu dokud simulovaný čas nedožene reálný. Doba výpočtu $UpdateGame(\Delta t_{FIX}) < \Delta t_{FIX}$.



Herní smyčka final

Kombinace předešlých dvou. Dokud je čas, tak aktualizujeme FIX, pak provedeme jeden update a render.



Vazba komponent herního engine na herní smyčku.

Každý komponent se aktualizuje při Update a FixedUpdate pomocí událostí OnRender(), OnCollision() atd.

Zpracování dlouhých procedur.

V Unity používat Coroutines() ve, kterých se dá nastavit, aby se čekalo na začátek dalšího frame nebo nějaký počet sekund.

Synchronizace fyzikální simulace a zobrazování.

Problém, kdy čas update není synchronizován s fixed update. Pohybující objekty mohou mít viditelné šubnutí nebo poskočení.

Řešení je buď interpolovat z posledních stavů nebo extrapolovat (předpovídat) na základě znalosti rychlosti pohybu.

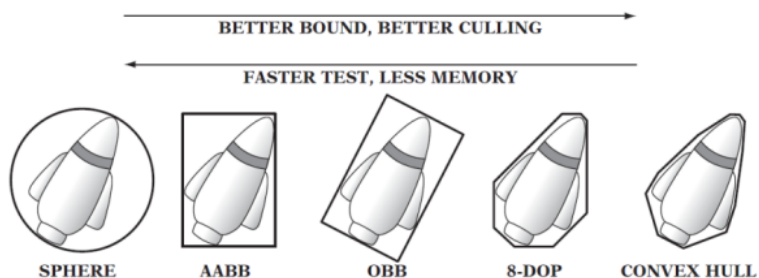
2. Detekce kolizí a základy herní fyziky

Motivace, princip, využití.

Simulace reálné i nereálné fyziky ve hrách. Různé požadavky pro různé herní žánry. Fyzikální simulace pomáhá i škodí ve hře s příběhem volná fyz. simulace může narušit příběh. Nemusí být real-time (animace).

Typy kolizních objektů, algoritmy pro detekci kolizí jednoduchých objektů (koule, AABB, OBB), teorem separačních rovin (SAT), princip algoritmu GJK.

Kolizní těleso aproximuje povrch objektu pomocí jednoduchého tělesa -> rychlé testování průniku.



Koule: Dána středem a poloměrem, velice rychlý výpočet průsečíku, většinou ne příliš těsná, snadno se aktualizuje - invariantní vzhledem k rotaci

Válec: Aproximace postavy, **Capsule:** Rychlejší na výpočet

Axis Aligned Bounding Box (Osově zarovnaný kvádr): Obvykle o něco těsnější než koule, jednoduchý výpočet minimálního AABB

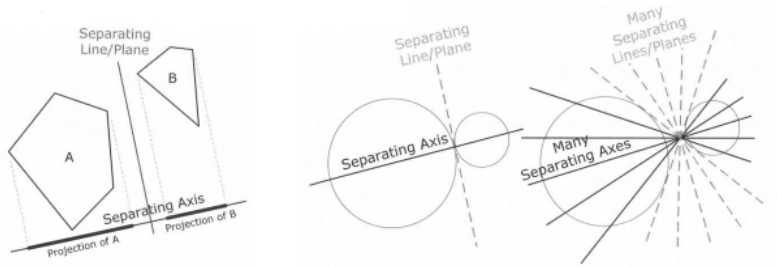
Oriented Bounding Box (Obecně natočený kvádr): Dobré aproximační vlastnosti, složitější výpočet kolizí - Separation Axis Theorem

k-DOP (orientovaný mnohostěn)

Konvexní obálka - test průniku náročnější, GJK algoritmus

SAT (Separation Axis Theorem)

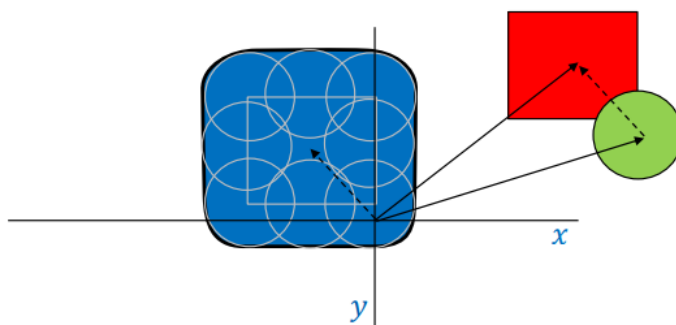
Vyloučení kolize \Leftrightarrow nalezení separační osy, separační osa = normála separační roviny



GJK algoritmus

Obecný algoritmus pro konvexní obálky, který testuje zda Minkowského rozdíl objektů obsahuje počátek. Prochází dvojice vrcholů objektů. Vytvoří části polyhedronu, které reprezentují Minkowského rozdíl - pokud neproniká s počátkem, není kolize, na druhou stranu je kolize nalezena pokud počátek leží v rozdílů.

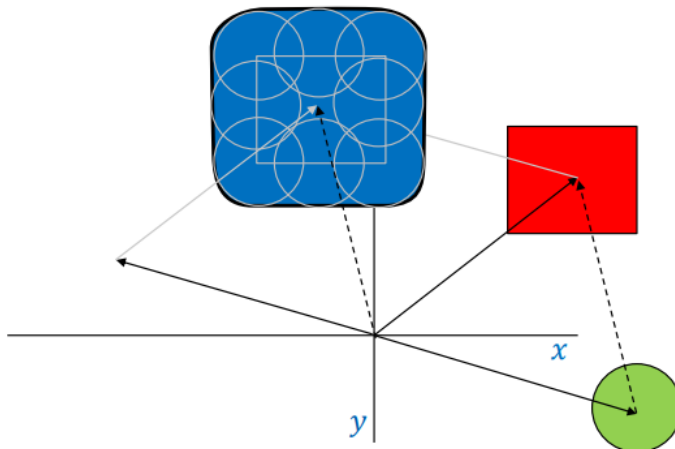
Minkowského rozdíl – je kolize



$$A - B = \{a - b | a \in A, b \in B\}$$

24

Minkowského rozdíl – není kolize



$$A - B = \{a - b | a \in A, b \in B\}$$

23

Fáze detekce kolizí, filtrování kolizí, akcelerace pomocí mřížky a hierarchie obalových těles.

Detekce kolizí:

- **Diskrétní (statická) detekce:** statický "snímek" scény, nestačí pro rychle se pohybující se objekty (bullet through paper problem)

- **Spojité (dynamická):** uvažuje pohyb i mezi snímky, Problém: nalézt první okamžik kontaktu
- **Tažená kolizní tělesa:** Model pohybu mezi snímky - Intervalová aritmetika
- **Pseudo-spojité detekce:** diskretní detekce po krátkých intervalech
 - Δt pomocí horního odhadu rychlosti tělesa
 - binární hledání v čase 0 až T pro přesný čas kontaktu

Výstupem je True/False jestli tělesa kolidují a zároveň informace o kolizi -> kolizní objekty, pozice, vzdálenost, normála, lokální souřadný systém

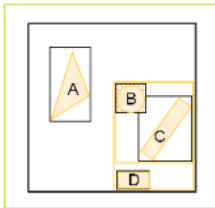
Fáze detekce kolizí:

Široká fáze (broad phase): konzervativní testy - najde skupiny těles které spolu mohou kolidovat, rychle oddělí ty, které určitě nekolidují

Akcelerační struktury: Vyhledávání v logaritmickém čase, nejdříve postavit a pak traverzovat, hlavně v široké fázi, aktualizace - rebuild (pomalé), přepočítat jen dané části

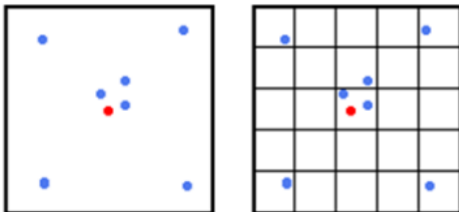
Dělicí objekty:

- Hierarchie obálek (Bounding Volumes Hierarchy)
 - Strom, kde každý uzel má obálku a n potomků
 - Detekce všech možných kolizí: Uzel A/B koliduje? ano - přidat všechny páry potomků na zásobník / ne - ukončit větev
 - Stavba: inkrementální (vkládání), shora dolů (dělení), zdola nahoru (slučování)



Dělicí prostor:

- Mřížka - dělí prostor na buňky - seznam objektů, který jí protínají, v konstantním čase zjistíme ve které jsme buňce



- Octree
- kd-tree
- BSP (Binary Space Partitioning)

Filtrování kolizí:

- Vrstvy, Kolizní maska - Povolení / zakázání některých kolizí

Kolizní vrstvy - Maska vrstev pro každý objekt (Havok) - Collision matrix (Unity) Callbacks - AddContactPoint – herní logika může kontakt zrušit

Kolizní materiály - Materiál (textura) nese informaci o fyz. / kolizních vlastnostech, friction, bounciness

Kolizní dotazy a jejich varianty, reprezentace kolize.

- Kolize dynamických objektů - Detekce všech kolizí - napojení na fyzikální engine
- Vrhání paprsků (Ray cast) - Nalezení nejbližšího průsečíku s paprskem, střelba, AI, detekce polohy, kola vozidla,
- Vrhání tvarů (Shape cast) - Nalezení nejbližších průsečíků s koulí, kapsulí, kvádrem, atd., pohyb postav, plánování cest, polohování kamery, ...

Smyčka fyzikální simulace.

Smyčka fyzikální simulace:

1. Aktualizuj herní objekty (GUI inputs, herní logika)
2. Aktualizuj síly, aplikuj silové impulzy, uprav vazby

3. Simulační krok

1. Numerická integrace
2. Detekce kolizí
3. Nejsou kolize - vyskoč
4. Řešení kolizí

4. Aktualizuj "fyzikální" herní objekty

5. Kolizní dotazy (předešlá kapitola)

6. Render

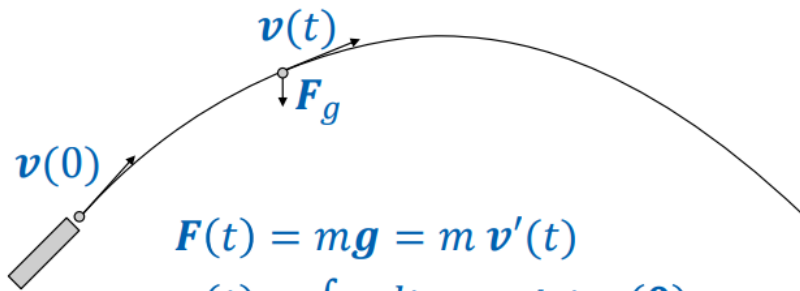
Analytické řešení pohybové rovnice, řešení numerickou integrací, Eulerova metoda, příklady.

Pohybová rovnice podle Newtonova 2. zákona síly.

$$\begin{aligned} F(t) &= m a(t) = m \frac{dv(t)}{dt} = m \frac{d^2 r(t)}{dt^2} \\ &= m v'(t) = m r''(t) \end{aligned}$$

Analytické řešení - pomocí integrace

Příklad: Letící střela, analytické řešení


$$\begin{aligned} F(t) &= mg = m v'(t) \\ v(t) &= \int g dt = g t + v(0) \\ r(t) &= \int v(t) dt = \frac{1}{2} g t^2 + v(0)t + r(0) \end{aligned}$$

Numerické řešení - výpočet pomocí diskrétních kroků

Eulerova metoda -

$$r'(t) = v(t)$$

$$v(t + \Delta t) = v(t) + \frac{F(t)}{m} \Delta t$$

$$r(t + \Delta t) = r(t) + v(t) \Delta t$$

předpoklad ... $v(t)$ je konst. během snímku

(https://cw.fel.cvut.cz/b221/_media/courses/b4b39hry/protected/05-fyzika.pdf)

Typy vazeb mezi objekty a jejich využití. Výpočet reakce na kolize a fyzikální materiály.

Simulace pohybujících se těles: Mnoho těles a potenciálních interakcí, soustava diferenciálních rovnic, omezující podmínky = vazby mezi objekty, iterativní minimalizace chyby řešení soustavy

Pevný kontakt

- objekty jsou pevně ukotveny k sobě
- příklad: chytání objektů rukama ve VR, lepení objektů na sebe

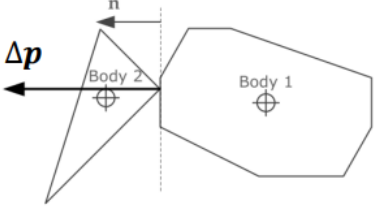
Kloub

- Ukotvení k bodu v prostoru, možnost rotace
- příklad: joystick
- **Pant (hinge)**
- ukotvení 2 objektů, rotace pouze podél specifikované osy
- příklad: dveře, kolo
- **Píst (slider)**
- pohyb po ose, max, min
- příklad: tlačítka, píst
- **Tlumená pružina**
- tuhost, tlumení, min a max délka
- příklad: pružiny

Reakce na kolize – výpočet impulzu

$$p_2 = m_2 v_2$$

$$p'_2 = p_2 - \Delta p$$



$$p_1 = m_1 v_1$$

$$p'_1 = p_1 + \Delta p$$

$$p_1 + p_2 = p'_1 + p'_2 = \text{const}$$

$$\Delta p = ?$$

$$\Delta p = \frac{(\epsilon + 1)(v_2 n - v_1 n)}{\frac{1}{m_1} + \frac{1}{m_2}} n$$

$$v'_1 = v_1 - \frac{\Delta p}{m_1}$$

$$v'_2 = v_2 + \frac{\Delta p}{m_2}$$

Celková hybnost izolované soustavy těles je konstantní

ϵ odrazivost
 0 plastický kontakt
 1 elastický kontakt

immo vypočítá to impuls při kolizi který odrazí jeden ten objekt

Fyzikální materiály:

- Odrazivost (bounciness)
 - 0 – neodráží se, tělesa se pohybují společně (plastický kontakt)
 - 1 – veškerá energie dopadu se přemění v energii odrazu (elastický kontakt)
- Tření (friction)
 - Koeficient statického tření (před uvedením objektu do pohybu)
 - Koeficient dynamického (kinematického) tření (objekty v pohybu)
- Interakce mezi objekty
 - způsob kombinace tření (avg, min, max, multiply, ...)
 - způsob kombinace odrazivosti (avg, min, max, multiply, ...)

3. Reprezentace a výpočet animací.

Typy animací a jejich využití.

Cíl: Rozpohybovat objekty

Animace obrázků

- Cell, sprite animace, animování jednotlivých snímků, 2D hry

Rigidní hierarchická animace

- změna transformací v hierarchii scény
- stoje, jednoduché pohyby

Animace vrcholů (per vertex)

- morphing (blend shapes) - tvář, výrazy
- procedurální animace - vodní hladina, šumové funkce
- fyzikální simulace - šaty, látka

Skeletální animace - viz další kapitola

Podrobný popis skeletální animace, reprezentace kostry.

Kostra je pomocná struktura určující jak se bude objekt deformovat. Jednoduší animace, úspora paměti.

Vstup:

- Kostra (zadána stromem, uzly jsou klouby, hrany jsou kosti)
- Pokožka - 3D model
- Oba jsou navrženy v referenční poloze T-pose

2 fáze:

Animace kostry

- rigidní transformace kostry
- každý kloub je v hierarchii a platí na něj skládání transformací
- pořadí - Scale -> Quaternion rotation -> Translate
- ale většinou se u kloubů používá jen rotace a translace celé kostry
- 4 různé animační techniky
 - **Dopředná kinematika** - přímé ovládání kostry, náročné na tvorbu, Klíčování: určí se jen klíčové snímky, které se pak interpolují
 - **Motion capture** - Záznam pohybu herce, Následné vyčištění a zpracování
 - **Inverzní kinematika** - Nepřímá kontrola kostry, rotace kloubů se spočítají podle cíle, například aby se nohy postavy dotýkali země i na nerovném povrchu, iterativní řešení, nejednoznačné řešení (také nemusí existovat)
 - **Dopředná dynamika** - Fyzikální simulace, např. zabití postavy a pád / odhození stranou: ragdoll effect

Skinning - pozdější kapitola

Animační klipy - vytváření, reprezentace, výpočet animované kostry.

Animační klip je základní (krátký) animační sekvence. Například: Chůze, běh, Útok vpravo-vlevo atd. Definované v lokálním čase. Reprezentováno klíčovými snímky, které se pak interpolují.

Míchání animačních klipů - způsob výpočtu a příklady.

Potřeba když chceme přehrávat několik klipů dohromady, nebo přecházet z jednoho do jiného. Obecně LERP na SQT transformačních kloubů. Blending Tree - možnost míchání animací podle parametru

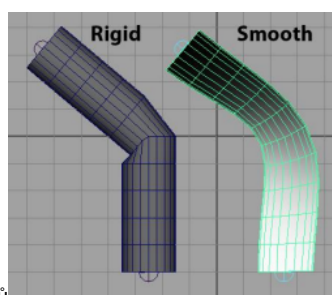
Animační automat (ASM)

- Množina stavů (př. Idle, Motion, Jumping, Fighting)
- Každý stav svůj strom míchání klipů
- Přechody mezi stavy (cross-fade), míchání v čase (LERP)
- Jeden automat je jedna vrstva. Jednotlivé vrstvy se dají míchat (váhy) a mají nastavitelnou masku (např. animace jenom části těla). Aditivní míchání - obě vrstvy ovlivňují animaci

Skinning, způsob výpočtu pozic vrcholů metodou linear blend skinning.

Skinning - obalení kostry kůží (modelem aneb detailní polygonální sítí)

Rigid skinning



- každý vrchol má daný kloub, nevhodné pro vrcholu k kloubů

Linear Blend Skinning

- každý vrchol je ovlivňován několika klouby (2-4 typicky) s danými váhami
- váha je konvexní - větší než 0 a součet všech vah je roven 1

$F(j)$ je (složená) transformace kloubu v animované poloze

$A(j)$ je (složená) transformace kloubu v referenční poloze (T-pose)

- Výpočet - w - váha kloubu pro bod

$$v' = w_1 F(j_1) A^{-1}(j_1) v + \dots + w_m F(j_m) A^{-1}(j_m) v$$

- Rychlý výpočet, jednoduchá implementace -> pokročilé -> redukce artefaktů

Animační křivky - princip, využití, Hermitova kubika, Catmull-Rom spline.

viz. PGR, VGO, potřetí už fakt ne

4. Základní optimalizační metody pro herní engine.

Cíl - Zachovat konstantní FPS (60 fps, vr 90 fps). Nejvíce času zabírá rendering.

Dopředné a odložené vykreslování, princip, srovnání.

Dopředné vykreslování

- **Jednoprůchodové** osvětlování - Každému objektu spočítáno osvětlení od všech světél společným shaderem. Omezený počet světél
- **Víceprůchodové** osvětlování - Několik vykreslovacích průchodů (skupiny světél). Různé, ale jednodušší shadery. Násobné vykreslování geometrie, opakované výpočty

Odložené vykreslování

- **Postup**
 - Vykreslení objektů bez osvětlení
 - Pro každý pixel data předpřipravená pro osvětlení (G-buffer)
 - Shading jako postprocess s využitím G-bufferu
- **G-buffer** - Každý pixel má uložená data potřebná pro výpočet osvětlení jako barvu, koeficient a ostrost spekulárního odrazu, normála, hloubka...
- **Výhody** - Zredukován počet vykreslovacích příkaz. Výpočty osvětlení se provádějí pouze nad viditelnými pixely. Snadné další postprocess efekty
- **Nevýhody** - Nároky na paměť, Transparentní objekty nutno řešit zvlášť, Problematický antialiasing (multisampling)

Celkový postup

- Deferred rendering pro neprůhledné objekty
- Forward rendering pro transparentní objekty
- Post-process - Screen space motion blur / depth of field, FXAA...

Optimalizace scény pomocí LOD, typy LOD a způsob jejich přepínání.

Scéna se dá **optimalizovat** pomocí:

- Redukování detailů a jejich reprezentací pomocí textur (bump, normal, mip-map, atlas).
- Efektivní indexace geometrie (indexed triangle list)
- Dělení modelu/herního světa na bloky
- Dělení na statické a dynamické modely
- LOD

LOD

- Model je definován ve více úrovních detailů, mezi kterými se přepíná podle toho kolik místa zabírají v pohledovém jehlanu/vzdálenosti.
- **Diskrétní** - několik přepočítaných LOD, skokové přepínání, hráč si toho všimá
- **Spojitě** - dynamicky se generují za běhu, plynulá úprava složitosti modelu, výpočetně náročnější, Nanite UE5

Předpočítání osvětlení pomocí osvětlovacích map a sond odrazu - princip, způsob použití, omezení.

Globální osvětlení scény jsou předpočítány a uloženy do textury (light map). To bohužel nejde pro lesklé povrchy a na to se používají předpočítané reflection probes.

Redukování podle viditelnosti: redukování pohledovým jehlanem, redukování podle zastínění (online a offline), potenciálně viditelné množiny (PVS) - výpočet a použití.

Culling - zahazování trojúhelníků, které nepotřebujeme vykreslovat -> zachovává se jen to co je vidět

Redukování pohledovým jehlanem - Všechny objekty jejichž bounding box (rozsah v prostoru) nezasahuje do vykreslovacího jehlanu, nejsou vykresleny. Výsledek nemusí být přesný a vyplatí se používat konzervativní algoritmus.

Redukování podle zastínění - Vhodné pro interiéry nebo koridorové scény.

- **Offline** - Přepočítává se viditelnost pro jednotlivé statické části scény. Pohledový prostor se rozdělí na pohledové buňky. Pro každou buňku je vypočítána viditelná množina objektů (PVS). Výpočet PVS je velmi náročný a je potřeba efektivně vzorkovat.
- **Online** - Výpočet PVS pro každý snímek. Sice jednodušší, protože známe přesnou pozici kamery, ale musí to být mnohokrát rychlejší. Možnost grafu sousednosti místností, kde dveře jsou portály ve kterých testujeme viditelnost. Možnost použít předešlého snímku při hierarchické reprezentaci scény, kde si postupně aktualizují, viditelnosti jednotlivých odkazů.