Global Illumination of Point Models

Rhushabh Goradia: Fifth Annual Progress Seminar

http://www.cse.iitb.ac.in/~rhushabh

Guide: Prof. Sharat Chandran

Date: 30/09/09

ViGIL

IIT Bombay

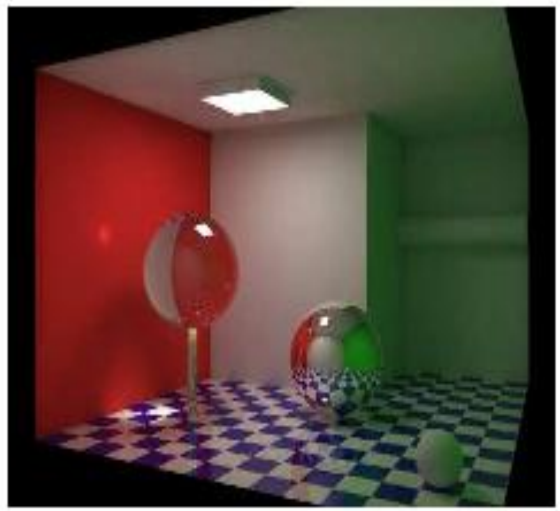
Problem Statement

- To compute a global illumination (GI) solution for complex scenes represented as point models

Problem Statement

- To compute a global illumination (GI) solution for complex scenes represented as point models

(GAS 08, ICVGIP)



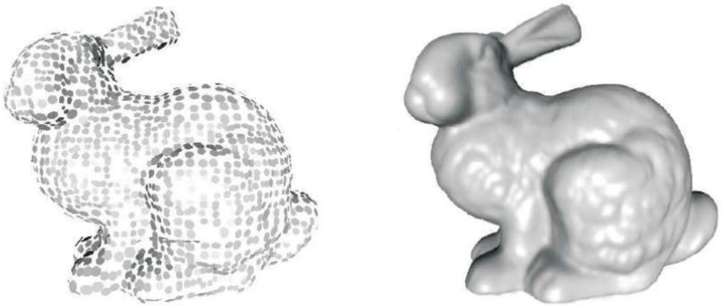
- Diffuse Effects
- Specular Effects

→

- Color Bleeding
- Soft Shadows
- Reflections and Refractions
- Caustics

Problem Statement

- To compute a global illumination (GI) solution for complex scenes represented as point models



- Point Models: Discrete representation of a continuous surface
- No connectivity information between points
- Each point has certain attributes, e.g. Co-ordinates, normal, color

Motivation

- Virtual walkthroughs -- The Digital heritage project (Microsoft Research, India)

- Preserving monuments and Renovation
- E-Museums!

Challenges

- **Problem Statement:** To compute a global illumination solution for complex scenes represented as point models



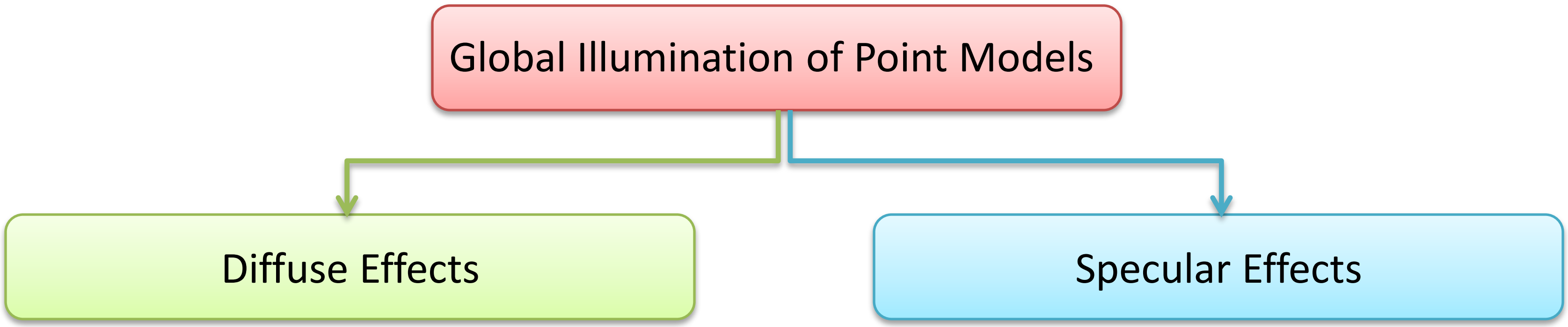
- **Hard to segment entities:** Inhibiting Surface Re-construction
- **GI Algorithm should handle all sort of surfaces:** Diffuse and Specular
- **Expensive!**
- **Fast Ray-Tracer**

Challenges

- Problem Statement: To compute a global illumination solution for complex scenes represented as point models



- Hard to segment entities: Inhibiting Surface Re-construction
- GI Algorithm should handle all sort of surfaces: Diffuse and Specular
- Expensive!
- Pioneers to give a complete GI package!


Global Illumination of Point Models
Diffuse Effects
Specular Effects

Plan

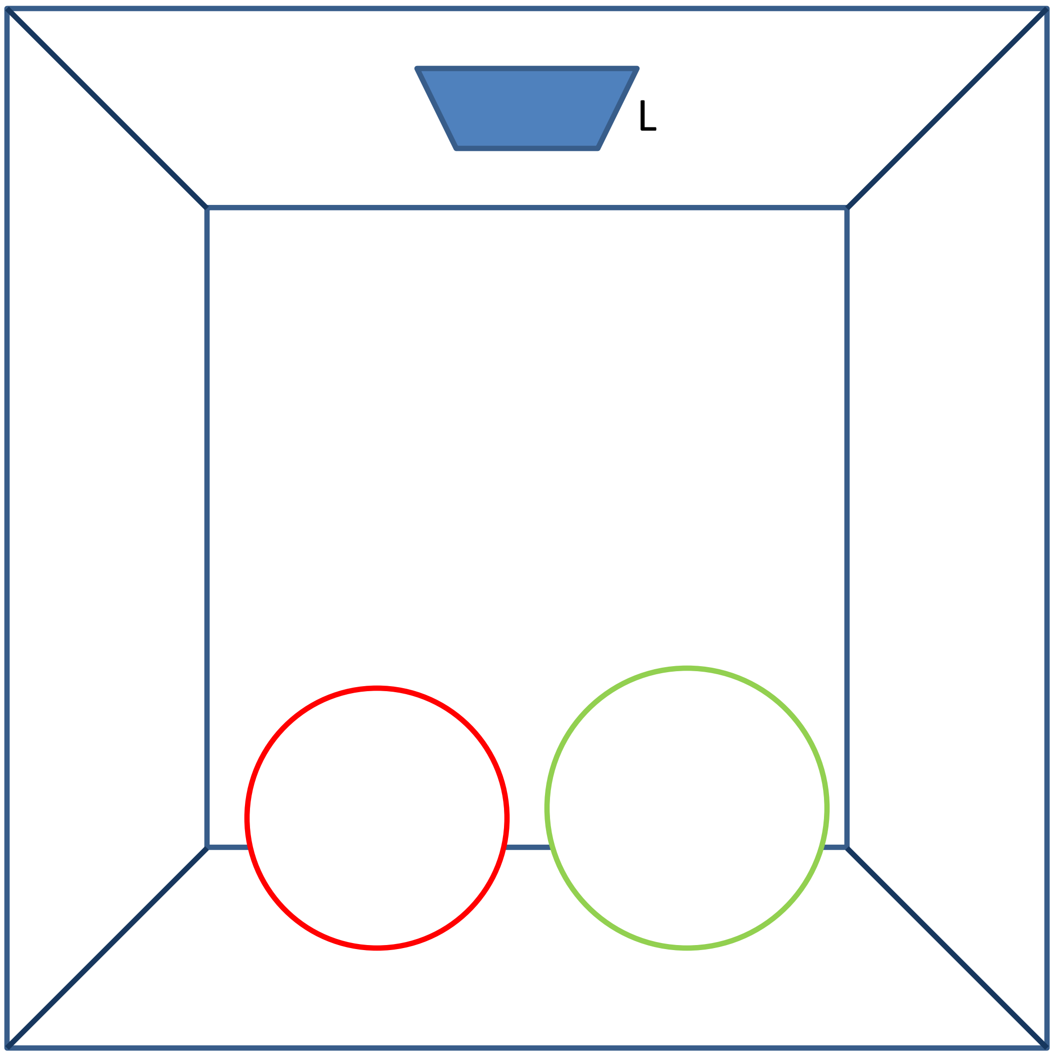
- Introduction – *Problem Definition*
- Diffuse Effects – *Overview*
- Specular Effects – *Details*
- Results
- Wrap-up

Plan

- Introduction – *Problem Definition*
- Diffuse Effects – *Overview*
- Specular Effects – *Details*
- Results
- Wrap-up

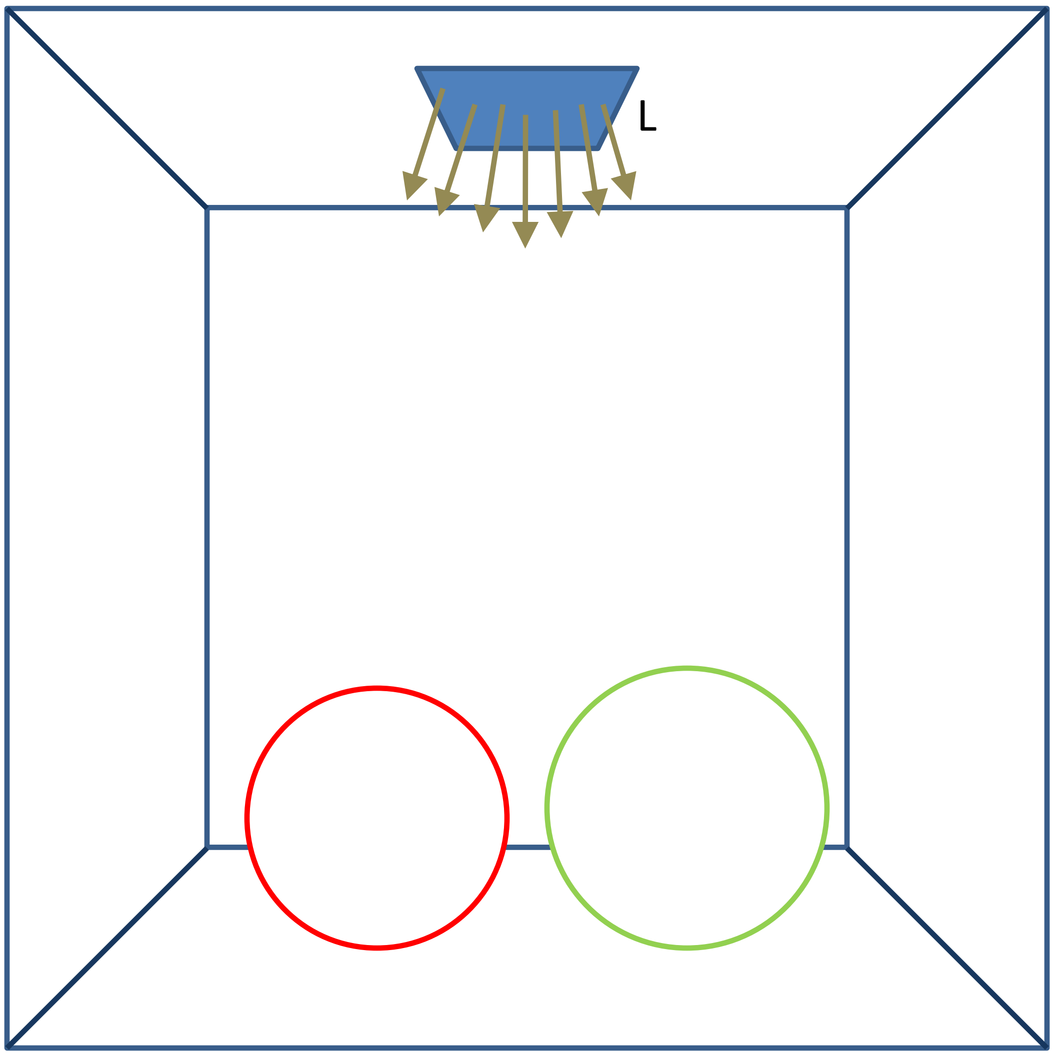
Diffuse Interactions

- A Diffuse scene



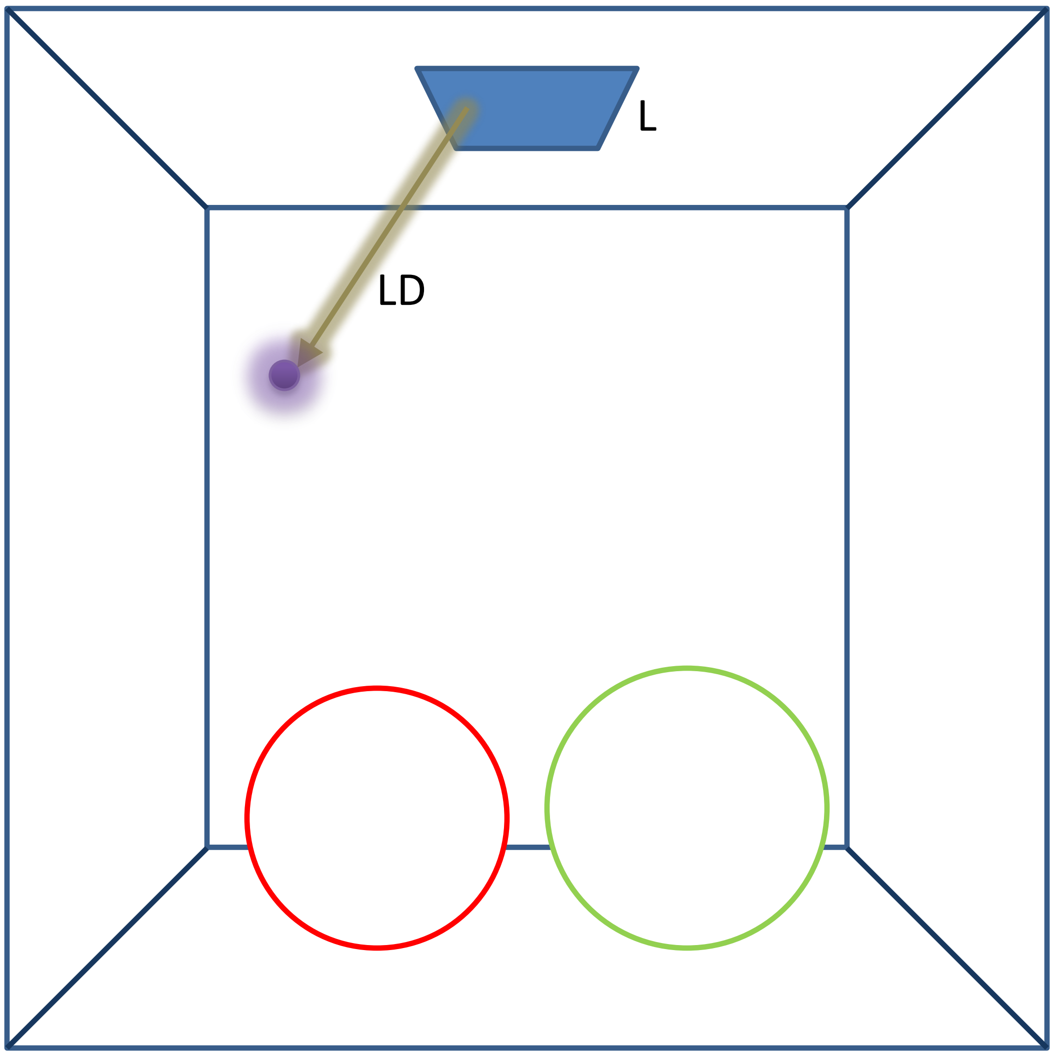
Diffuse Interactions

- Source emits light in all directions



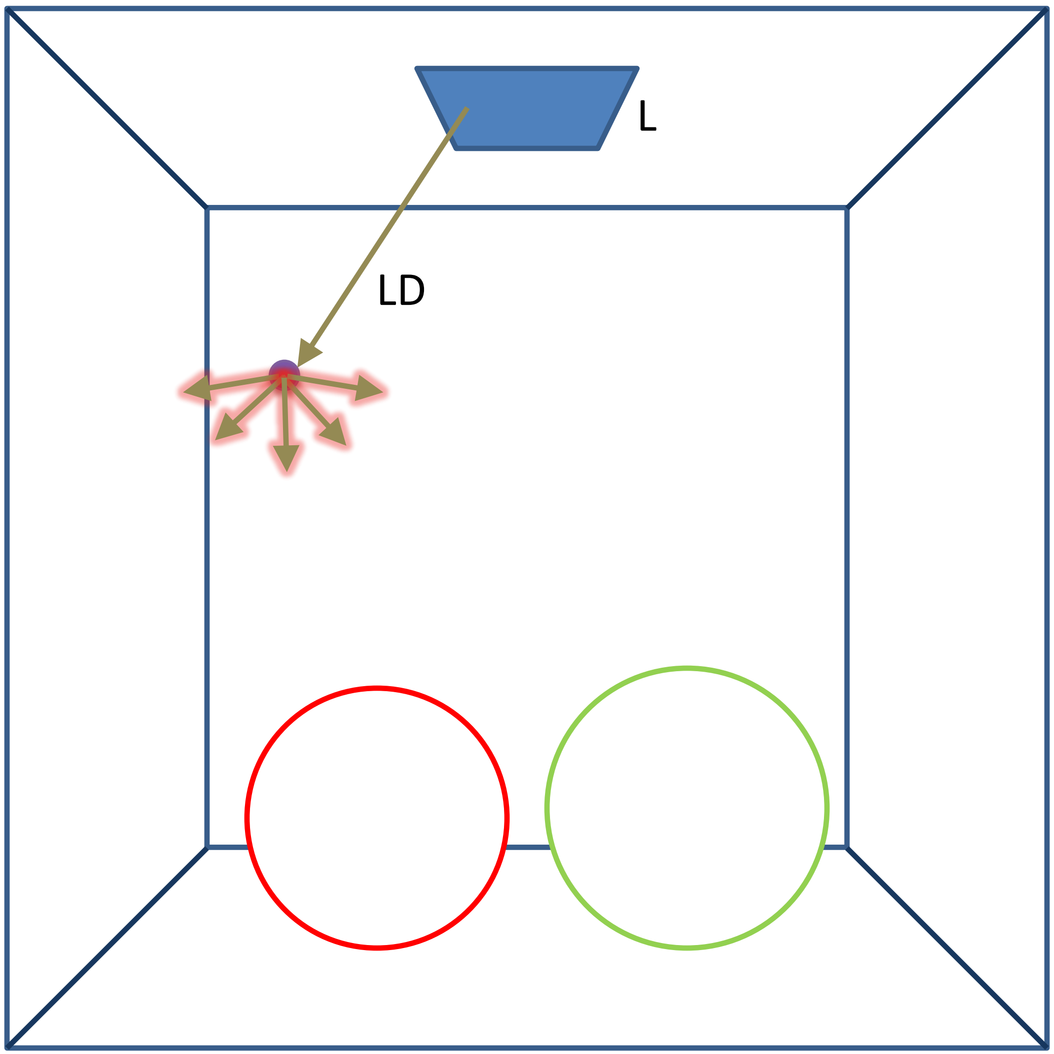
Diffuse Interactions

- Follow one particular ray
- LD path



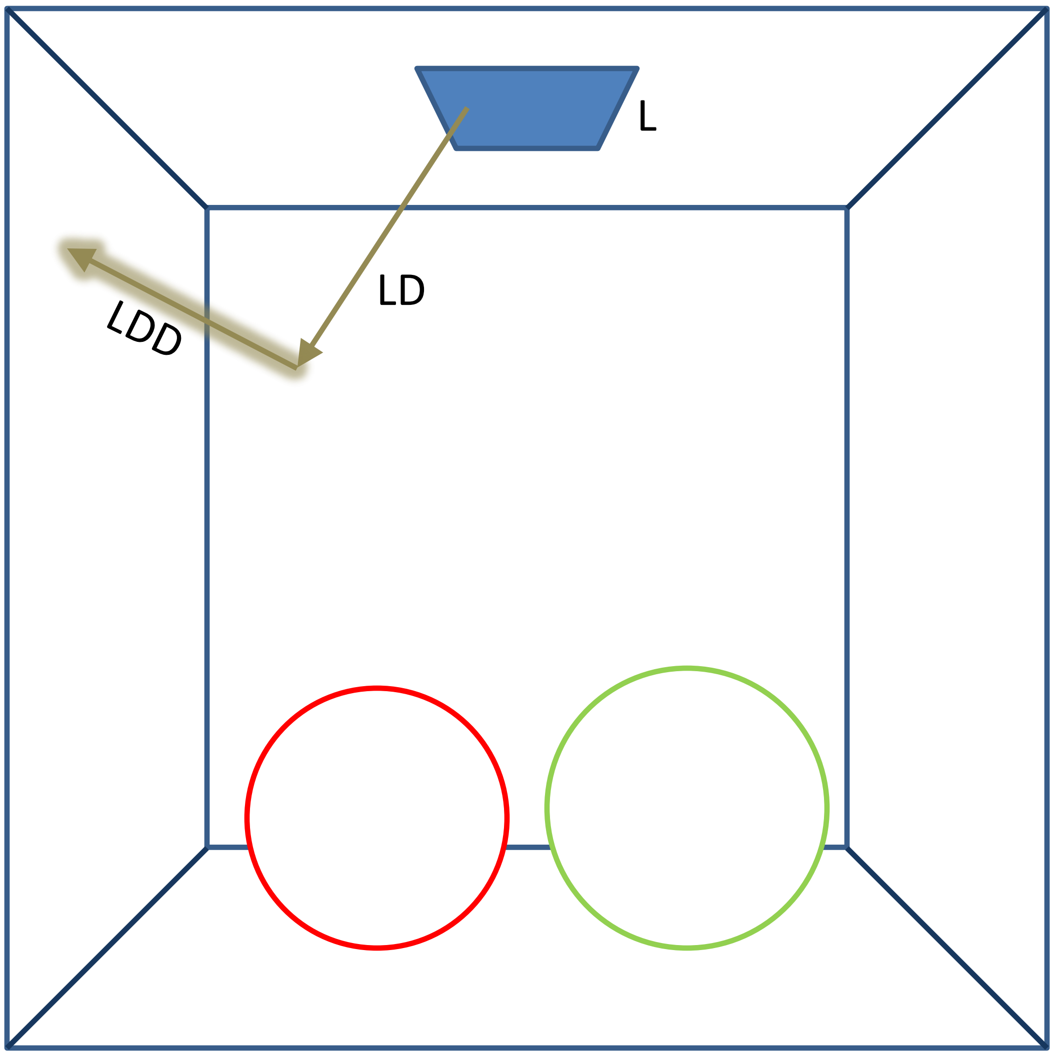
Diffuse Interactions

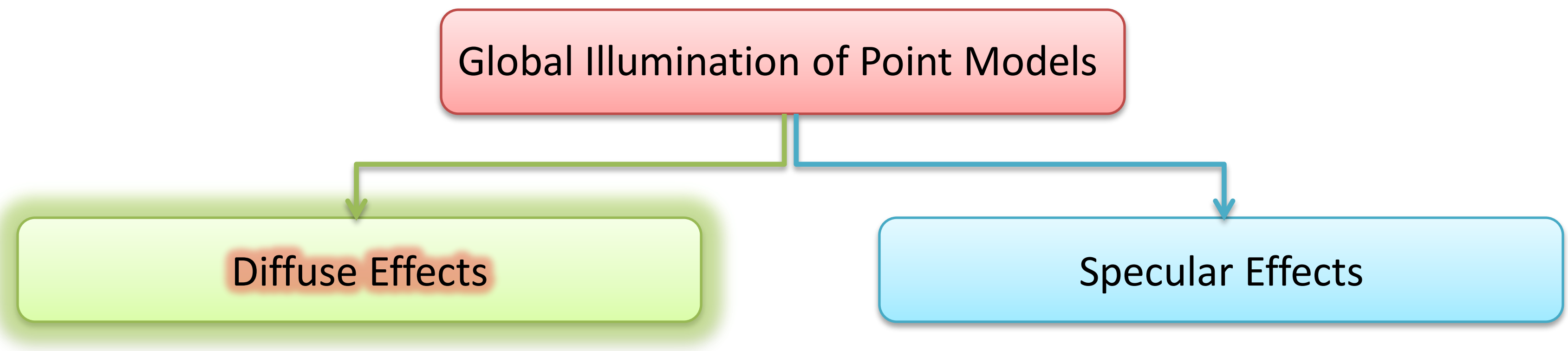
- Diffuse BRDF
- High Absorption



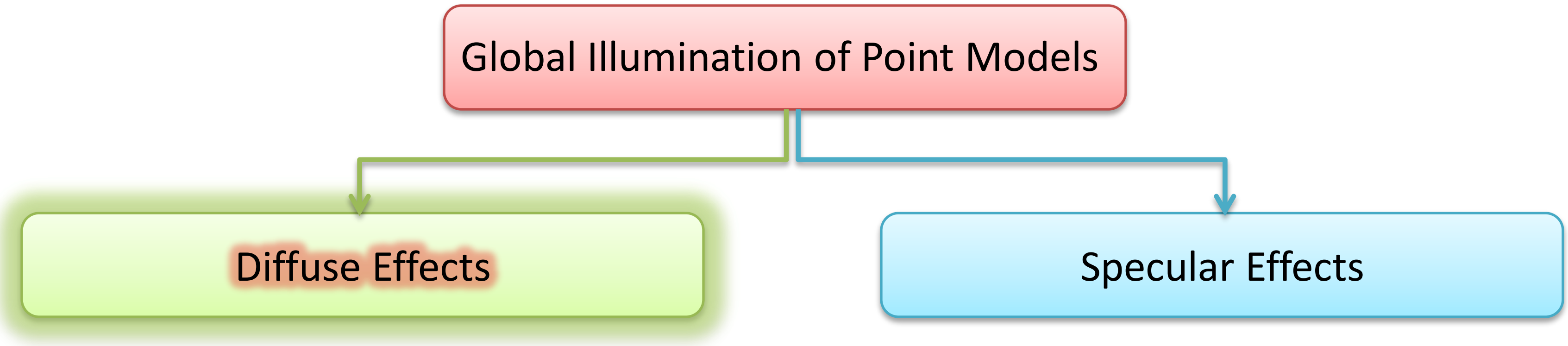
Diffuse Interactions

- LDD/LD$^+$ path

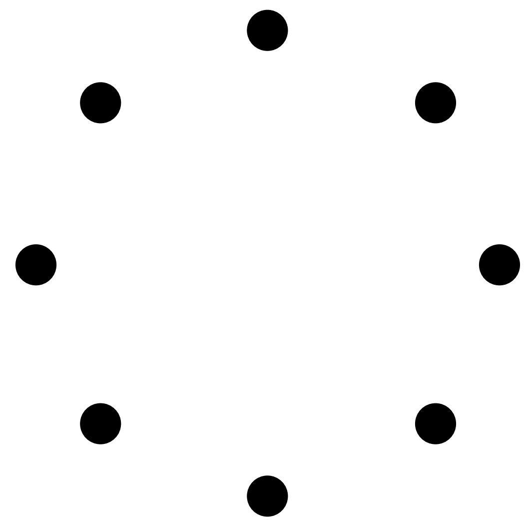




➡ Radiosity based GI Algorithm



- Radiosity based GI Algorithm
- A *N-body* problem

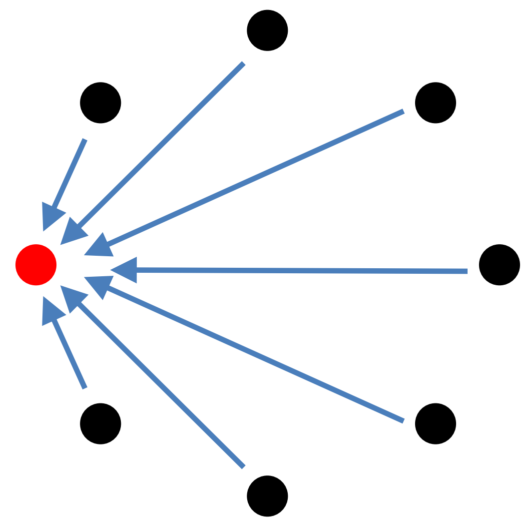


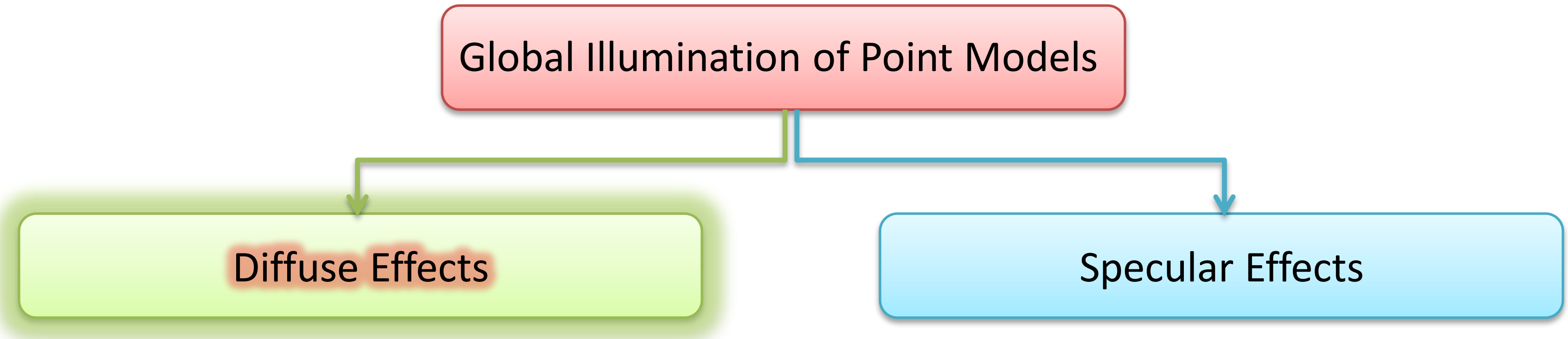
Global Illumination of Point Models

Diffuse Effects

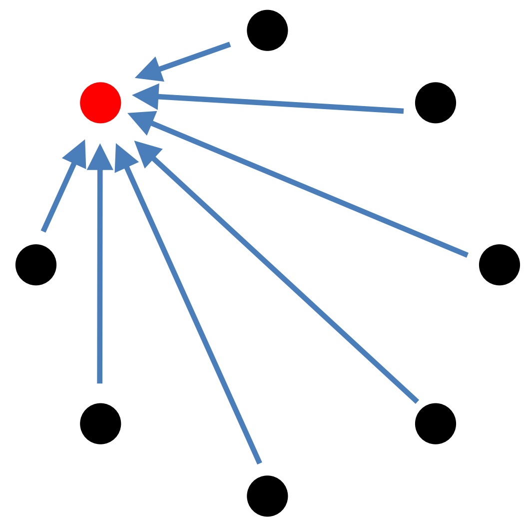
Specular Effects

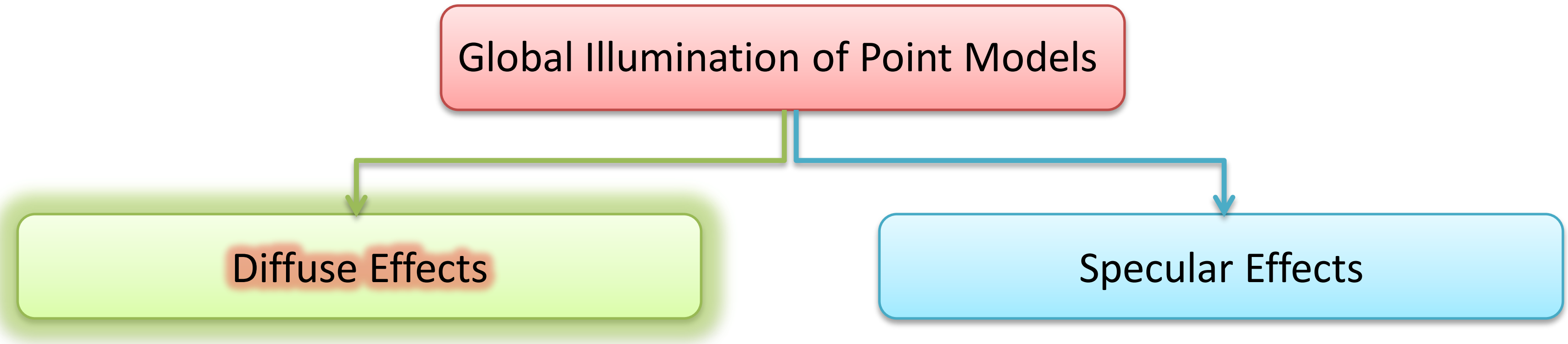
- Radiosity based GI Algorithm
- A *N-body* problem





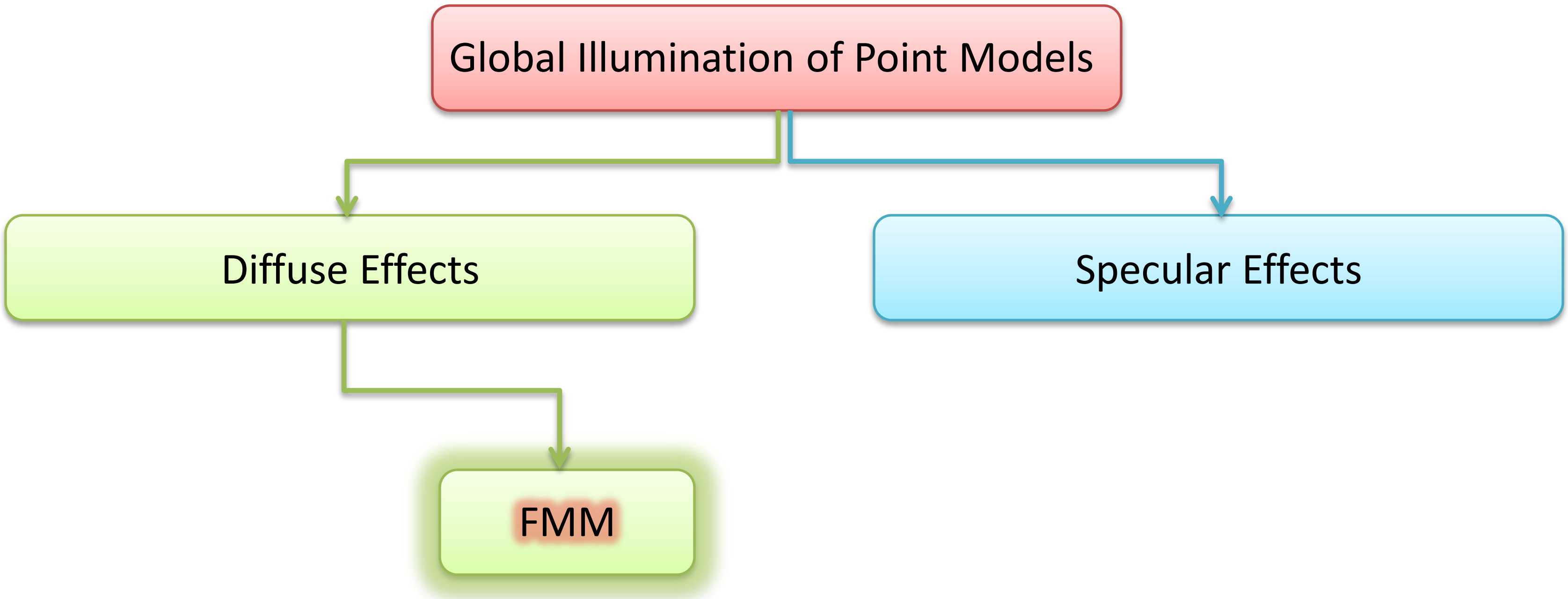
- Radiosity based GI Algorithm
- A *N-body* problem



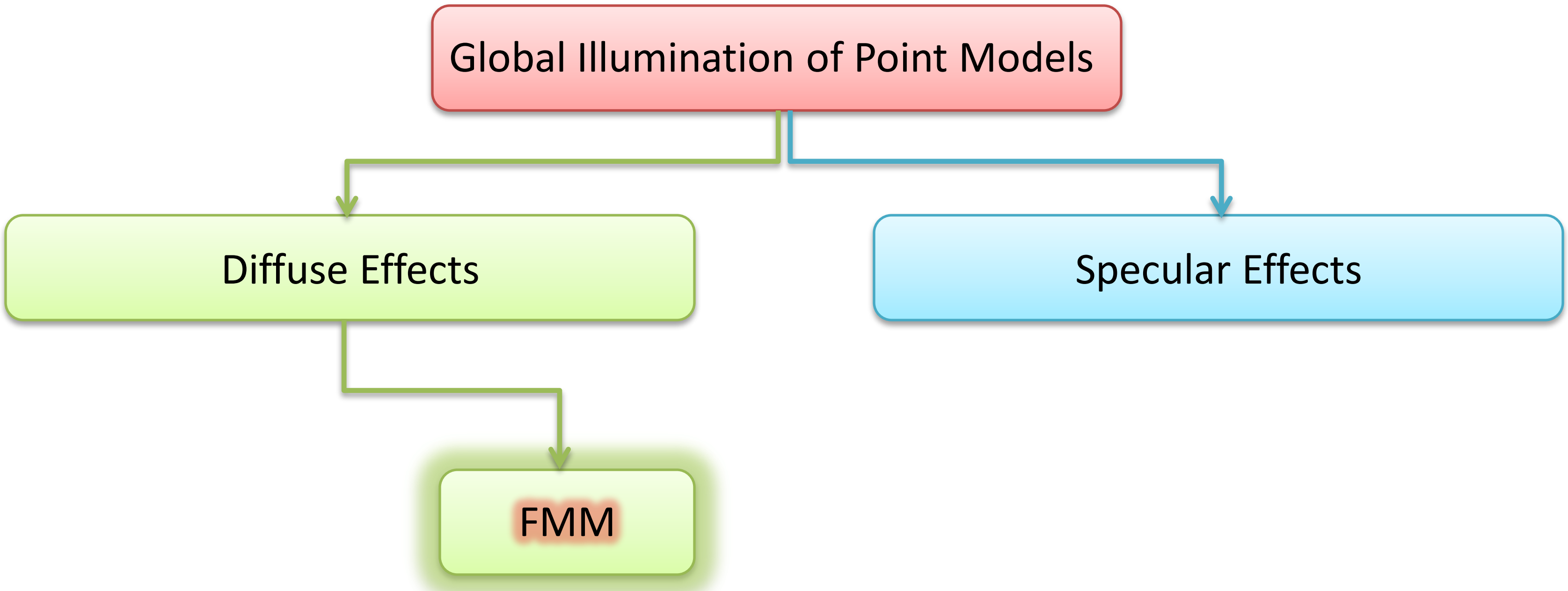


- Radiosity based GI Algorithm
- A *N-body* problem
- Size of Point Models: Hundreds of Thousands

  Not Practical to Implement with such high time complexity



- Fast Multipole Method (FMM) reduces the $O(N^2)$ time complexity to $O(N)$
- Follows Factorization and a Hierarchical Structure

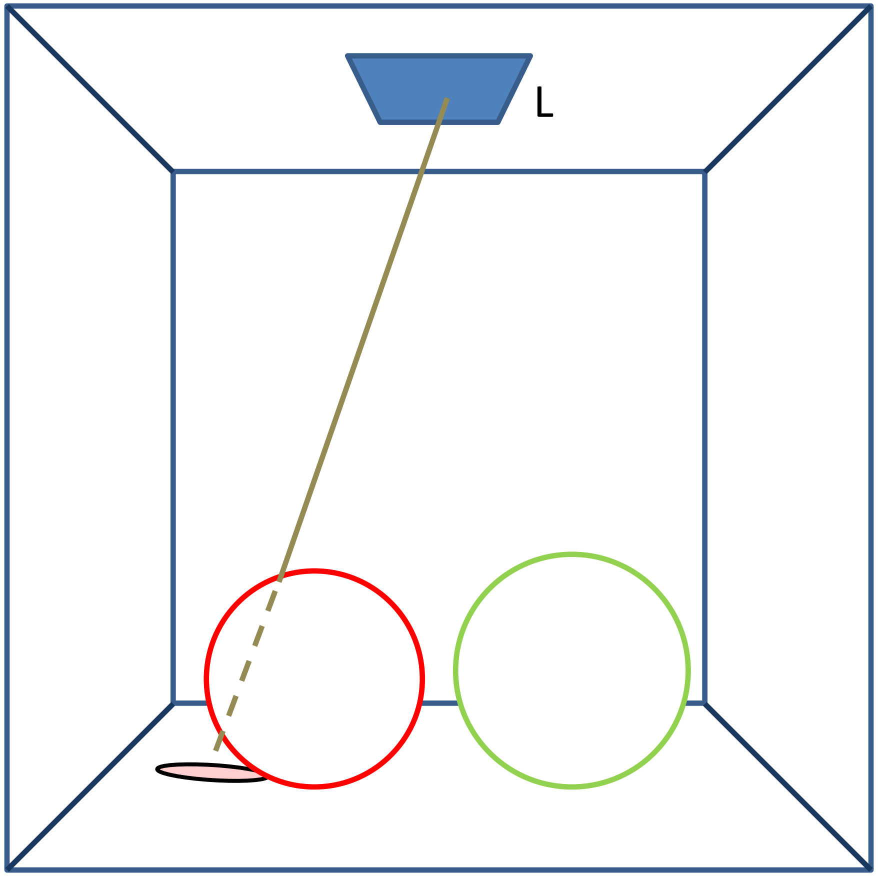


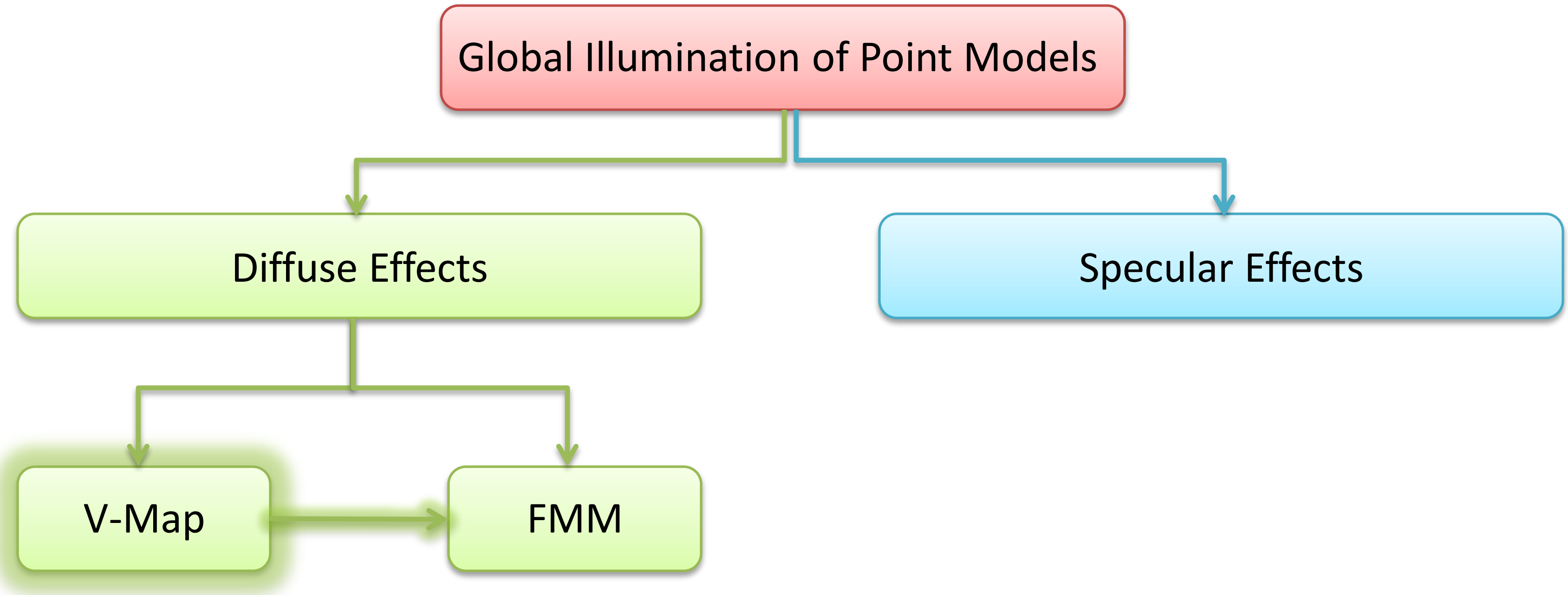
- Fast Multipole Method (FMM) reduces the $O(N^2)$ time complexity to $O(N)$
- Follows Factorization and a Hierarchical Structure

- **Work done in my 2nd year of PhD**

Visibility Between Point-Pairs

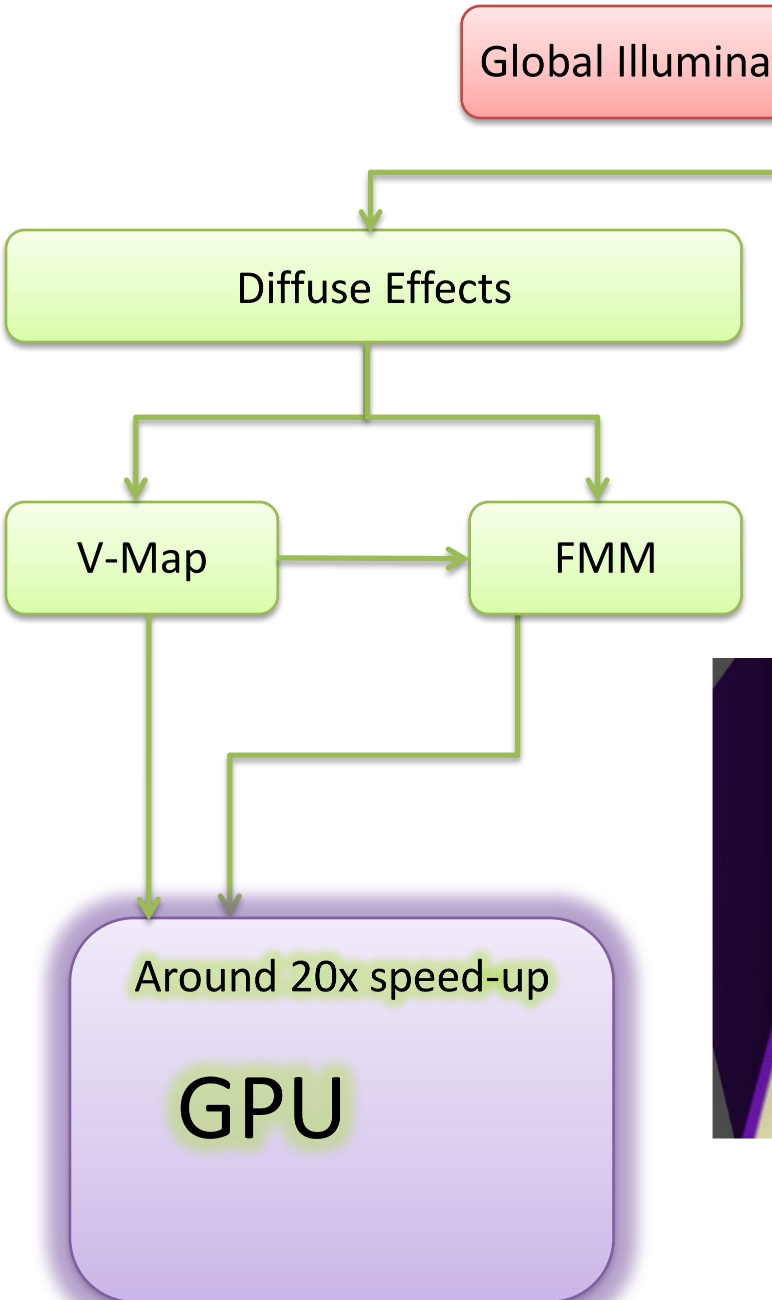
- Visibility calculation between point pairs is essential to give **correct GI** results as a point receives energy from other point only if it is **visible** – *O(N³)*



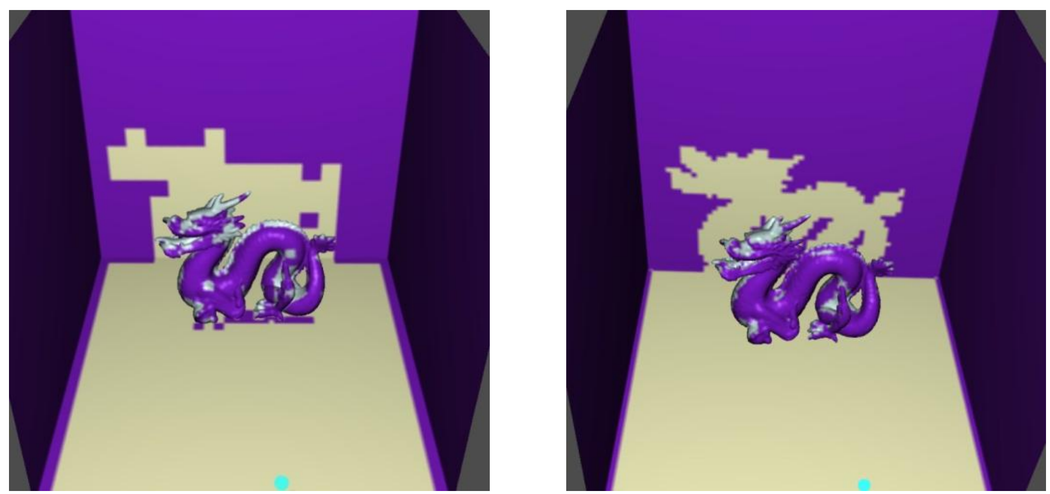


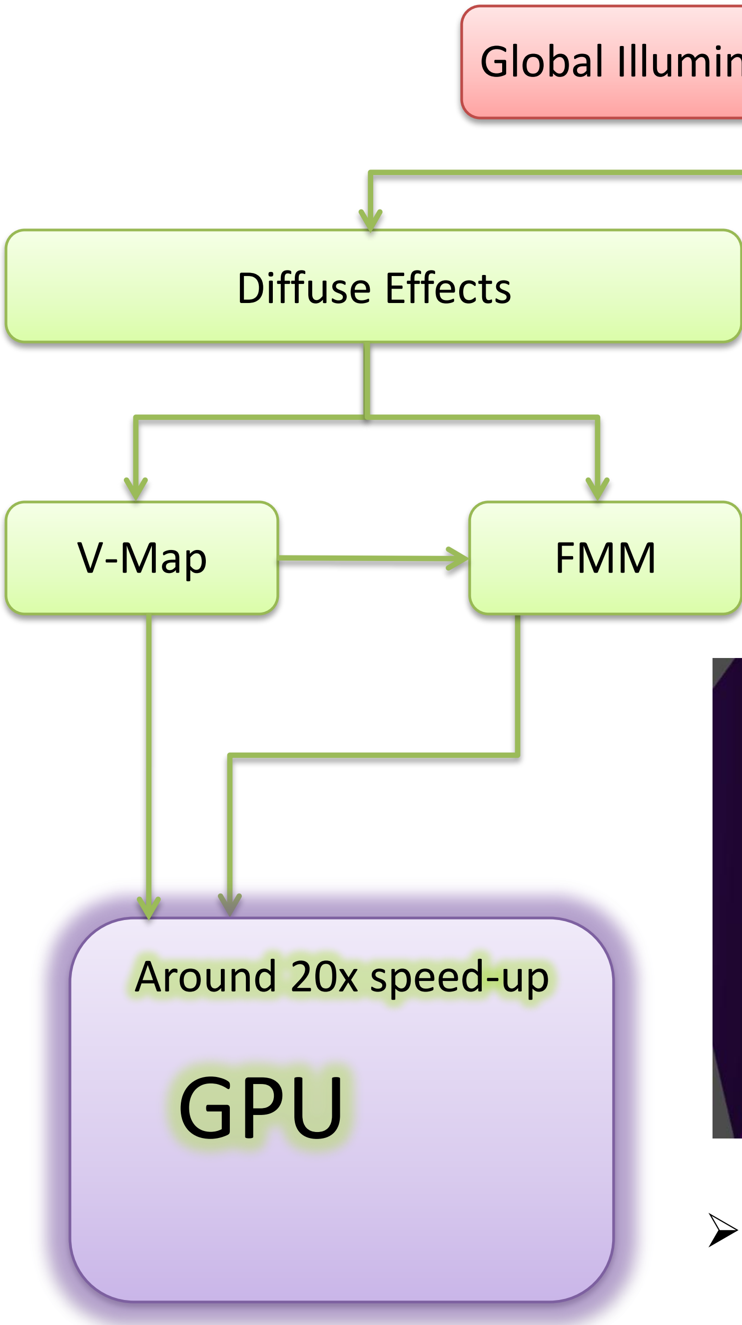
- Visibility Map (V-Map) gives a view-independent, hierarchical visibility solution for any given scene

- **Work done in my 3rd year of PhD**
- **[GKSD 07], Visibility Map for Global Illumination in Point Clouds, GRAPHITE 2007**

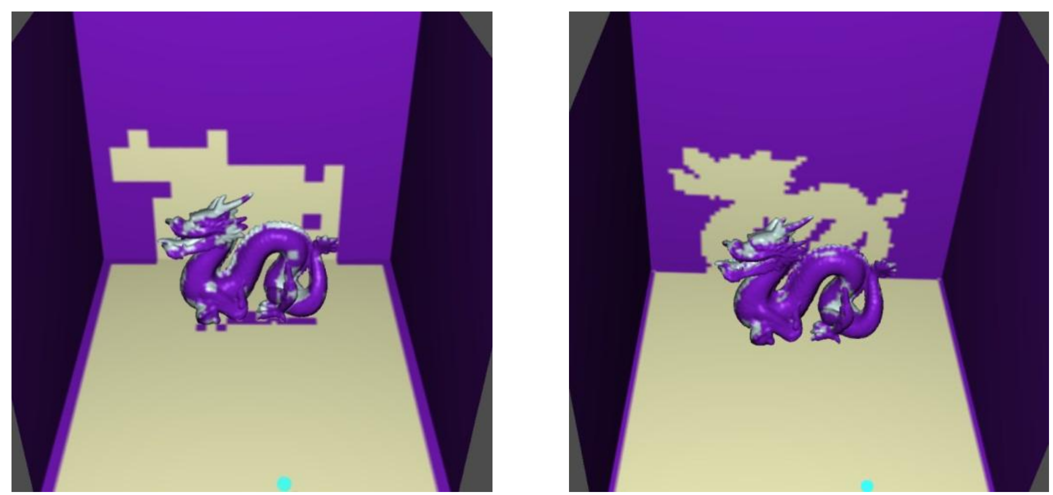


… but still not fast enough !

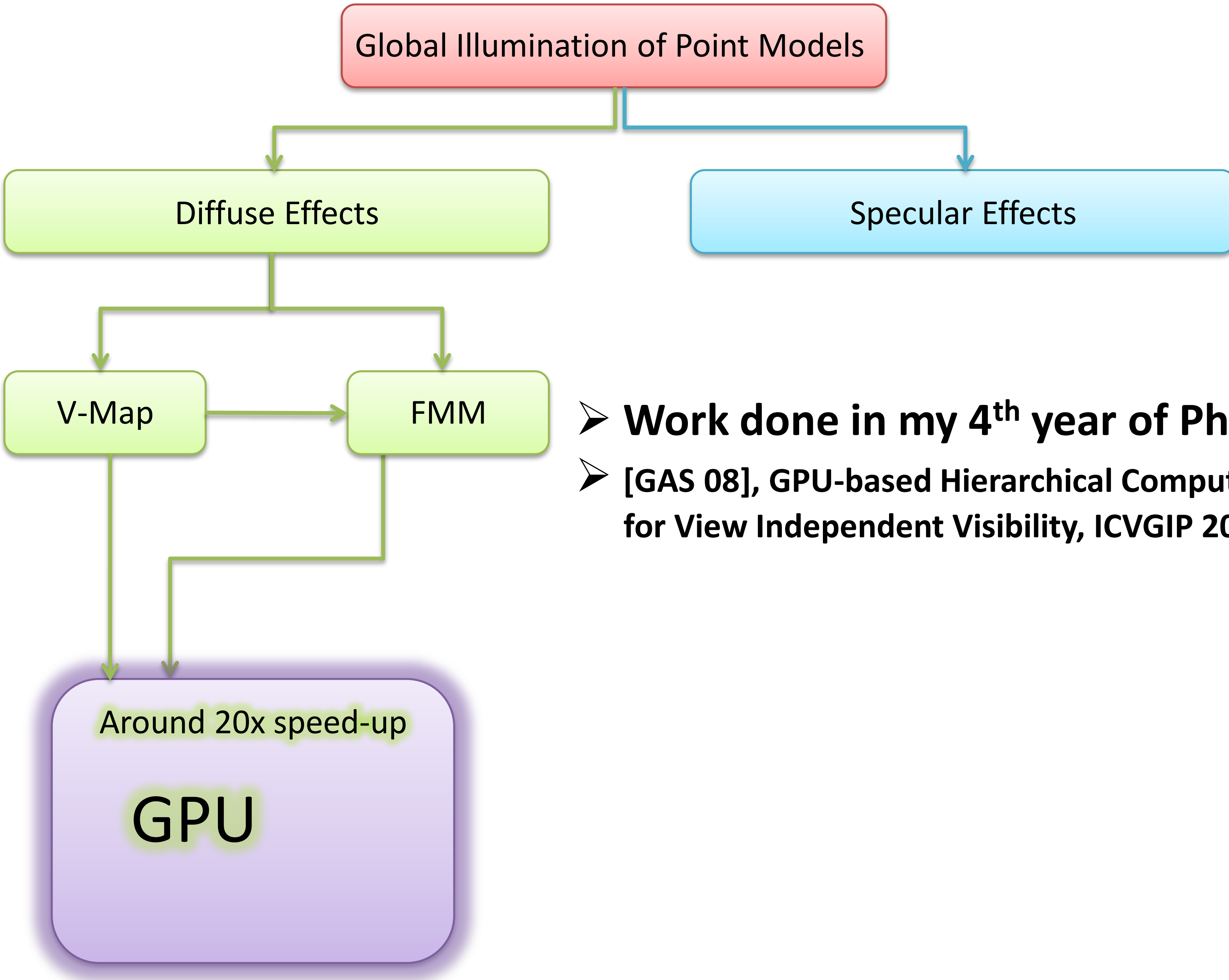




... but still not fast enough !



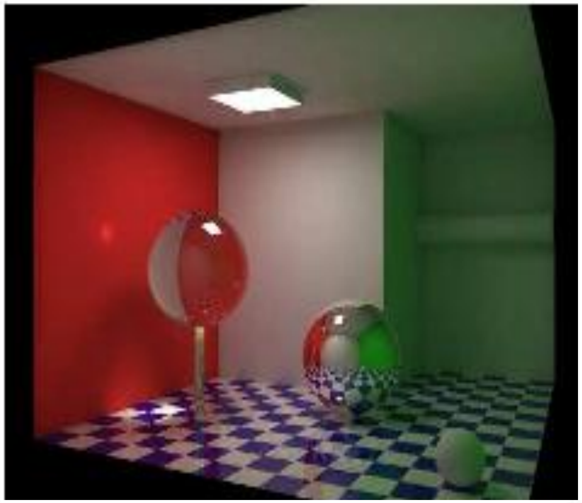
- **Running times reduced from hours to minutes**



- **Work done in my 4th year of PhD**
- **[GAS 08], GPU-based Hierarchical Computations for View Independent Visibility, ICVGIP 2008**

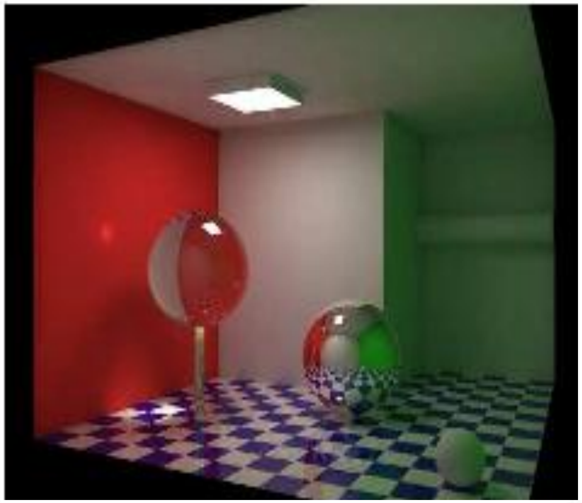
Plan

- Introduction – *Problem Definition*
- Diffuse Effects – *Overview*
- Specular Effects – *Details*
- Results
- Wrap-up



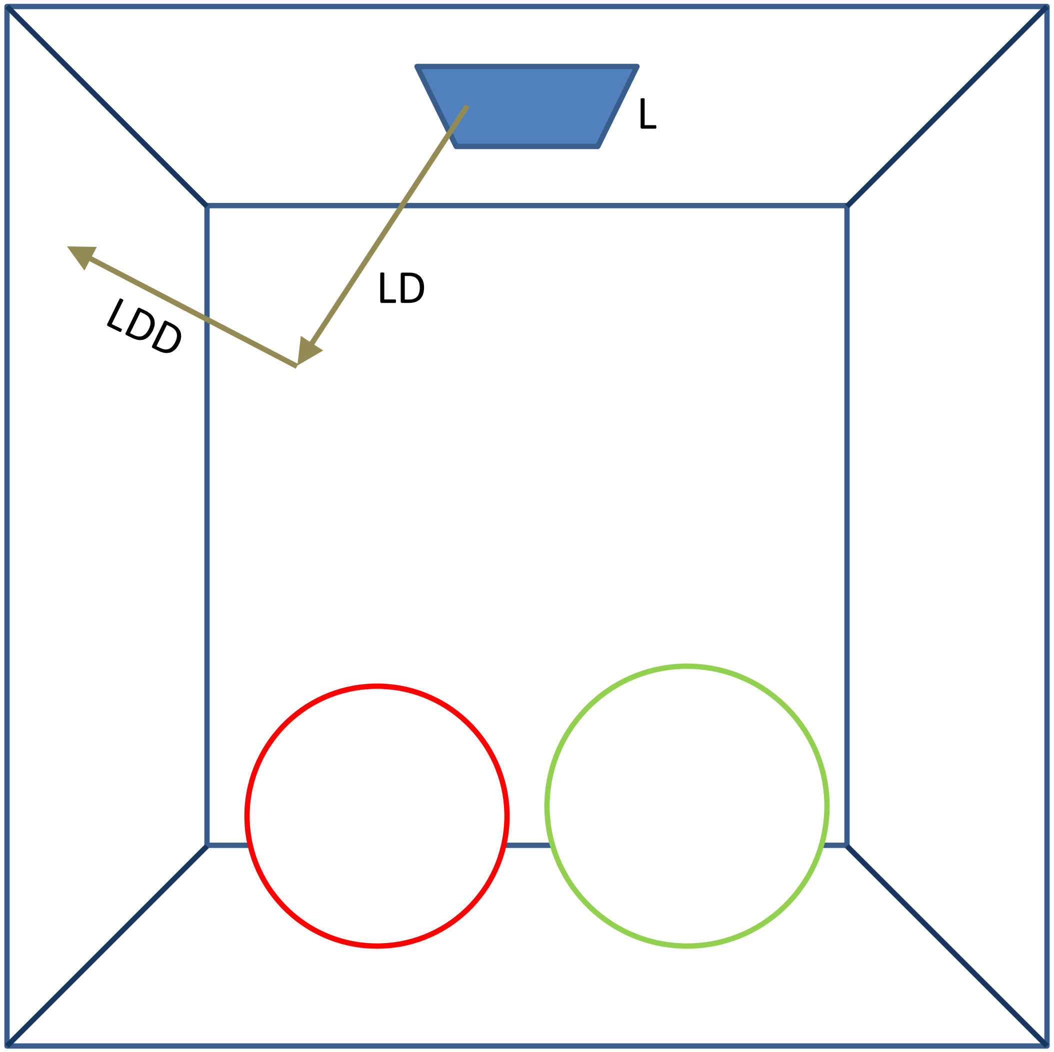
Plan

- Introduction – *Problem Definition*
- Diffuse Effects – *Overview*
- Specular Effects – *Details*
  - *How it affects diffuse interactions?*
  - *Details on generating specular effects*
- Results
- Wrap-up

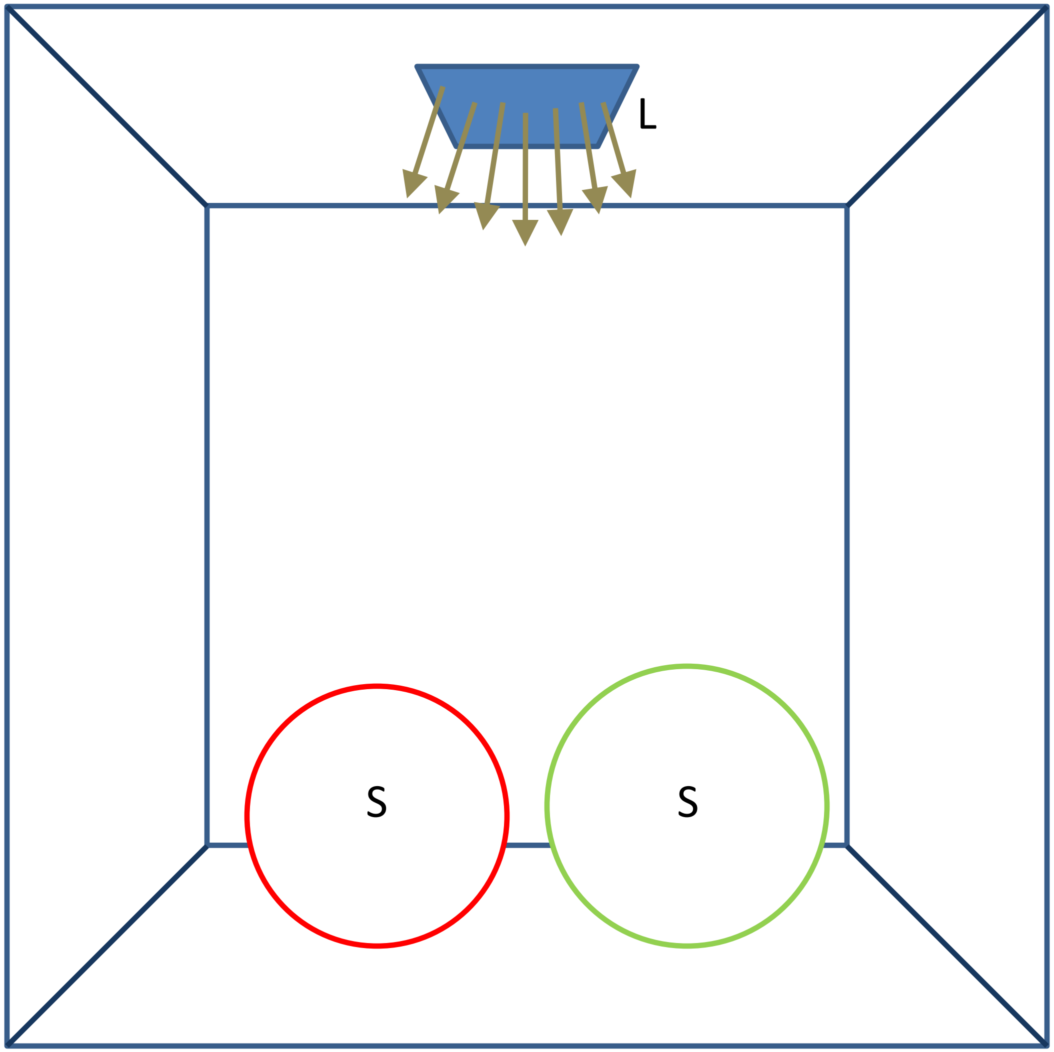


Diffuse Interactions

- LDD/LD$^+$ path

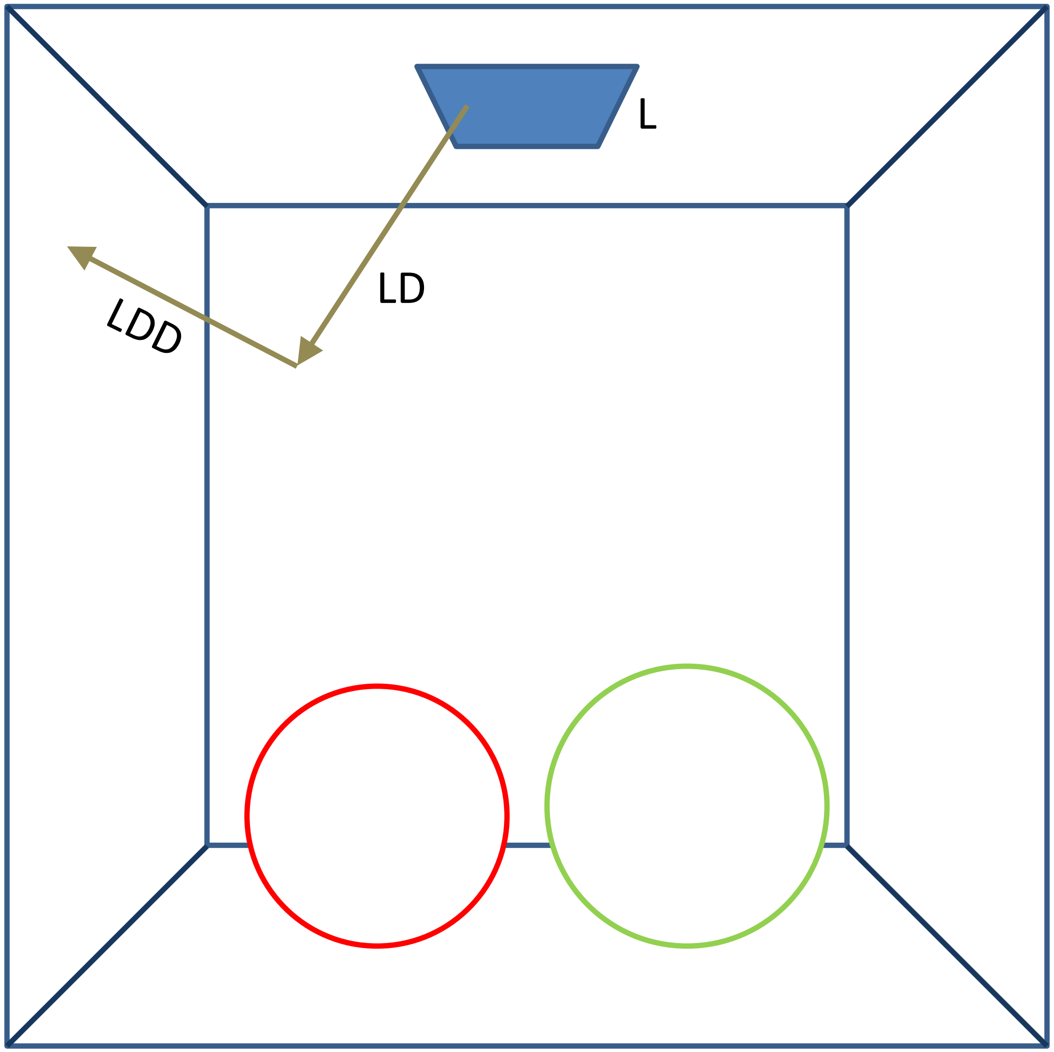


Diffuse-Specular Scene



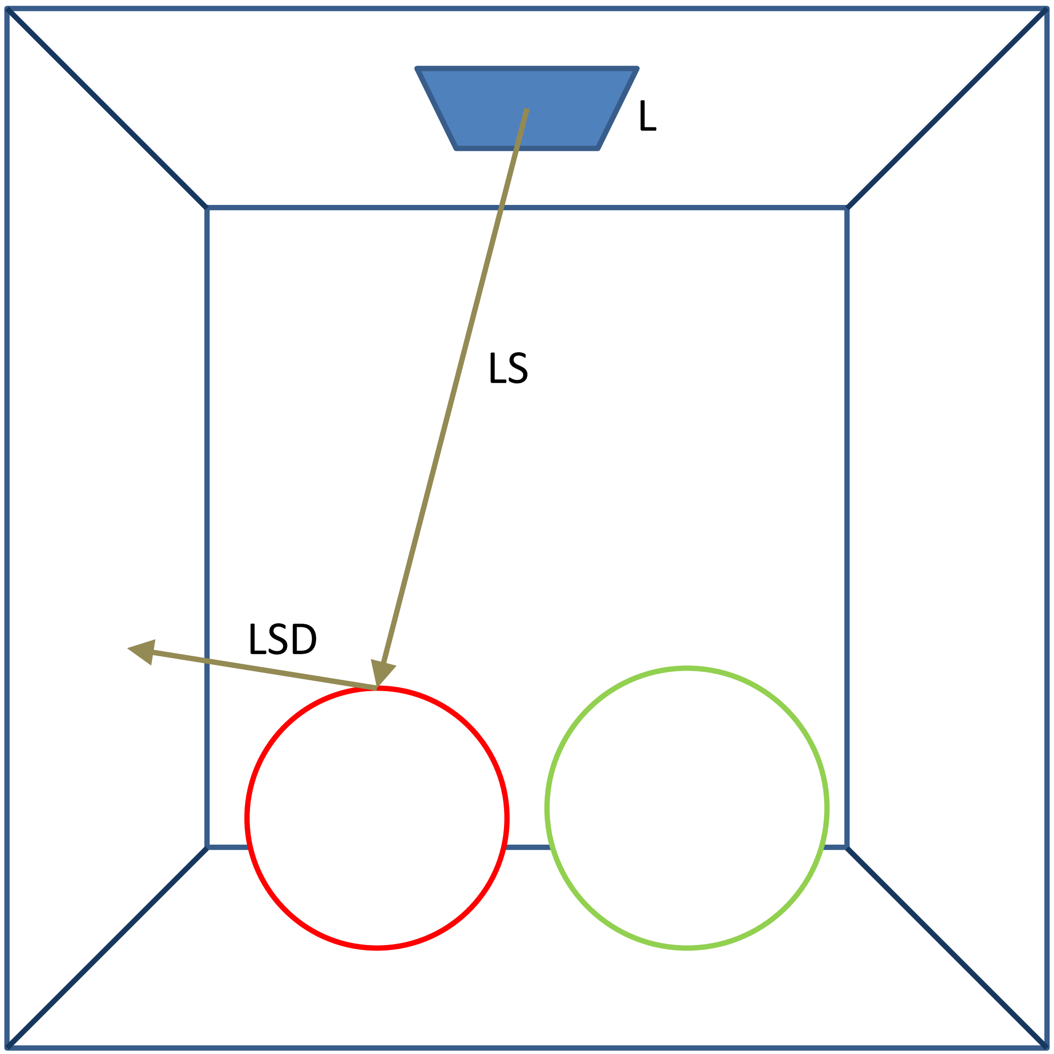
Diffuse-Specular Scene

- All possible paths taken up by light
  - LDD/LD^+
  - LSD/LS^+D
  - LDS/LD^+S^+D
  - LSS



Diffuse-Specular Scene

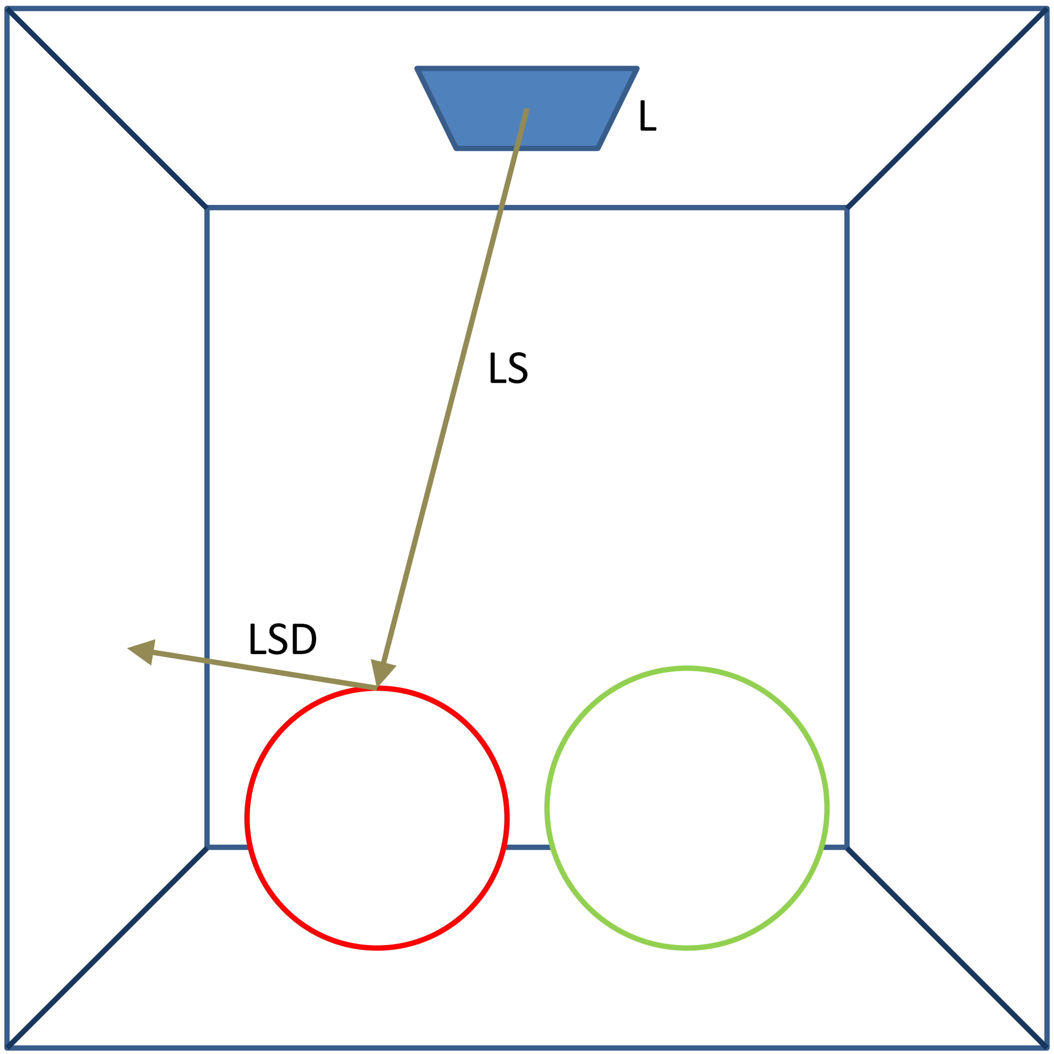
- All possible paths taken up by light
  - LDD/LD$^+$
  - LSD/LS$^+$D
  - LDS/LD$^+$S$^+$D
  - LSS



Diffuse-Specular Scene

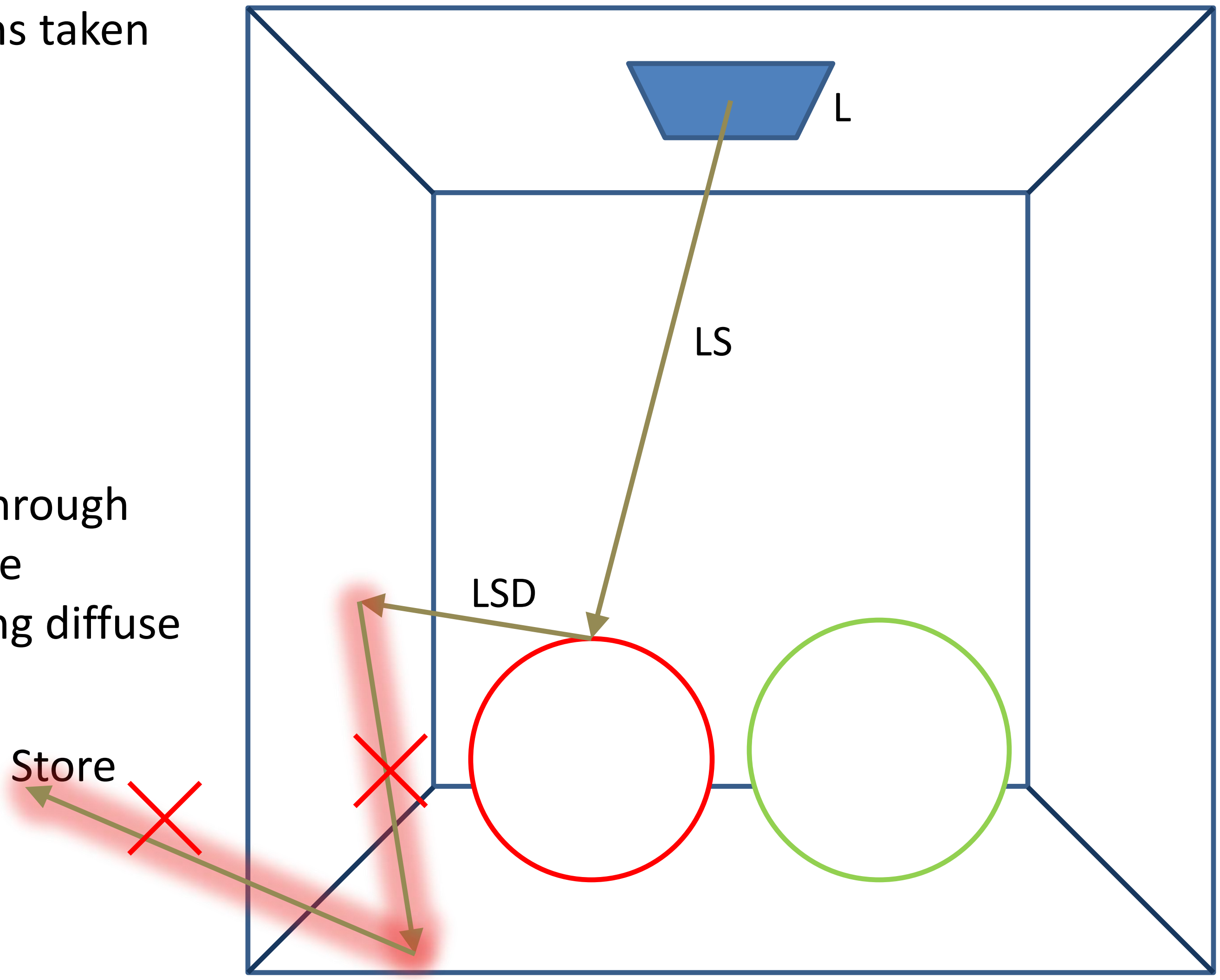
- All possible paths taken up by light
  - LDD/LD+
  - LSD/LS+D
  - LDS/LD+S+D
  - LSS

- Light received through LSD path must be distributed during diffuse inter-reflections



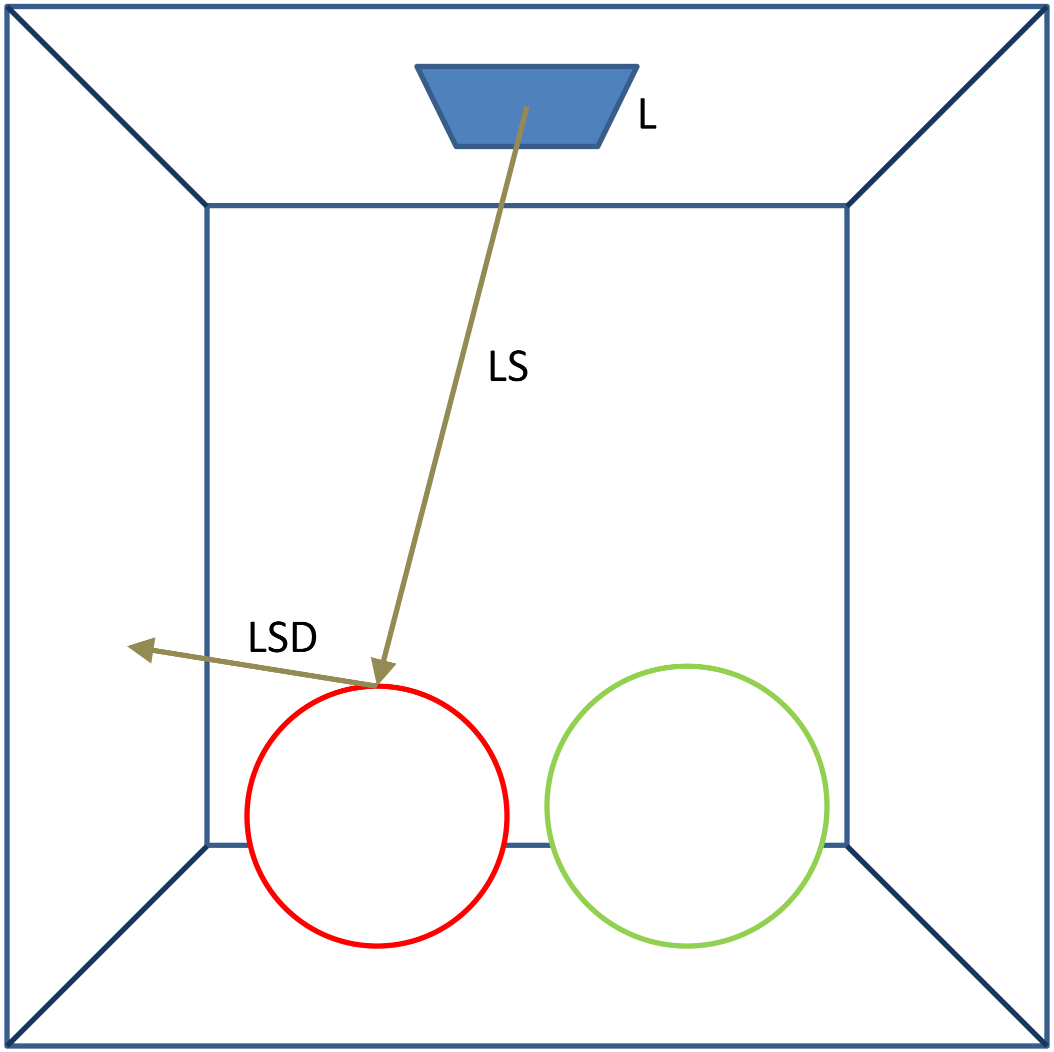
Diffuse-Specular Scene

- All possible paths taken up by light
  - LDD/LD$^+$
  - LSD/LS$^+$D
  - LDS/LD$^+$S$^+$D
  - LSS
- Light received through LSD path must be distributed during diffuse inter-reflections
- Pre-process and Store



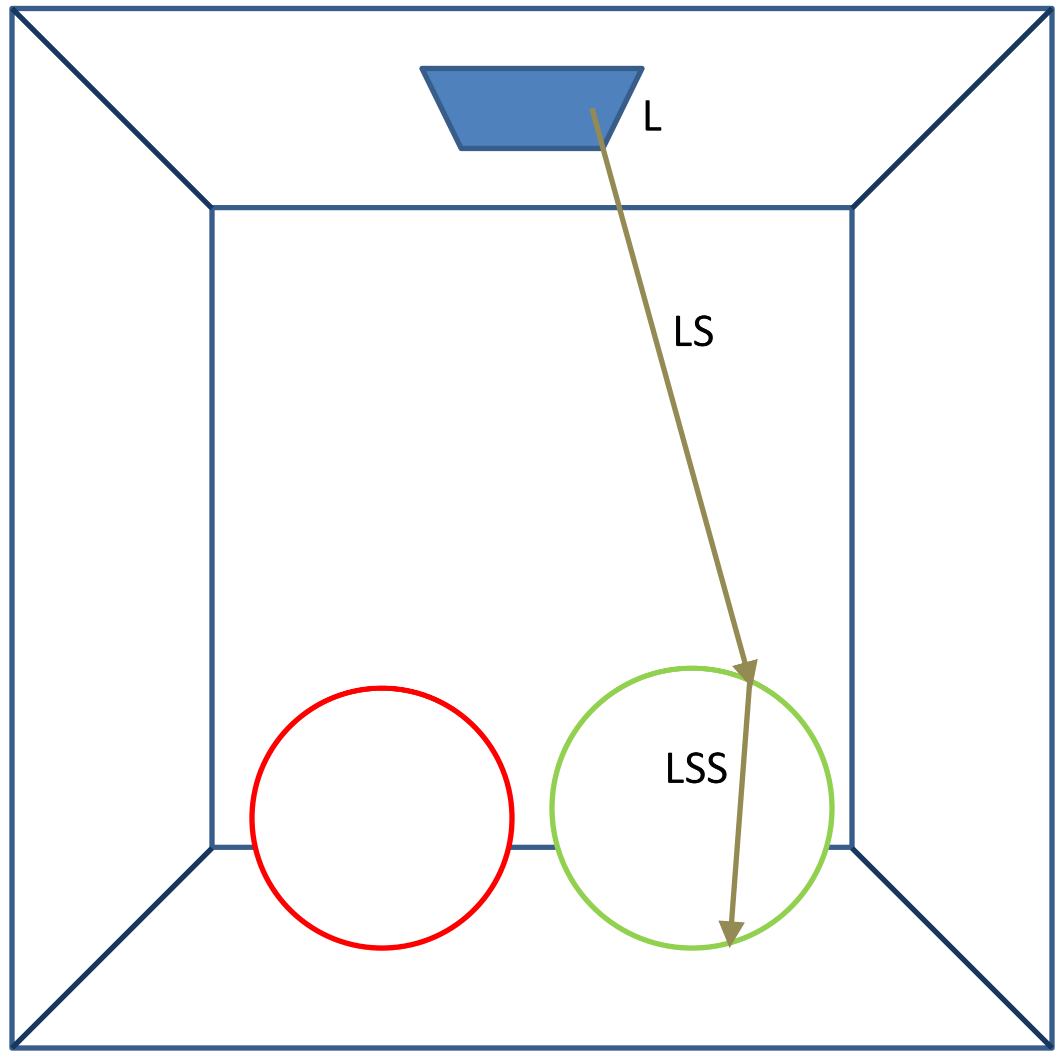
Diffuse-Specular Scene

- All possible paths taken up by light
  - LDD/LD$^+$
  - LSD/LS$^+$D
  - LDS/LD$^+$S$^+$D
  - LSS
- Light received through LSD path must be distributed during diffuse inter-reflections
- Pre-process and Store
- LS$^+$D -- Caustics!



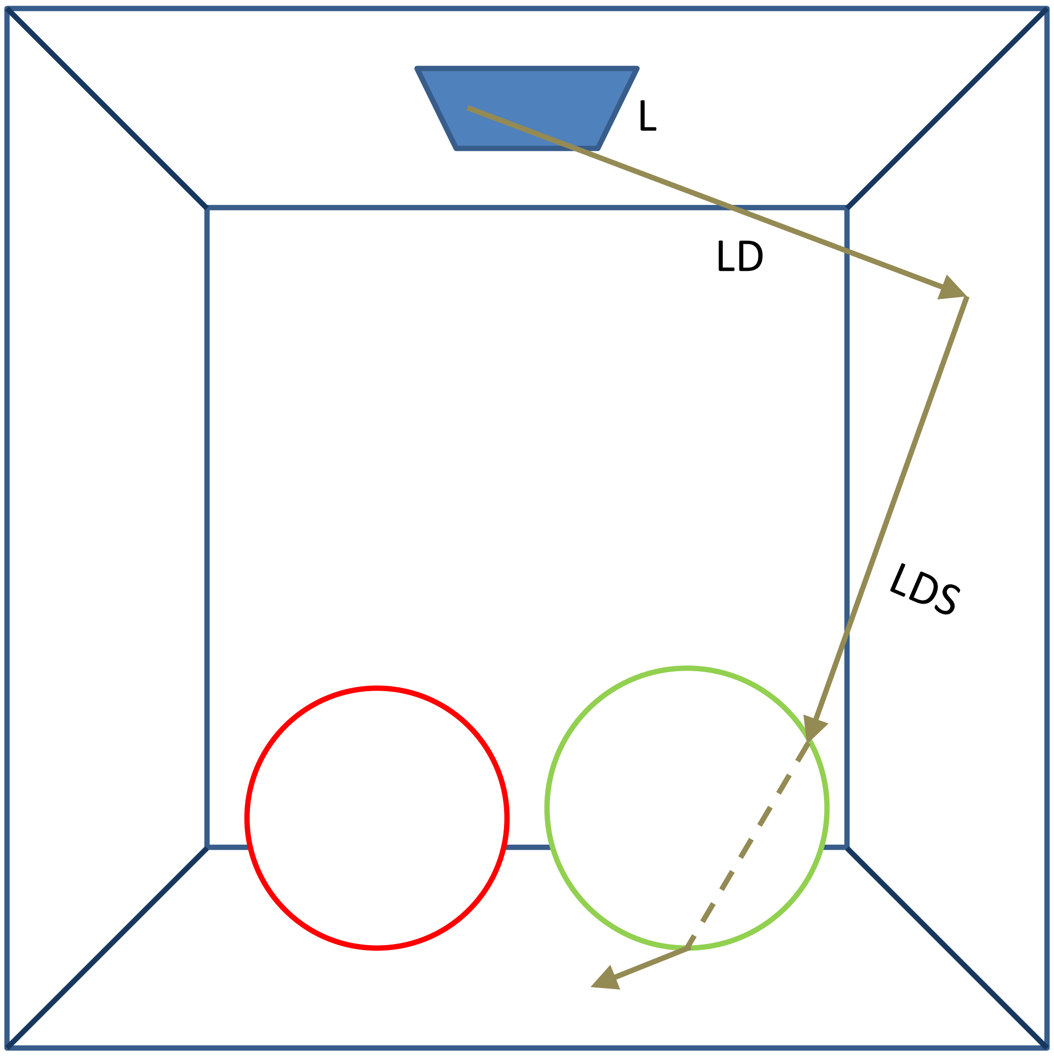
Diffuse-Specular Scene

- All possible paths taken up by light
  - LDD/LD$^+$
  - LSD/LS$^+$D
  - LDS/LD$^+$S$^+$D
  - LSS
- LSS similar to LS$^+$D
- Pre-processed Caustics!



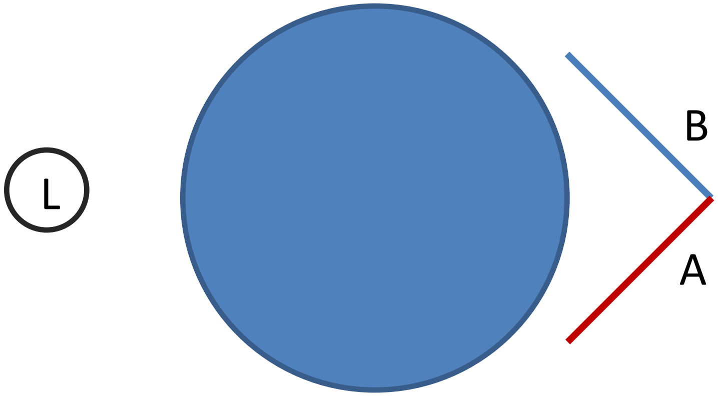
Diffuse-Specular Scene

- All possible paths taken up by light
  - LDD/LD+
  - LSD/LS+D
  - LDS/LD+S+D
  - LSS
- Time-consuming
- Energy transfer too low !
- We can do it, but temporarily Ignore these paths for the purpose of efficiency

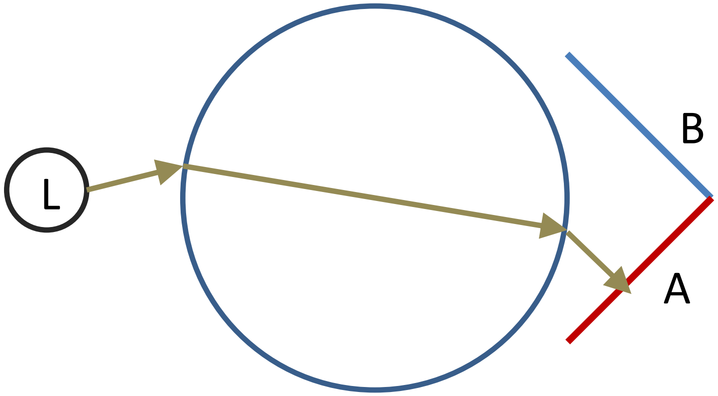


Diffuse-Specular Scene

- Diffuse case

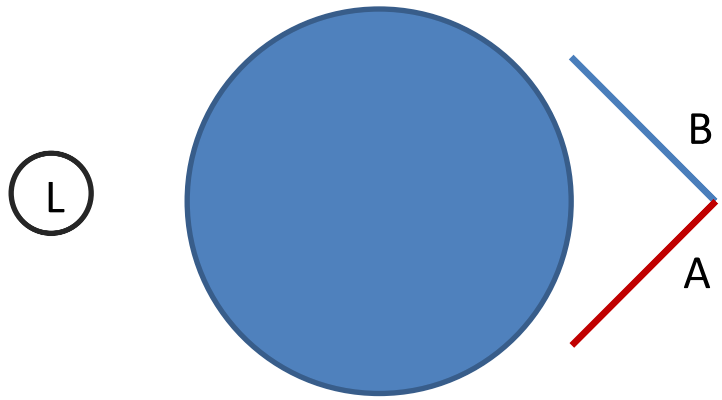


- Specular case

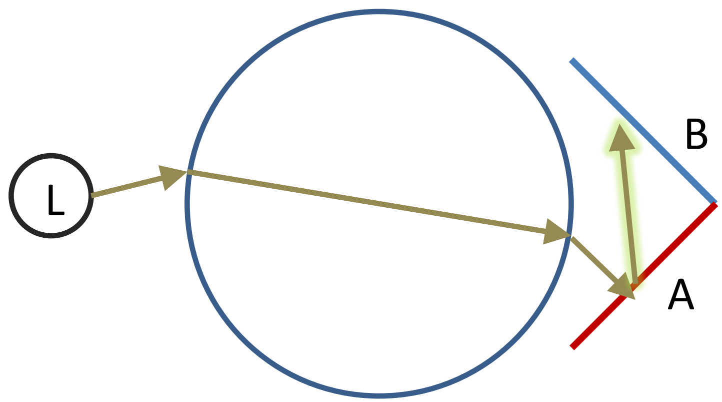


Diffuse-Specular Scene

- Diffuse case



- Specular case



- During FMM transfers, surfaces A and B will still be invisible to L !

Fusing Together: Diffuse and Specular Effects

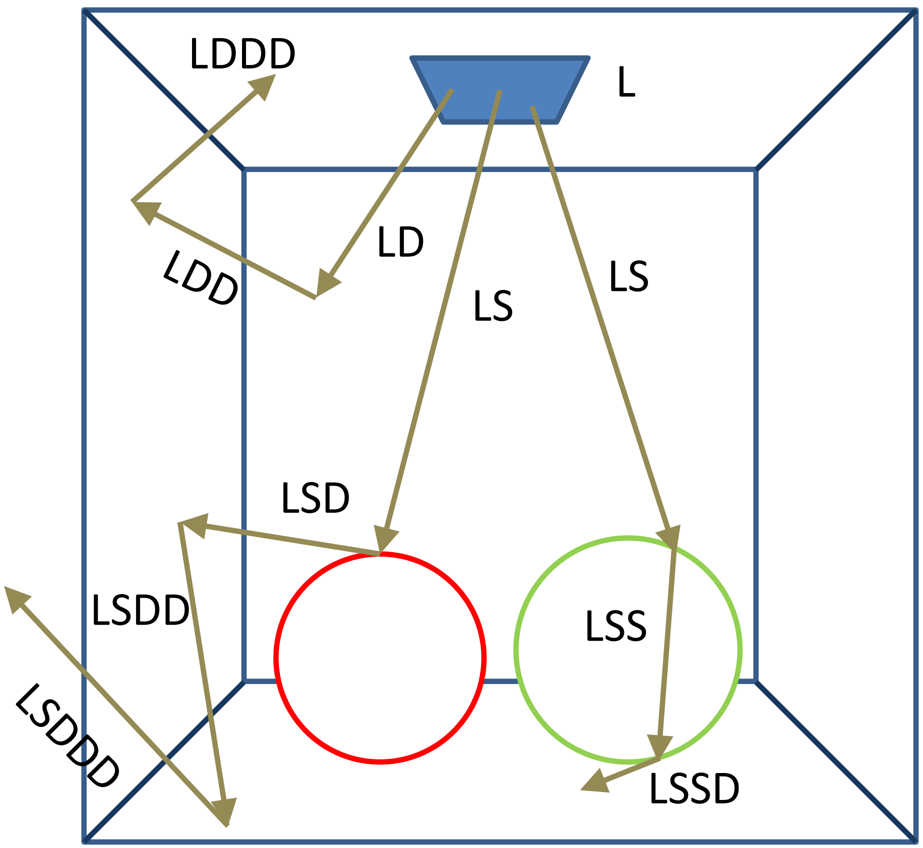
- $LS^{+}D$ handled by caustics
- FMM takes care of LD^{+} path
- FMM ignores contributions from specular splats ($LS^{+}D$)
- FMM does not transfer energy to specular splats ($LD^{*}S^{+}$)

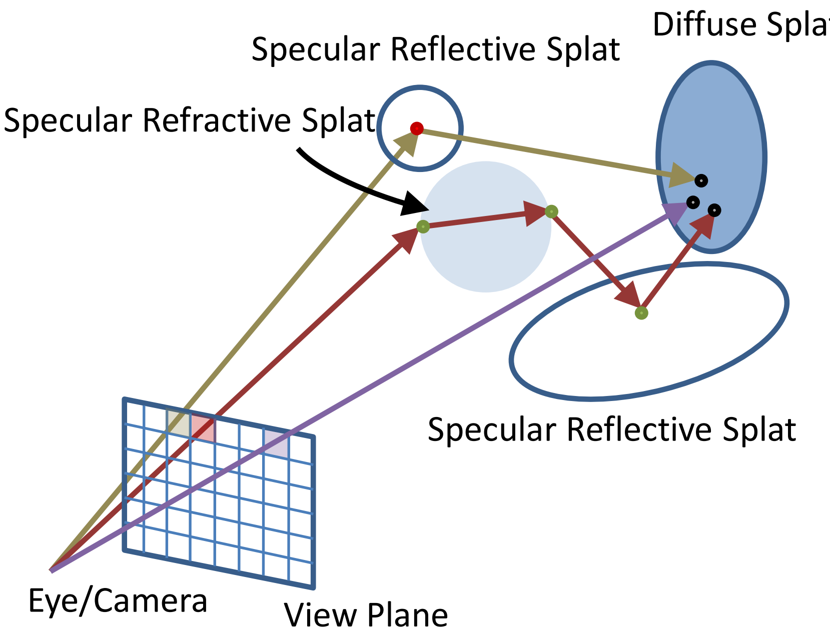
Fusing Together: Diffuse and Specular Effects

- *LS^+D* handled by caustics
- FMM takes care of *LD^+* path
- FMM ignores contributions from specular splats (*LS^+D*)
- FMM does not transfer energy to specular splats (*LD^*S^+*)

- *What about view-dependence? -- L(S|D)*E*

View-Independence v/s View-Dependence

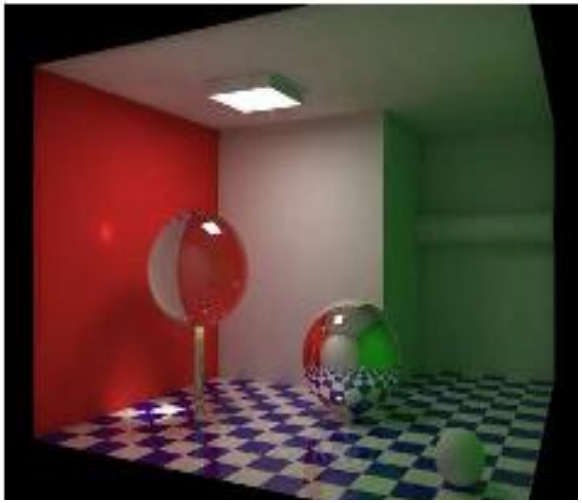


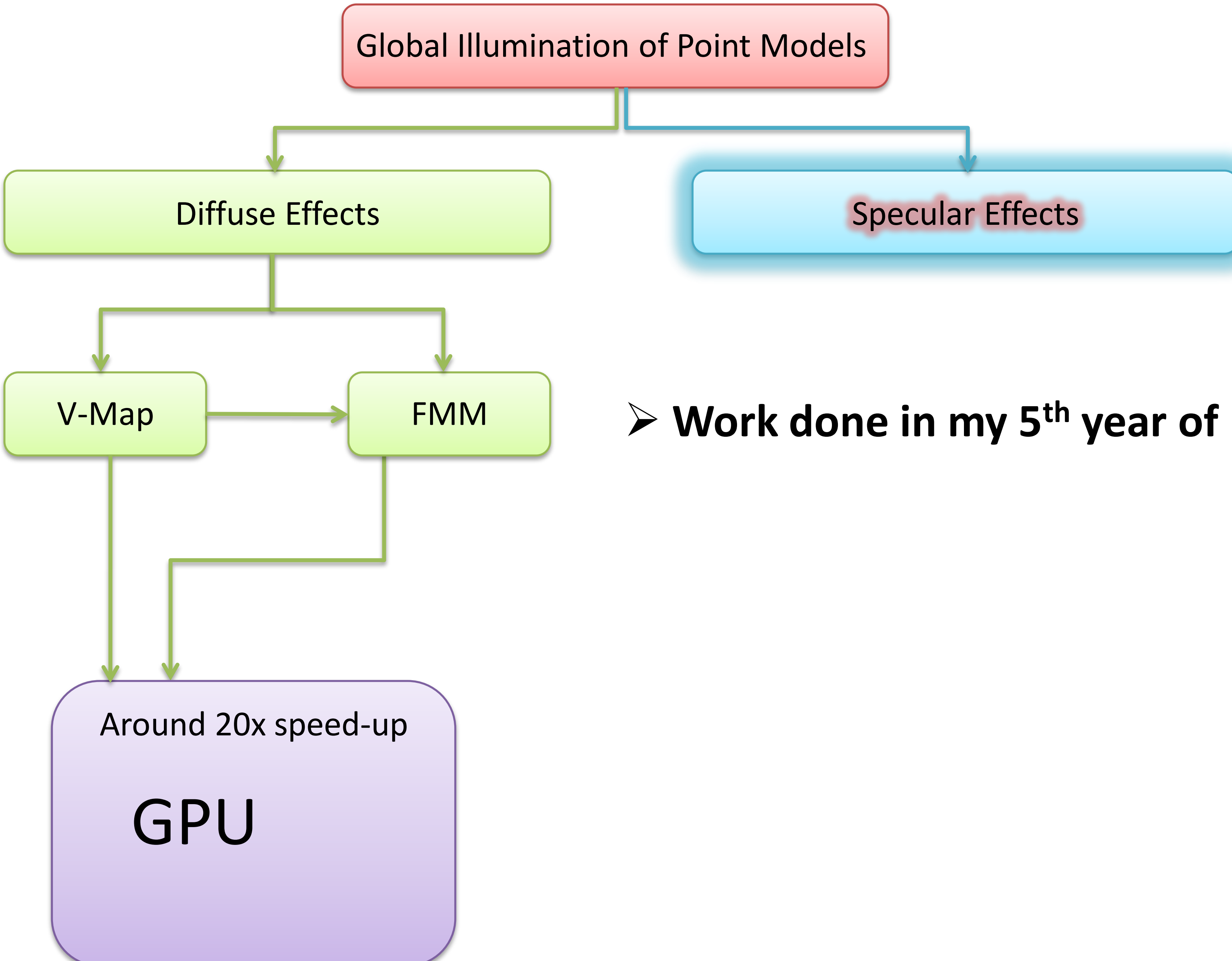


- *L(S|D)*E*

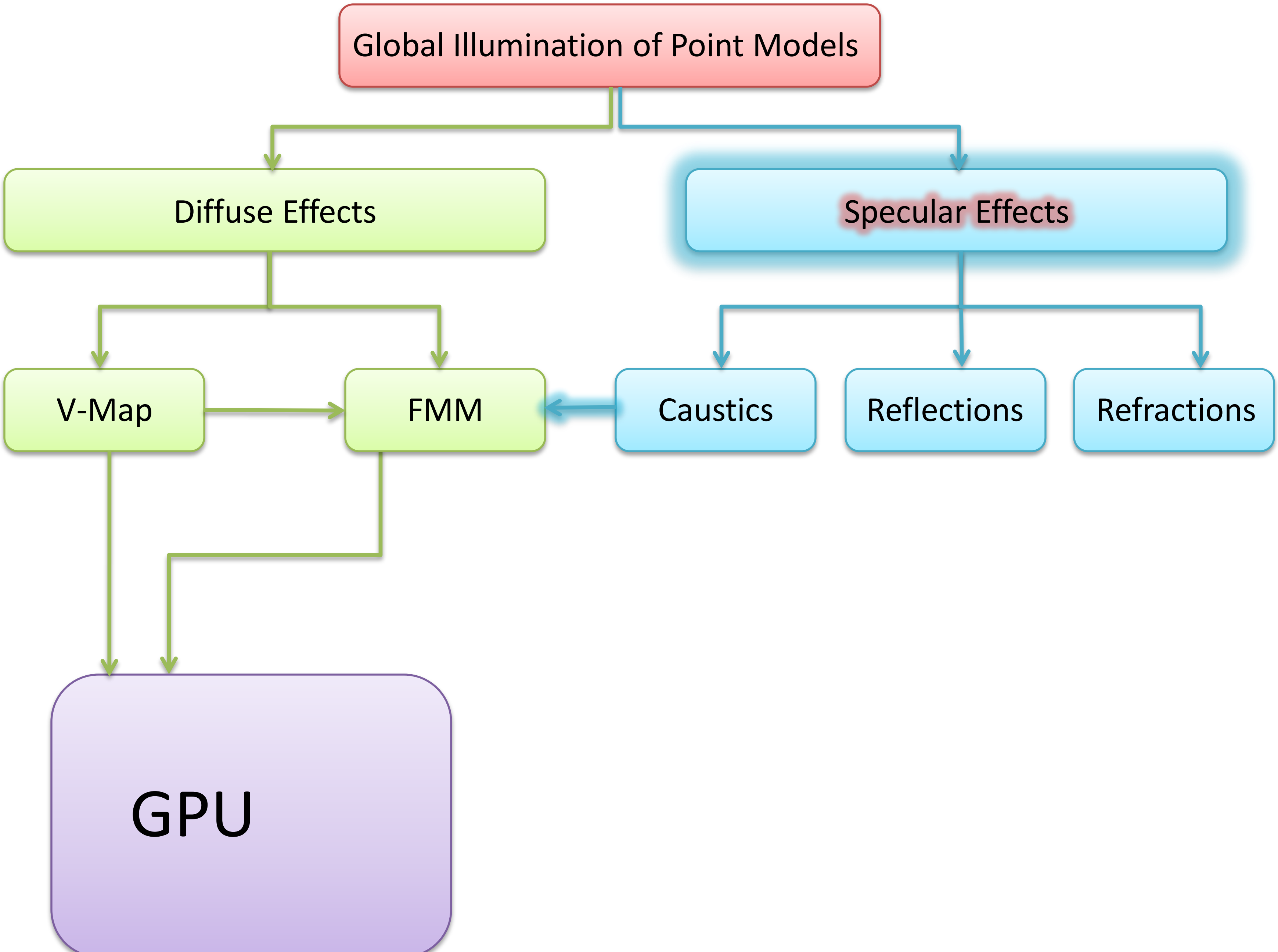
Plan

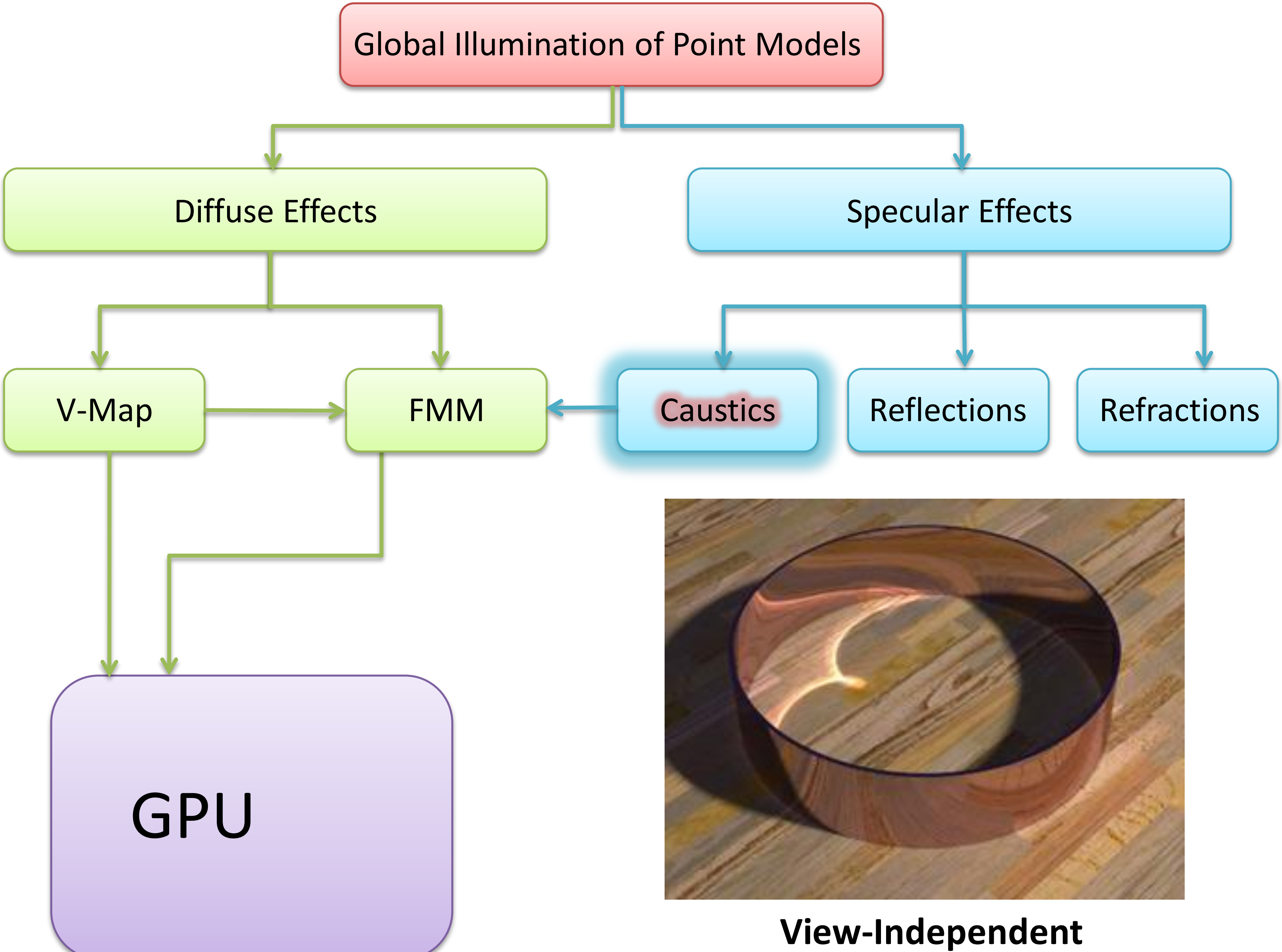
- Introduction – *Problem Definition*
- Diffuse Effects – *Overview*
- Specular Effects – *Details*
  - *How it effects diffuse interactions?*
  - *Details on generating specular effects*
- Results
- Wrap-up



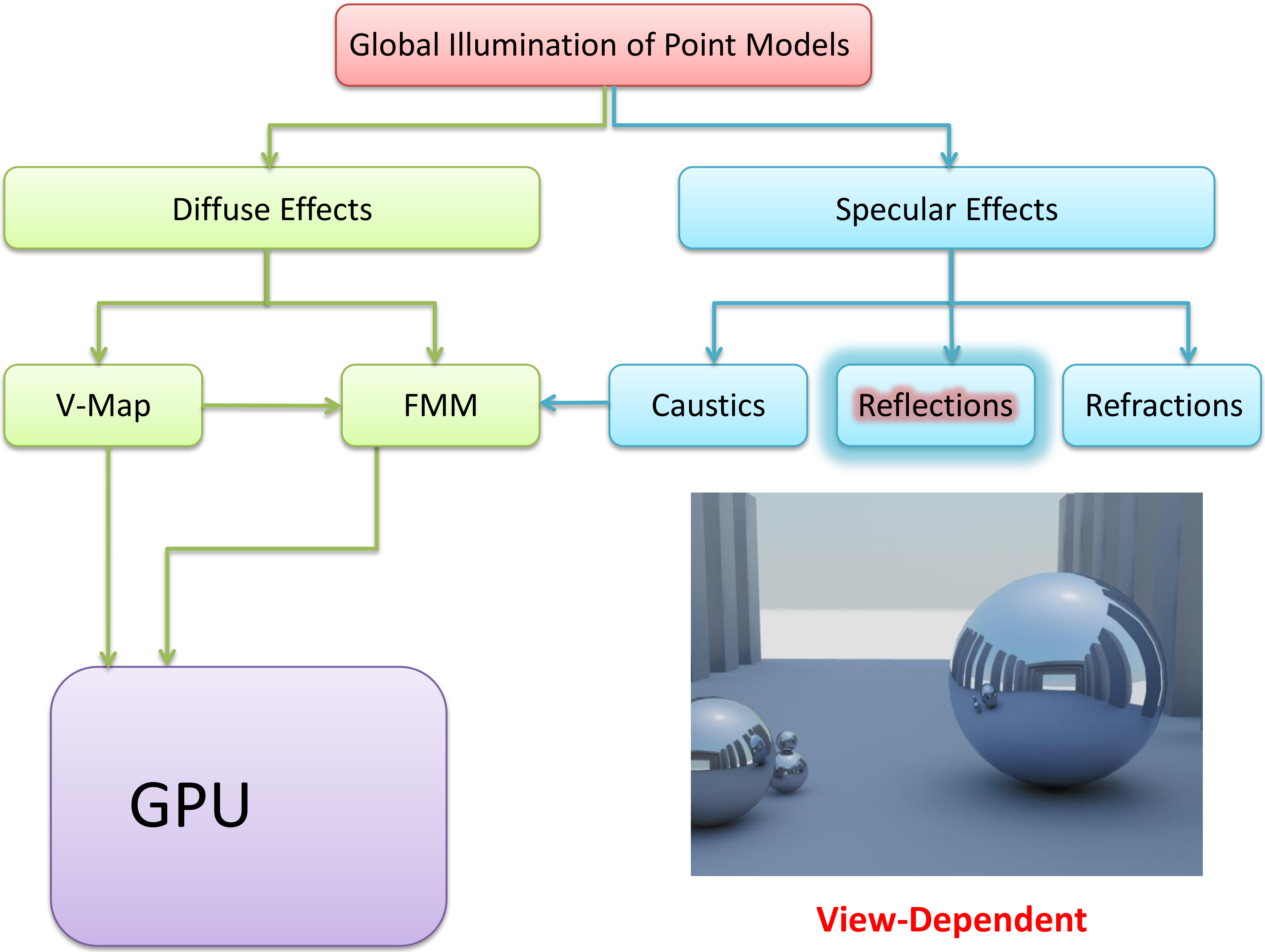


- **Work done in my 5th year of PhD**

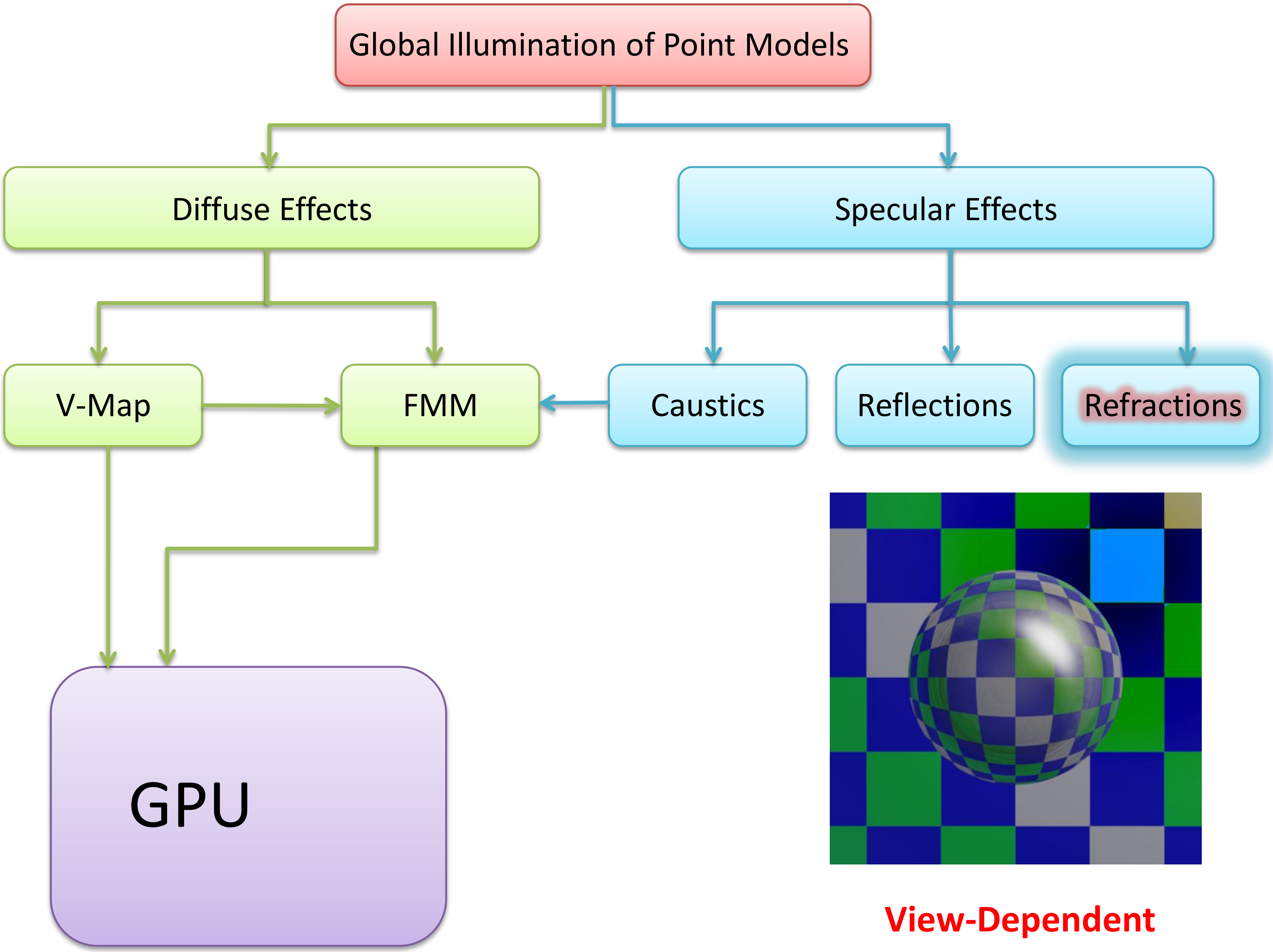

Global Illumination of Point Models
Diffuse Effects
Specular Effects
V-Map
FMM
Caustics
Reflections
Refractions
GPU



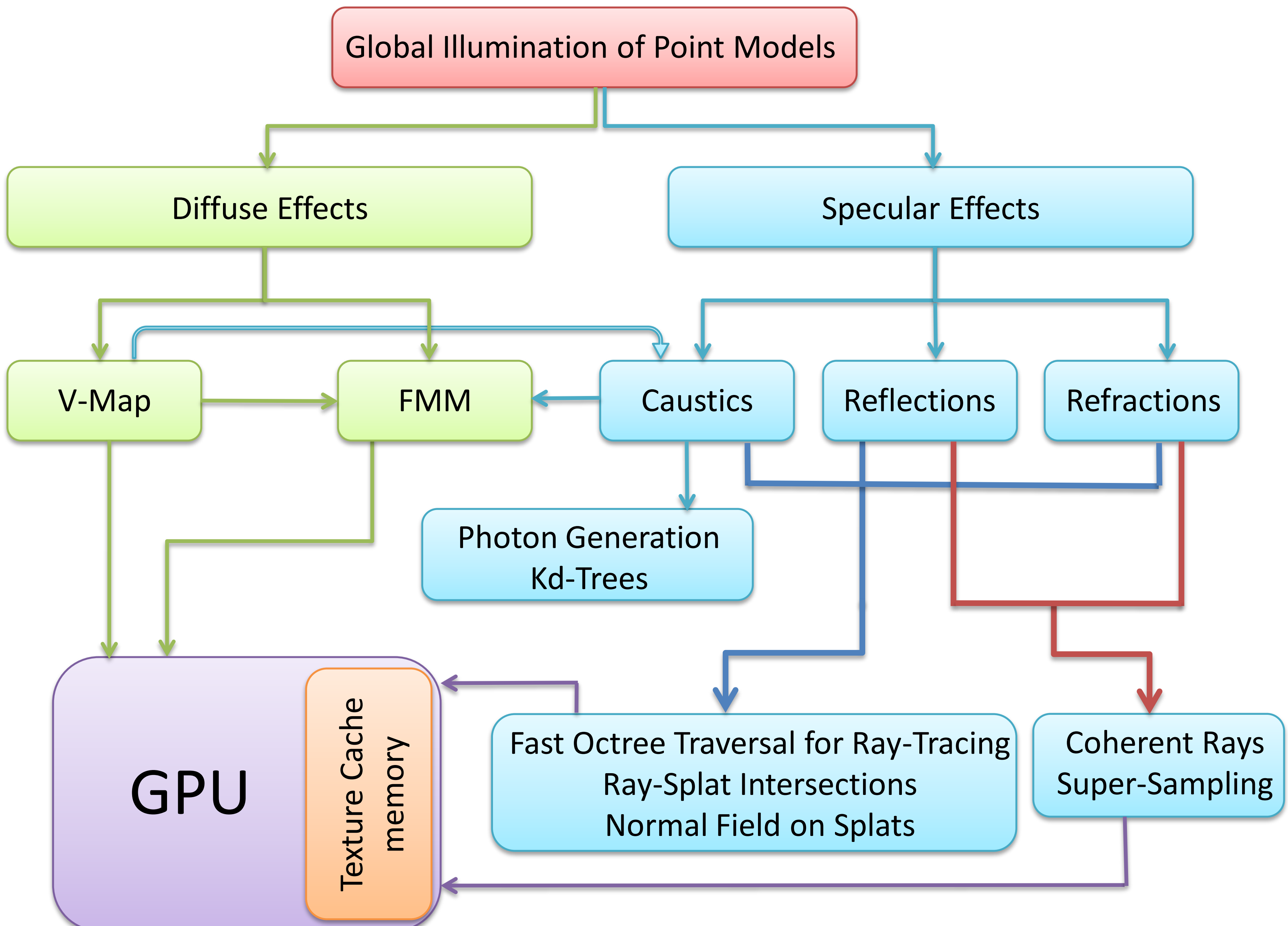
View-Independent

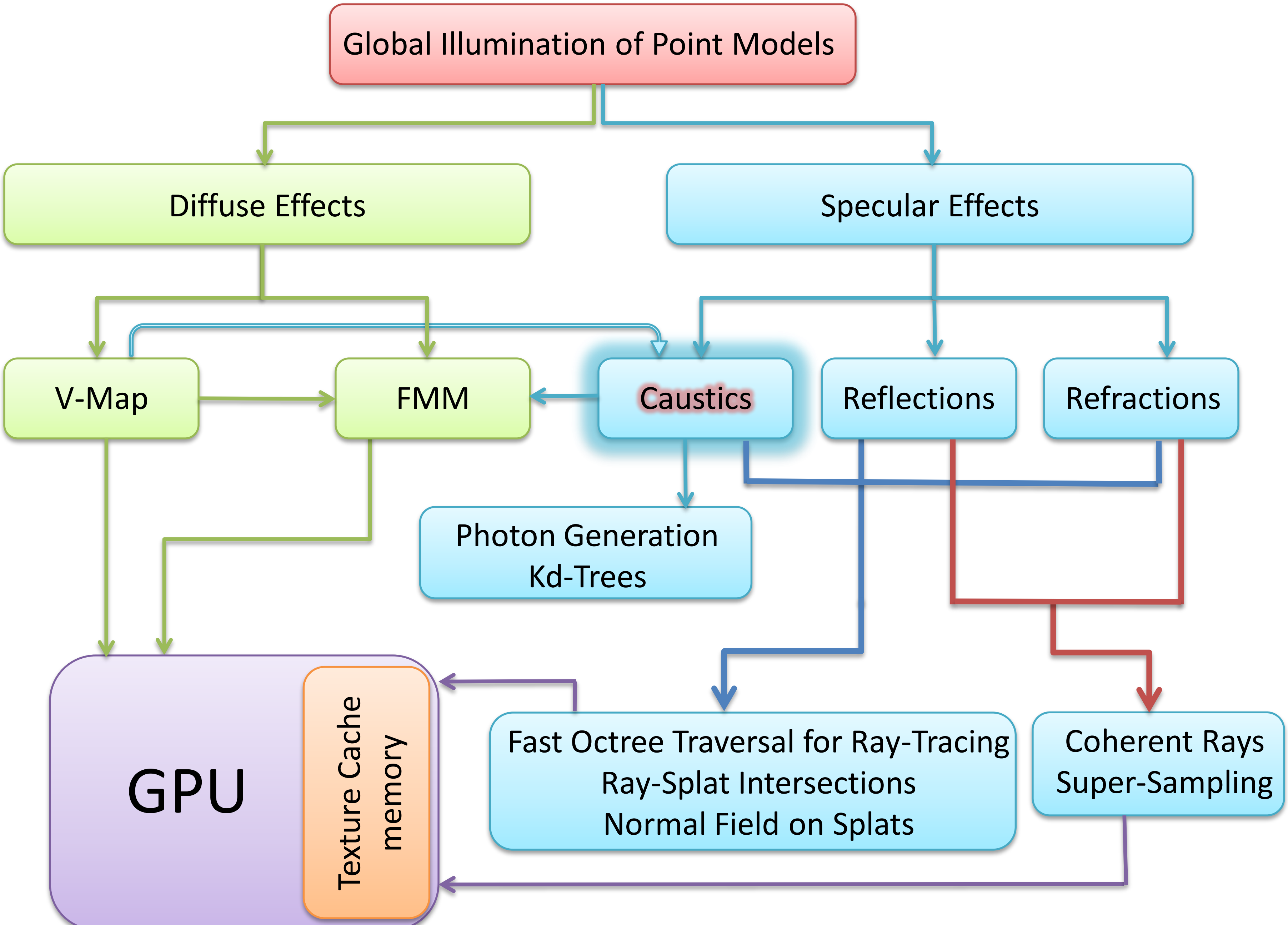


View-Dependent

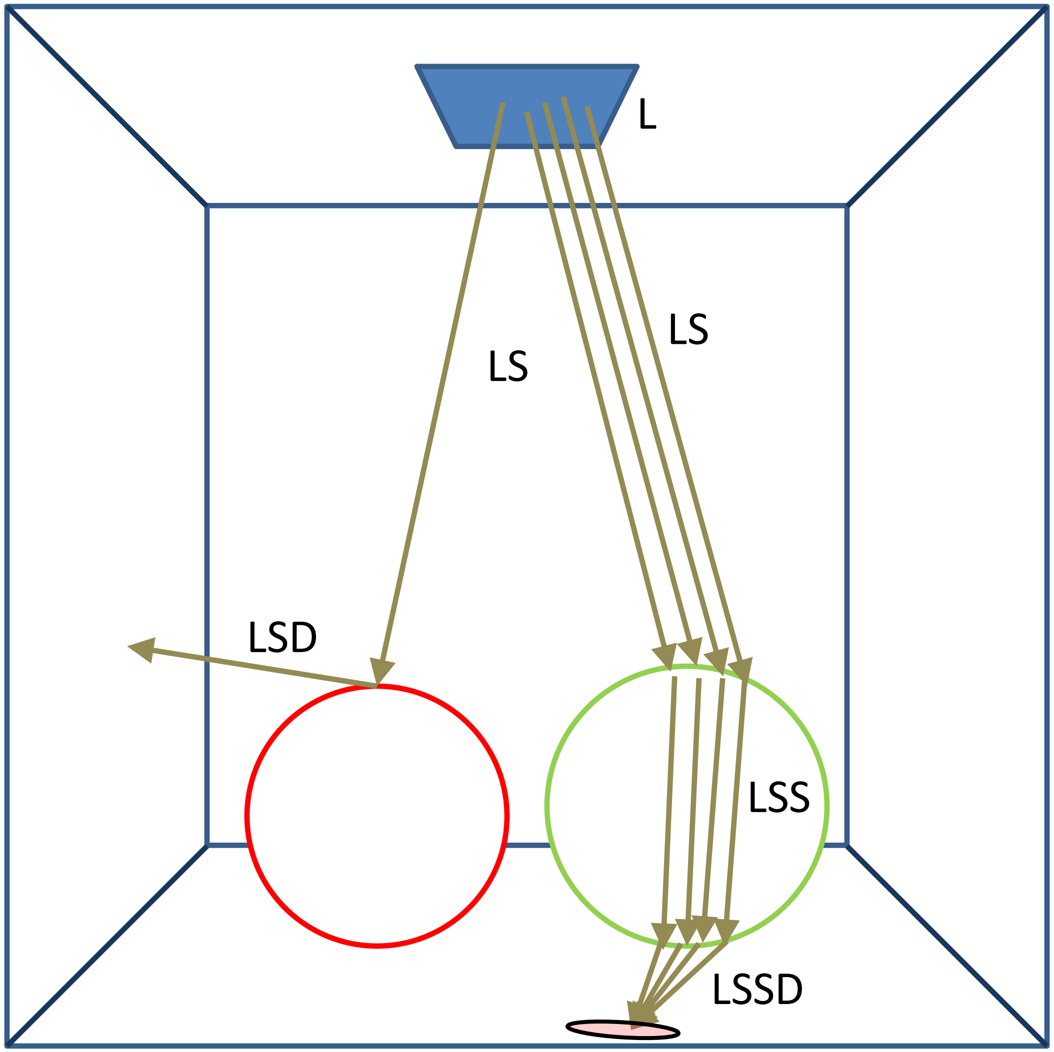


Global Illumination of Point Models
Diffuse Effects
Specular Effects
V-Map
FMM
Caustics
Reflections
Refractions
GPU
View-Dependent

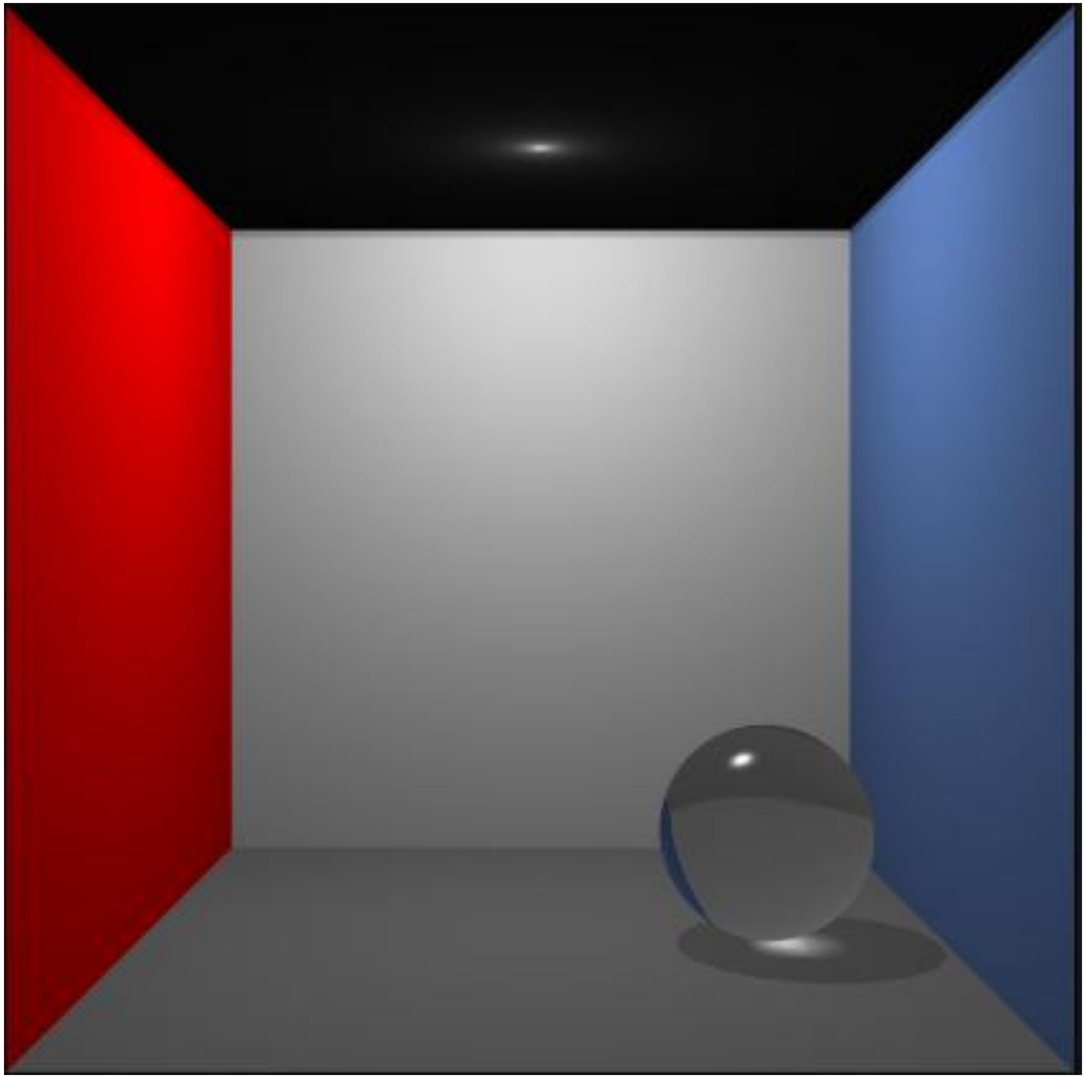

Global Illumination of Point Models
Diffuse Effects
Specular Effects
V-Map
FMM
Caustics
Reflections
Refractions
Photon Generation
Kd-Trees
GPU
Texture Cache memory
Fast Octree Traversal for Ray-Tracing
Ray-Splat Intersections
Normal Field on Splats
Coherent Rays
Super-Sampling


Global Illumination of Point Models
Diffuse Effects
Specular Effects
V-Map
FMM
Caustics
Reflections
Refractions
Photon Generation
Kd-Trees
GPU
Texture Cache
memory
Fast Octree Traversal for Ray-Tracing
Ray-Splat Intersections
Normal Field on Splats
Coherent Rays
Super-Sampling

Paths of Photons: Caustics

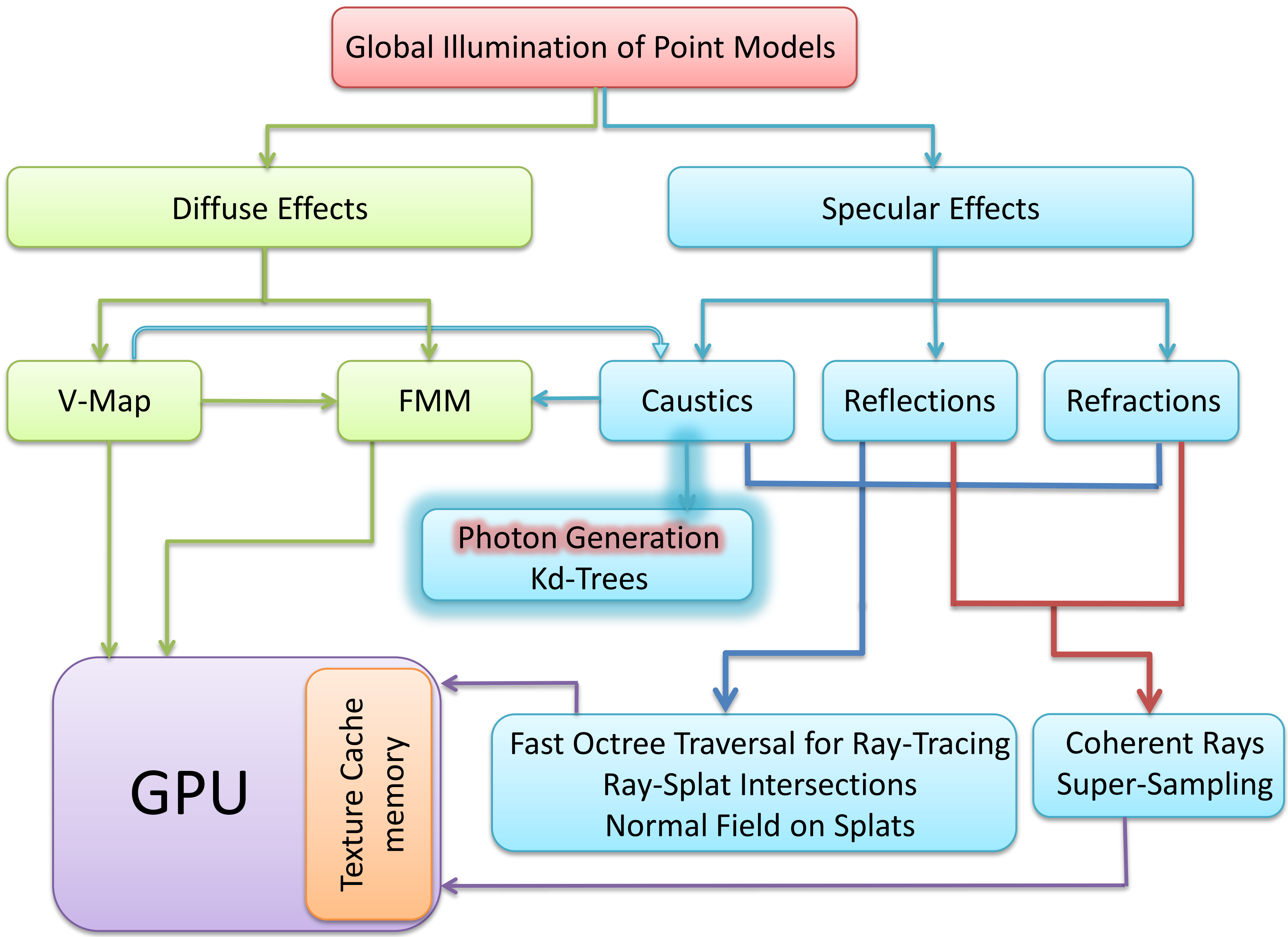


Paths of Photons: Caustic Example



Caustics on Point Models: Phases

- Generate caustic photons at the light source
- Trace photons (LS^+D) in the scene. Deposit them on diffuse surfaces after at least one hit from a specular surface.
  - Traverse photons through octree
  - Use Ray-Splat intersection
- Form a *kd-tree* on the caustic map for fast photon search during ray-tracing
- Render using ray-tracing (View-dependent)


Global Illumination of Point Models
Diffuse Effects
Specular Effects
V-Map
FMM
Caustics
Reflections
Refractions
Photon Generation
Kd-Trees
GPU
Texture Cache
memory
Fast Octree Traversal for Ray-Tracing
Ray-Splat Intersections
Normal Field on Splats
Coherent Rays
Super-Sampling

Caustics: Photon Generation

- The photons emitted have distribution corresponding to the distribution of emissive power of the light source

Rhushabh Goradia, Sharat Chandran

Caustics: Photon Generation

- The photons emitted have distribution corresponding to the distribution of emissive power of the light source

$$P_{photon} = P_{light} / n_e$$

P_{photon} = Power per photon
P_{light} = Emissive power of light
ne = Number of Photons

Caustics: Photon Generation

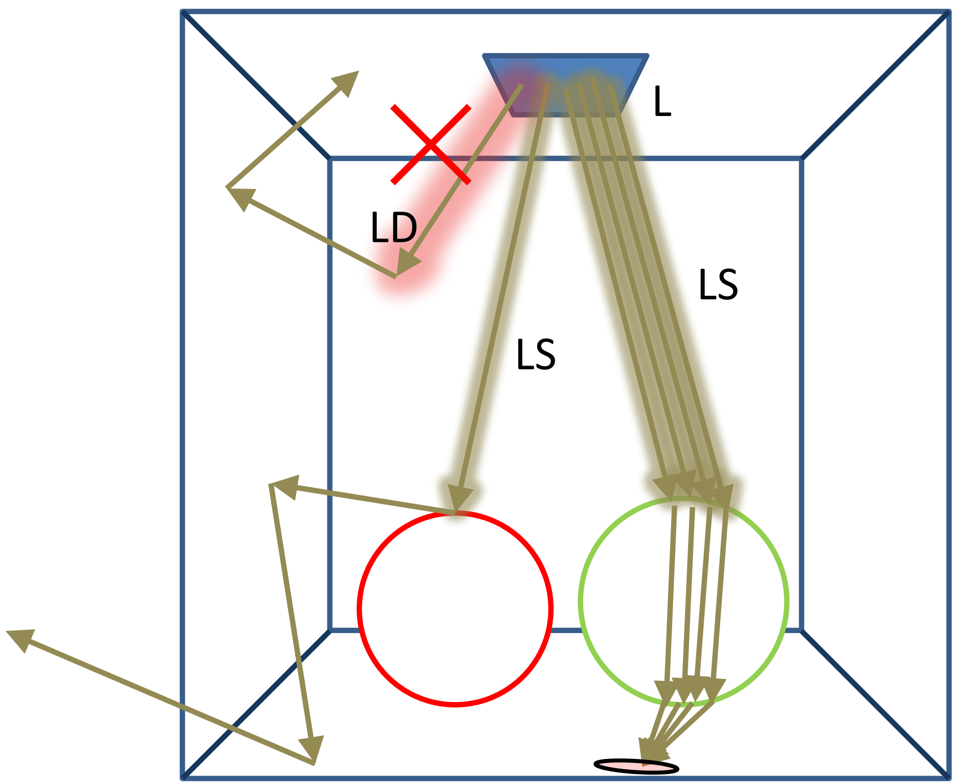
- The photons emitted have distribution corresponding to the distribution of emissive power of the light source

$$P_{photon} = P_{light} / n_e$$

P_{photon} = Power per photon
P_{light} = Emissive power of light
ne = Number of Photons

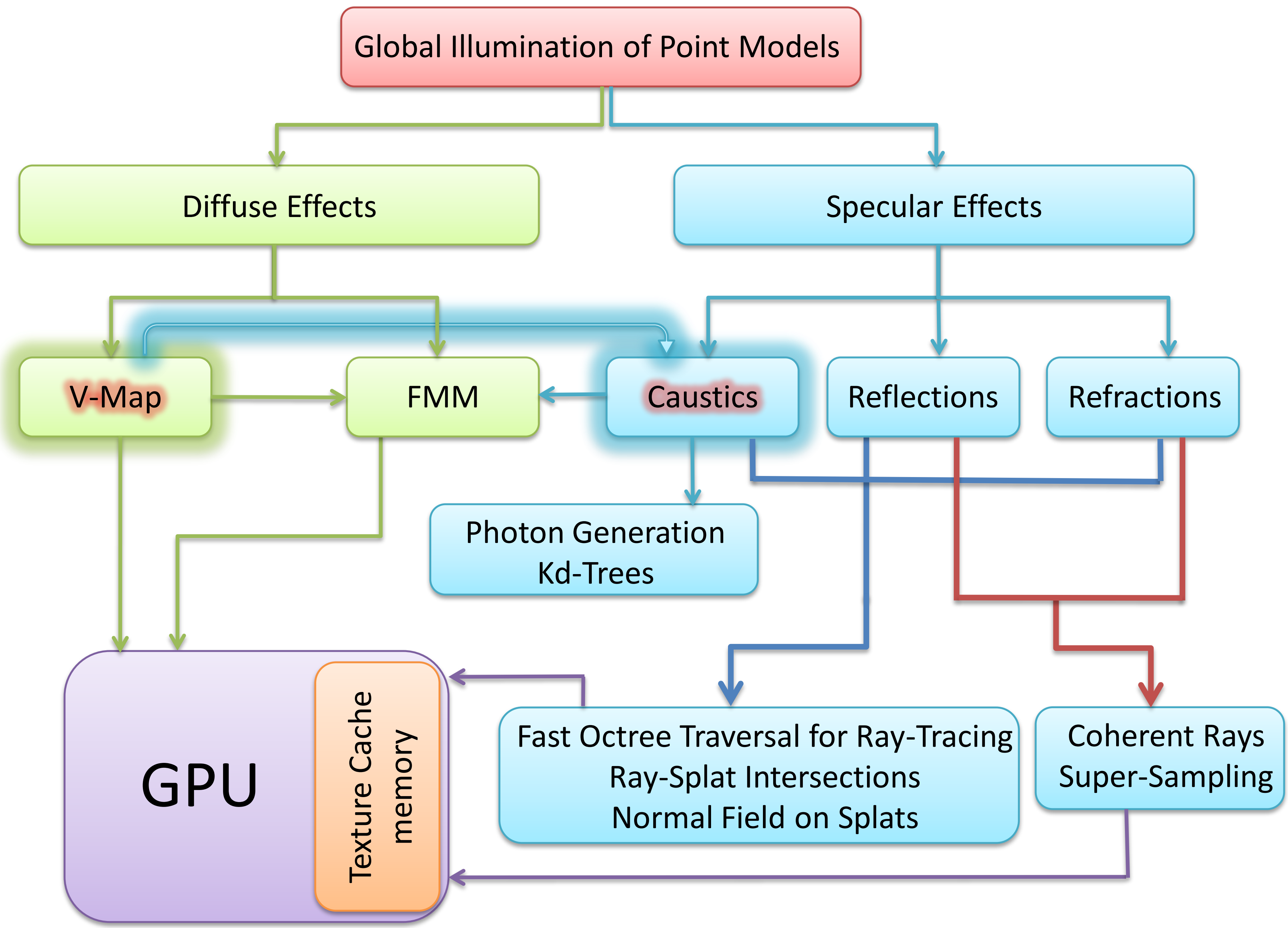
- The number of photons (n_e) is pre-defined (e.g. 4,00,000)
- The emissive power of light source (P_{light}) is also taken as input

Caustics: Usage of V-Map

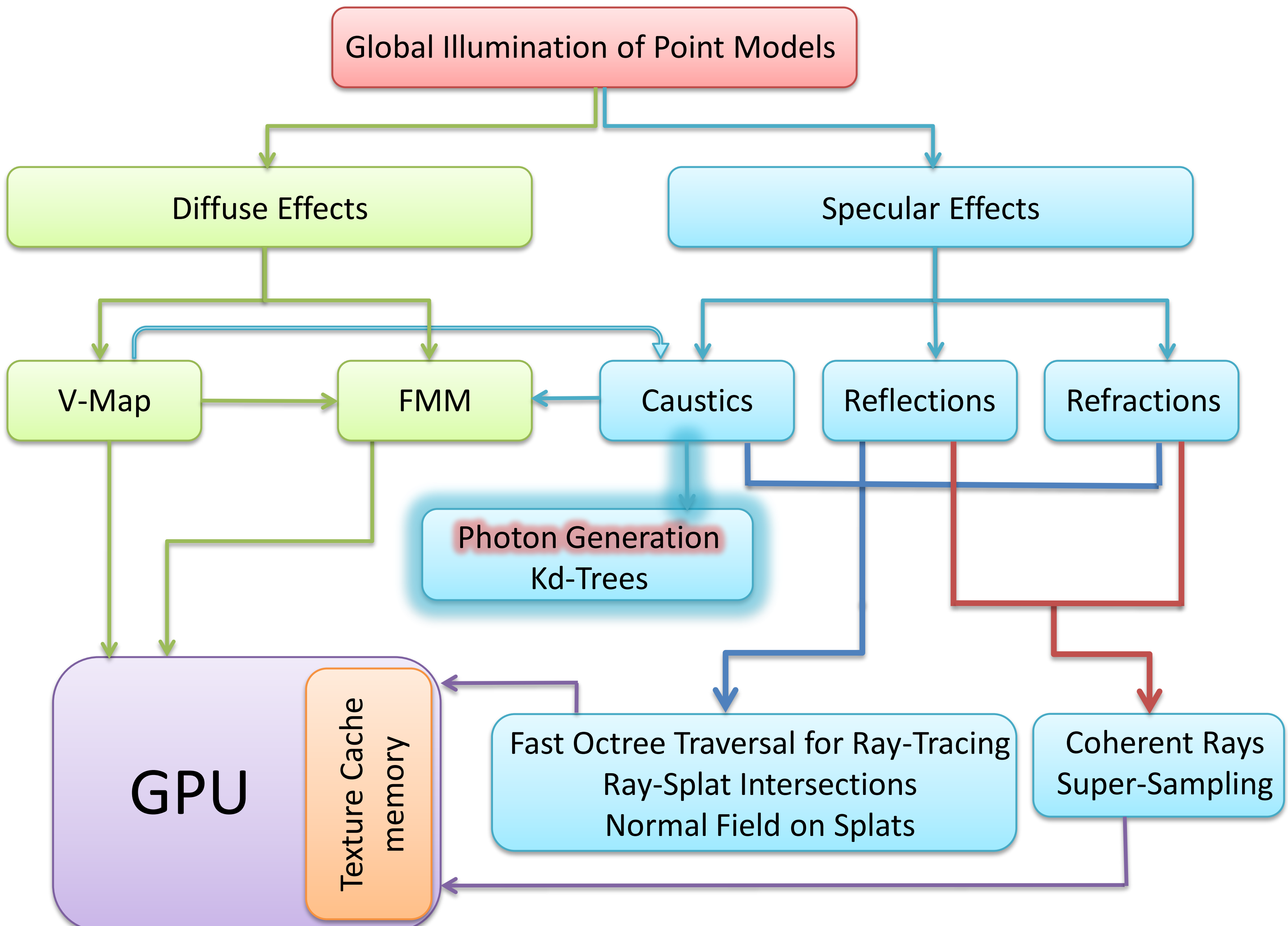


- Caustics paths: Only LS^+D
- Send rays only to visible specular leaf nodes (leaves which contain specular points)
- Use visible links of light source (*V-Map*)
- Saves time as we do not search for any caustic generators

- Remember – Scene is divided into an octree !



Global Illumination of Point Models
Diffuse Effects
Specular Effects
V-Map
FMM
Caustics
Reflections
Refractions
Photon Generation
Kd-Trees
GPU
Texture Cache
memory
Fast Octree Traversal for Ray-Tracing
Ray-Splat Intersections
Normal Field on Splats
Coherent Rays
Super-Sampling

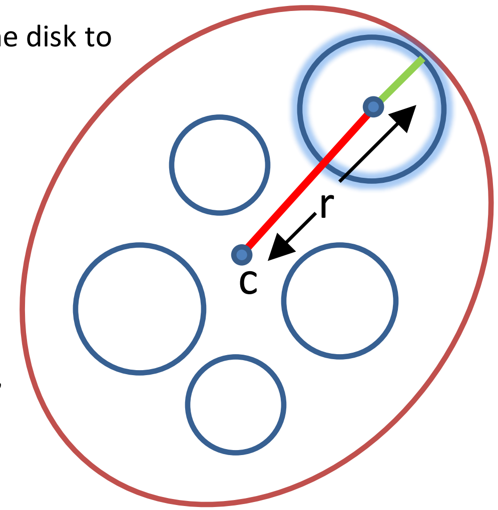


Global Illumination of Point Models
Diffuse Effects
Specular Effects
V-Map
FMM
Caustics
Reflections
Refractions
Photon Generation
Kd-Trees
GPU
Texture Cache
memory
Fast Octree Traversal for Ray-Tracing
Ray-Splat Intersections
Normal Field on Splats
Coherent Rays
Super-Sampling

Caustics: Number of Photons To Shoot Per Leaf

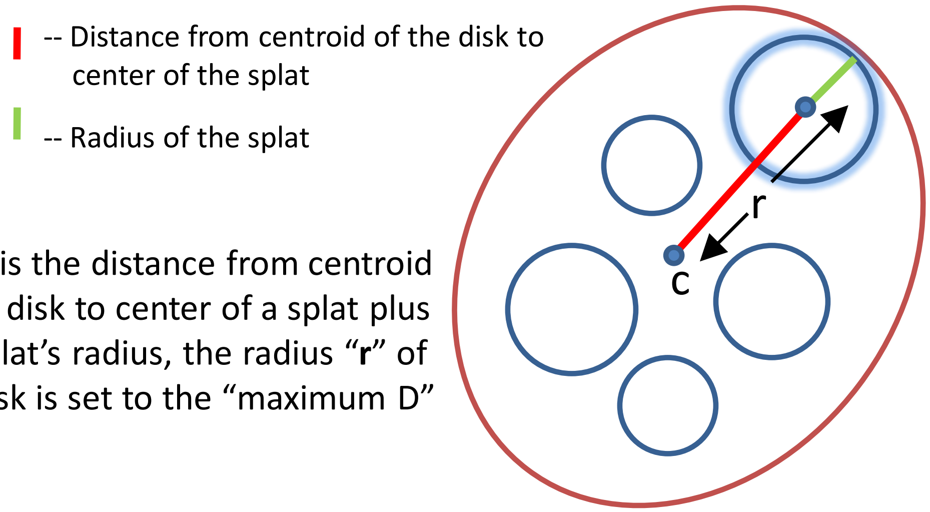
-- Distance from centroid of the disk to center of the splat

-- Radius of the splat



If "D" is the distance from centroid of the disk to center of a splat plus the splat's radius, the radius "**r**" of the disk is set to the "maximum D"

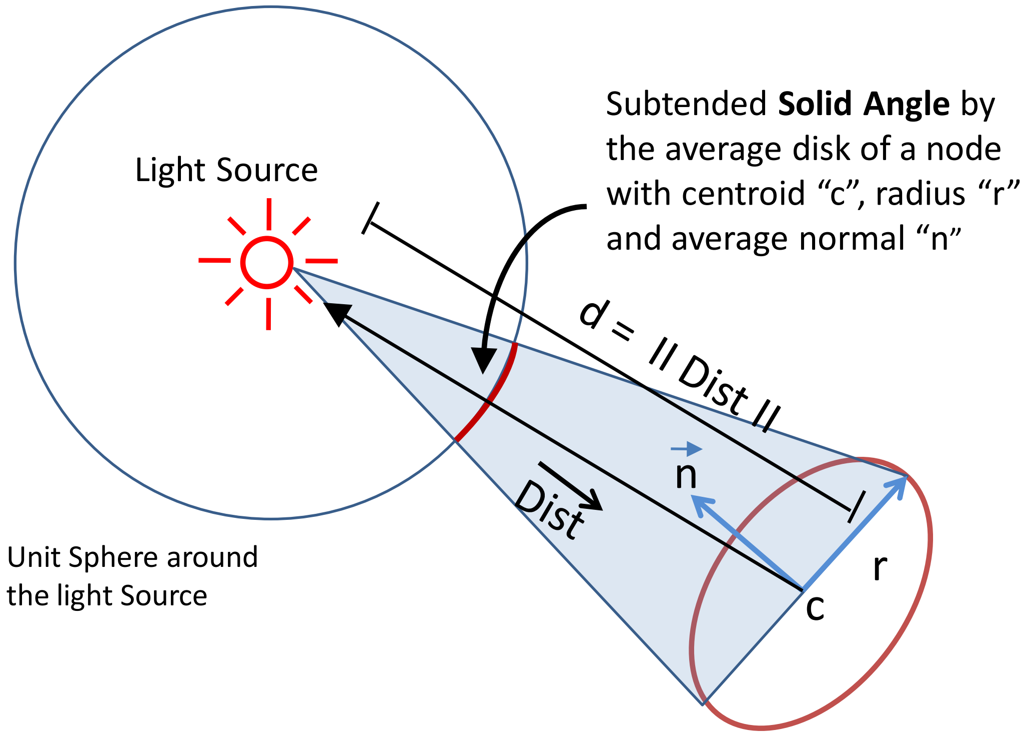
Caustics: Number of Photons To Shoot Per Leaf



If "D" is the distance from centroid of the disk to center of a splat plus the splat's radius, the radius "**r**" of the disk is set to the "maximum D"

- We find the ***solid angle*** subtended by this average disk *D* of *leaf A* from the light source

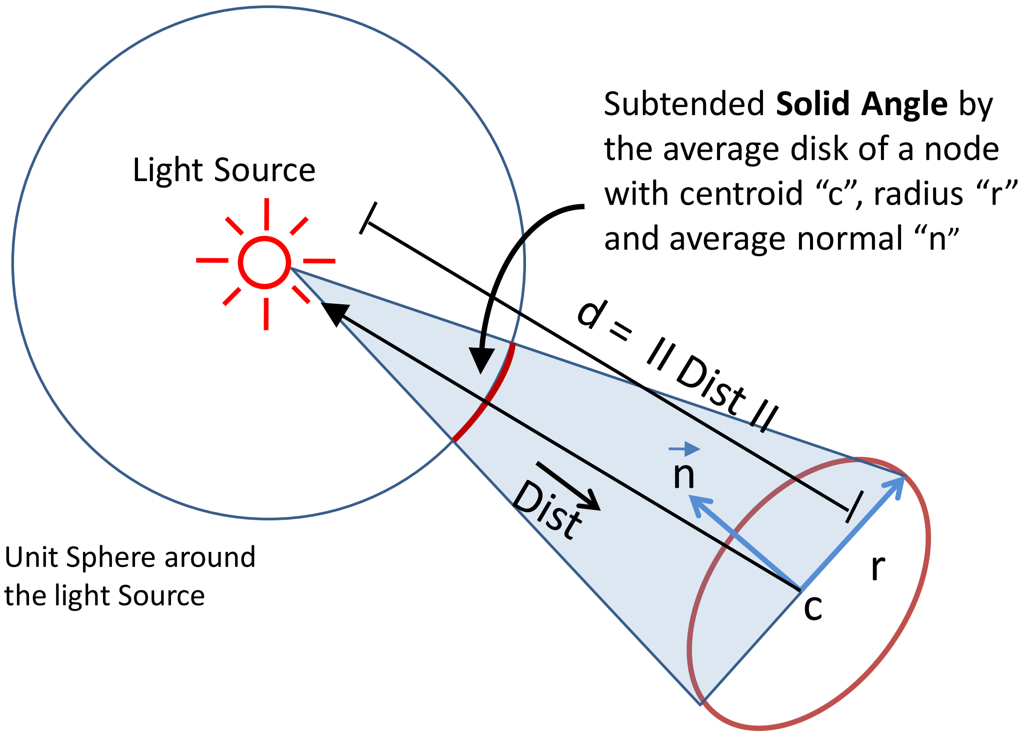
Caustics: Number of Photons To Shoot Per Leaf



- We find the ***solid angle*** subtended by this average disk *D* of *leaf A* from the light source

$$S.A. = (Area * \cos\theta) \,/\, d^2$$

Caustics: Number of Photons To Shoot Per Leaf



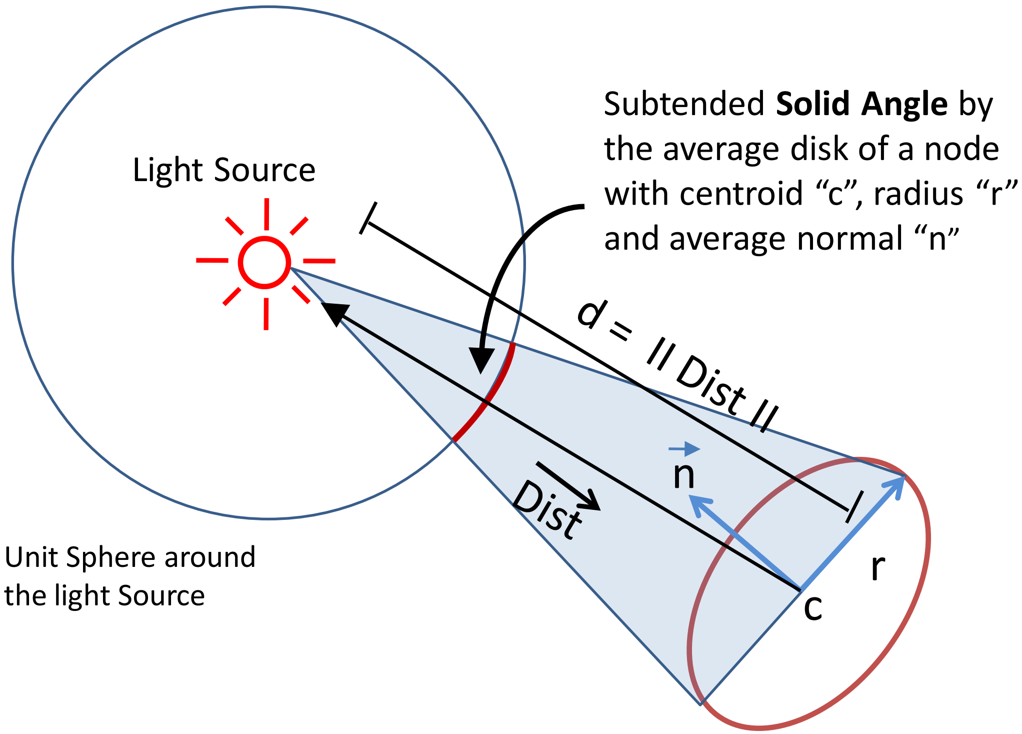
- We find the ***solid angle*** subtended by this average disk *D* of *leaf A* from the light source

$$S.A. = (Area * \cos\theta) \,/\, d^2$$

$$S.A.(A) = (\pi r^2 * DP) \,/\, d^2$$

where $DP = dot(Dist, n)$

Caustics: Number of Photons To Shoot Per Leaf



- We find the ***solid angle*** subtended by this average disk *D* of *leaf A* from the light source

$$S.A.(A) = (\pi r^2 * DP) \,/\, d^2$$

$$N_{photons}(A) = n_e * S.A.(A) \,/\, 4\pi$$

Caustics: Number of Photons Per Splat

$$P.P.S(A) = N_{photons}(A) \;/\; N.P(A)$$

where,

$P.P.S(A) =$ *Photons per splat in A*

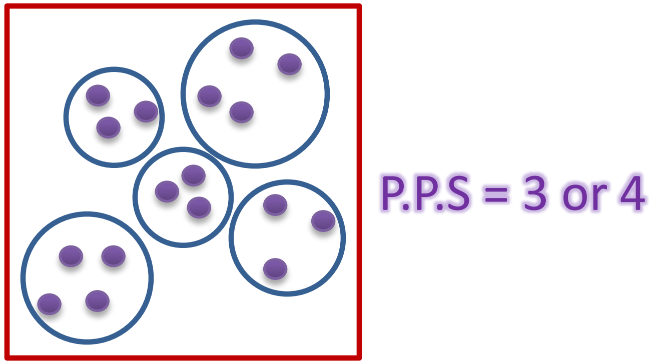
$N_{photons}(A) = ne * S.A.(A) \;/\; 4\pi$

$N.P.(A) =$ *Number of points in A*

- The remaining $N_{photons}(A)\%N.P.(A)$ number of photons are distributed randomly to splats in A

Caustics: Number of Photons Per Splat

$$P.P.S(A) = N_{photons}(A) \,/\, N.P(A)$$



Leaf A

where,

$P.P.S(A) =$ *Photons per splat in A*

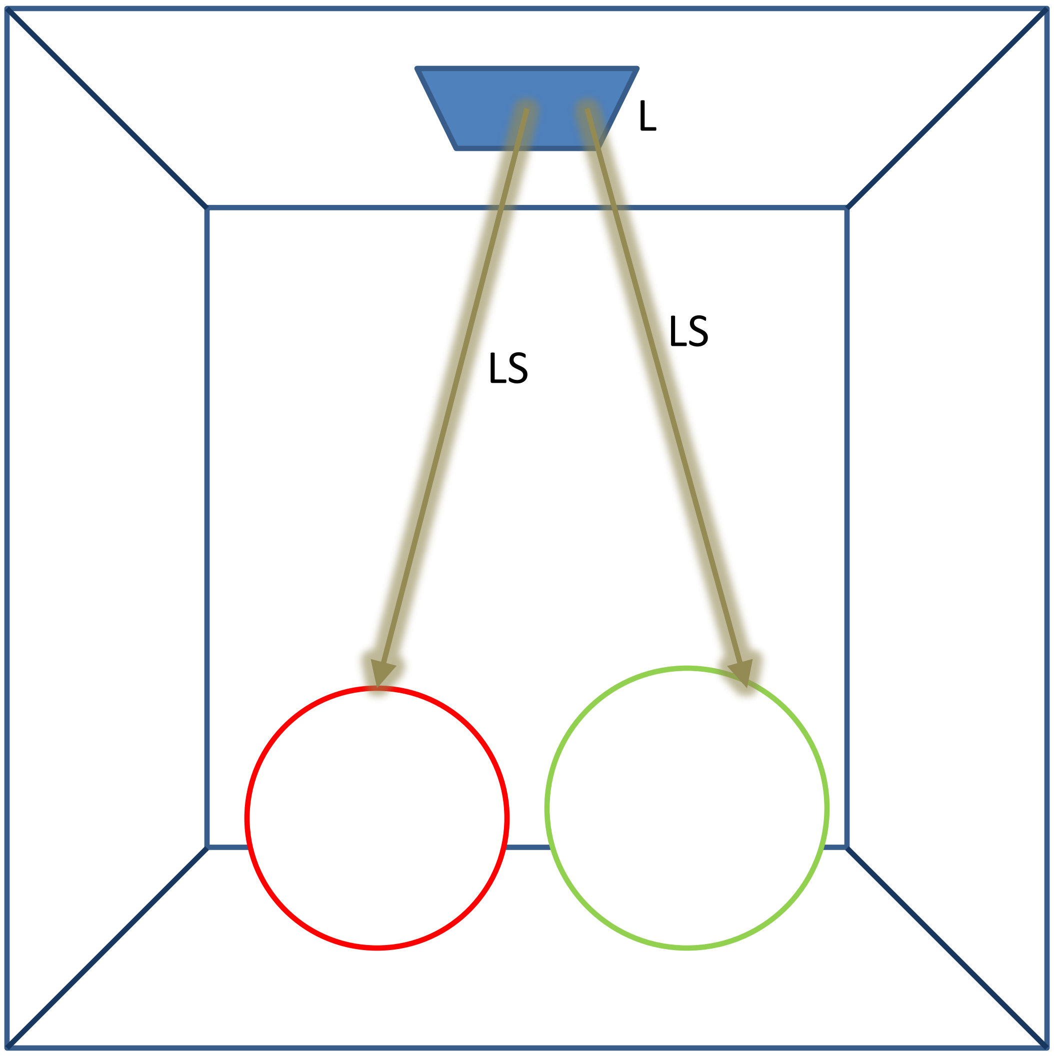
$N_{photons}(A) = ne * S.A.(A) \,/\, 4\pi$

$N.P.(A) =$ *Number of points in A*

- The remaining $N_{photons}(A)\%N.P.(A)$ number of photons are distributed randomly to splats in A
- Use random sampling on each splat of *A* to get photon hit locations, the number of samples on each splat equal to *P.P.S* of that splat

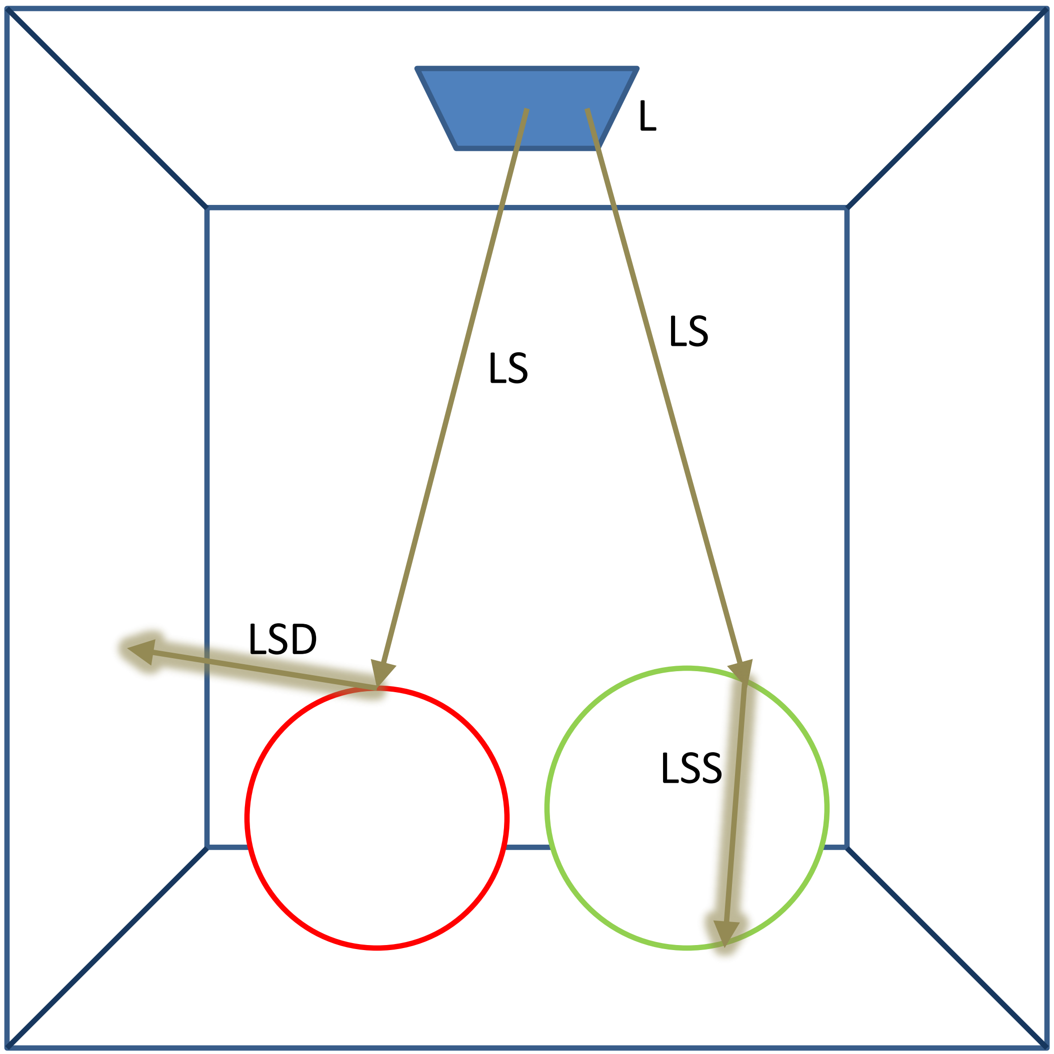
Ray-Splat Intersections: Normal Field

- Photons sent to specular leaves from light source



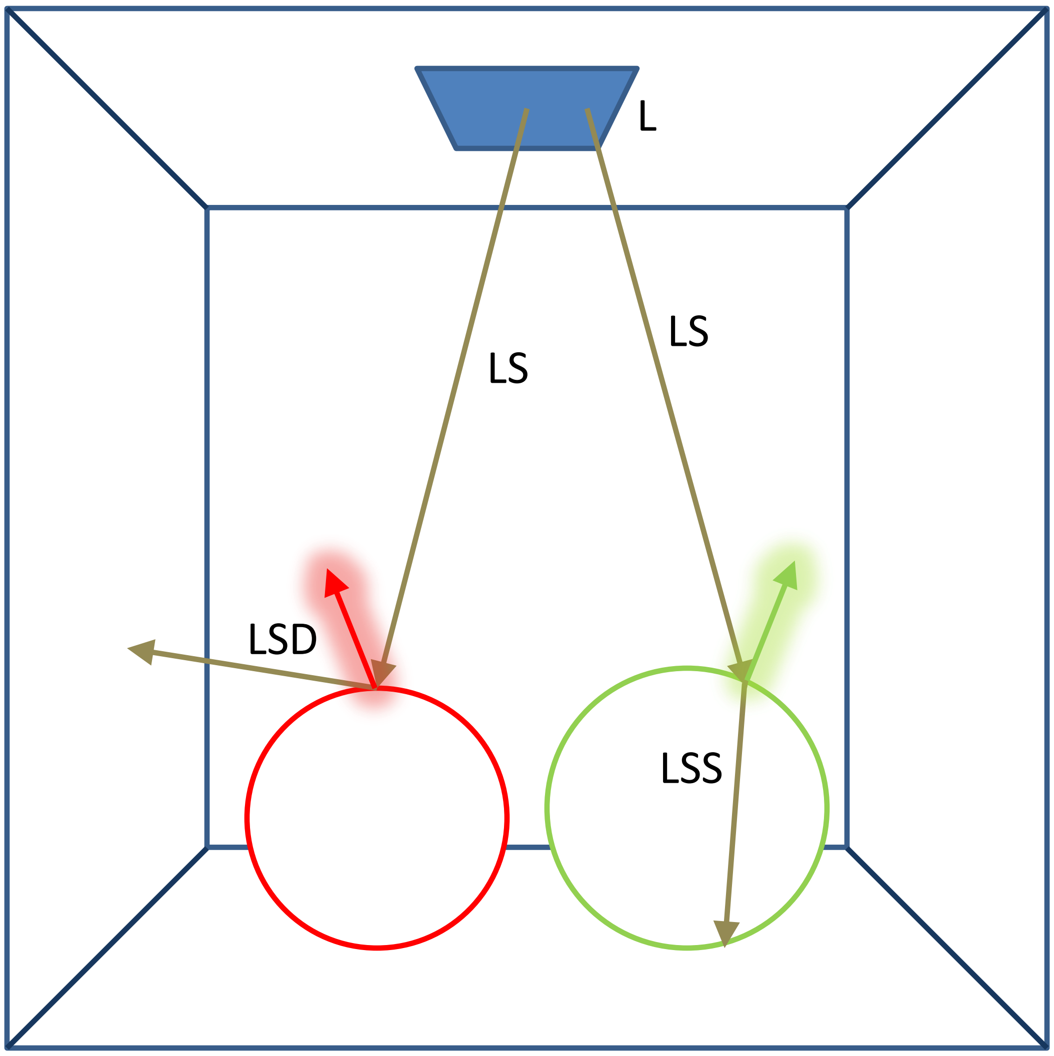
Ray-Splat Intersections: Normal Field

- Generate new reflective or refractive rays



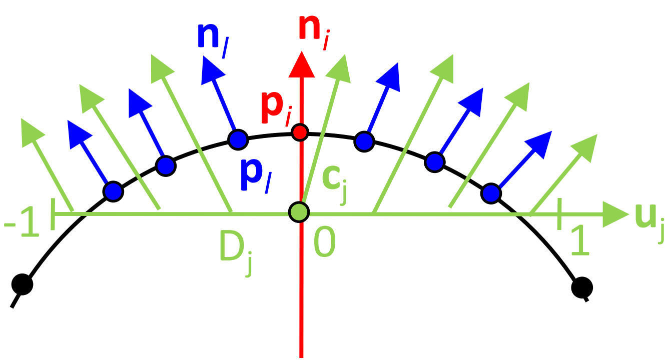
Ray-Splat Intersections: Normal Field

- We need normal at the intersection point on the splat



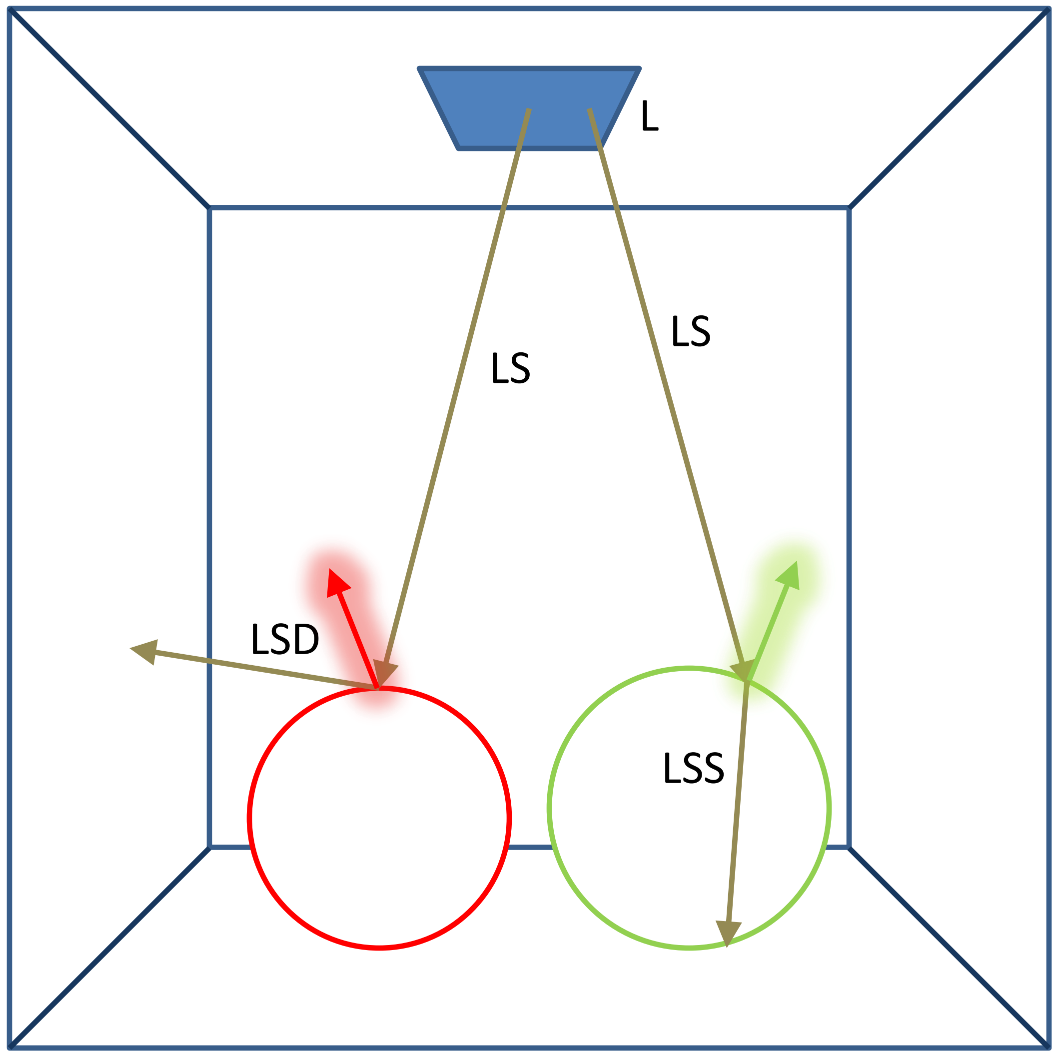
Ray-Splat Intersections: Normal Field

- We need normal at the intersection point on the splat



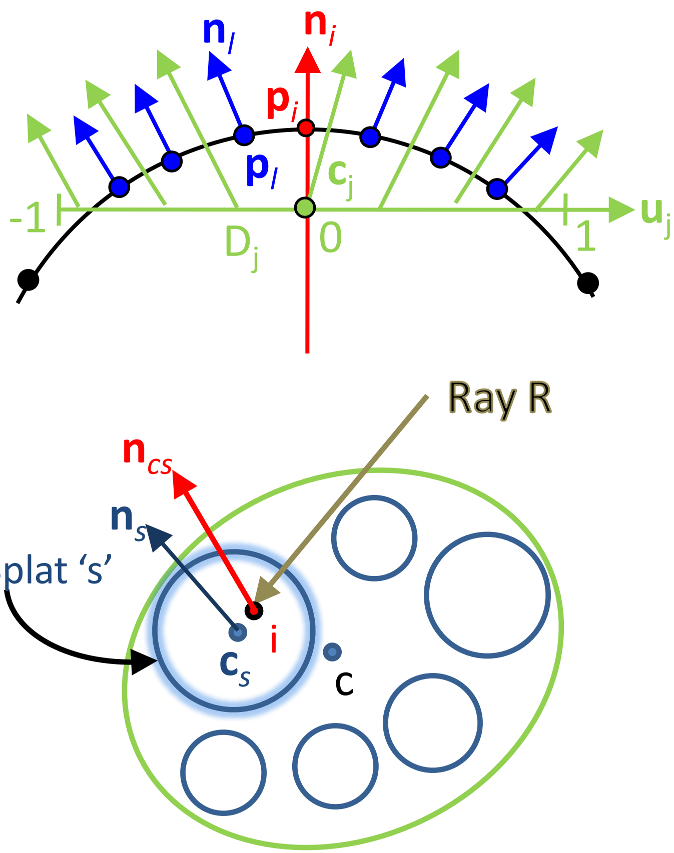
- [LMR 07], Splat Based Ray-Tracing of Point Clouds, Journal of WSCG, 2007

- Pre-process and Store

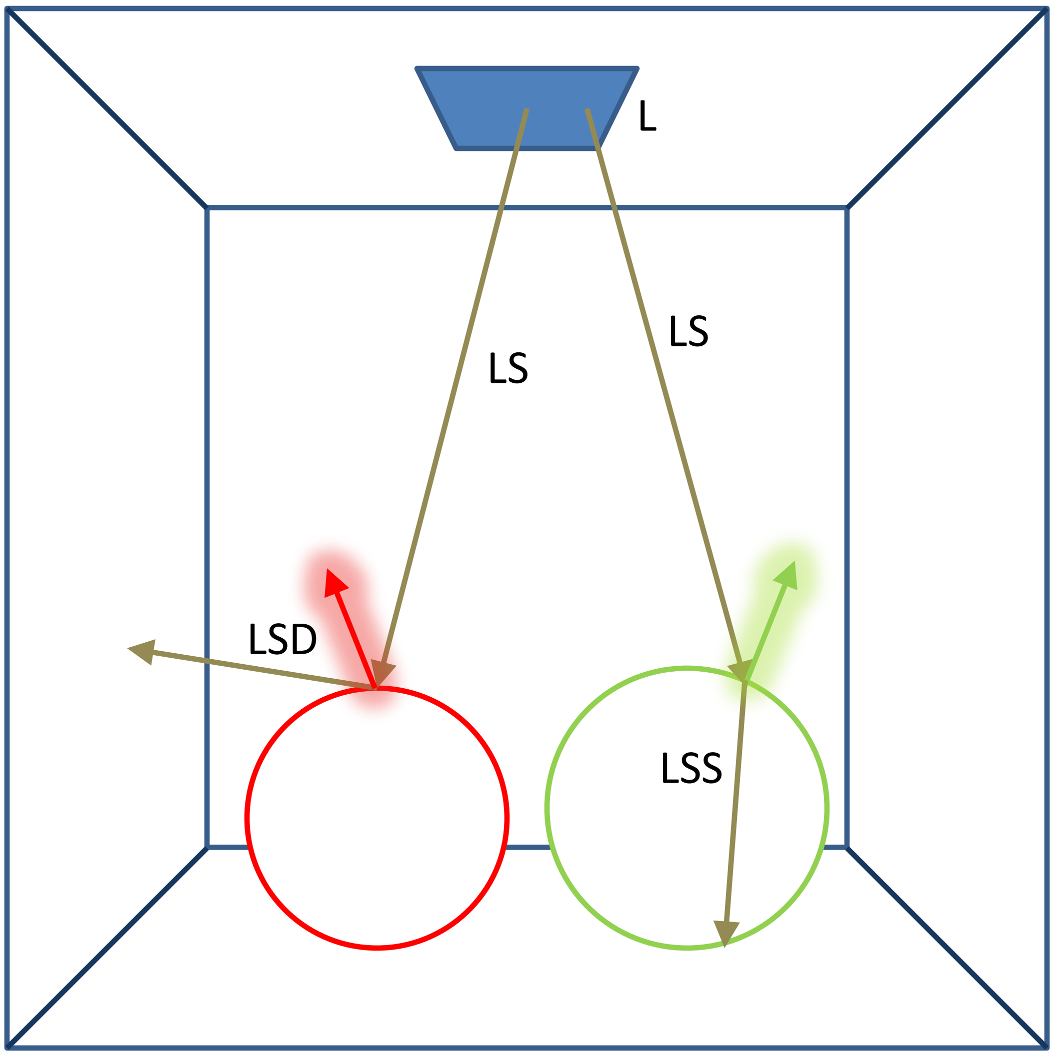


Ray-Splat Intersections: Normal Field

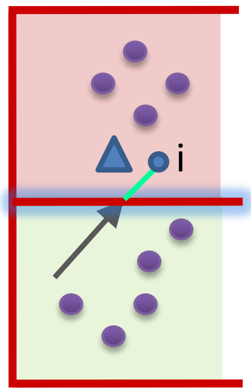
- We need normal at the intersection point on the splat

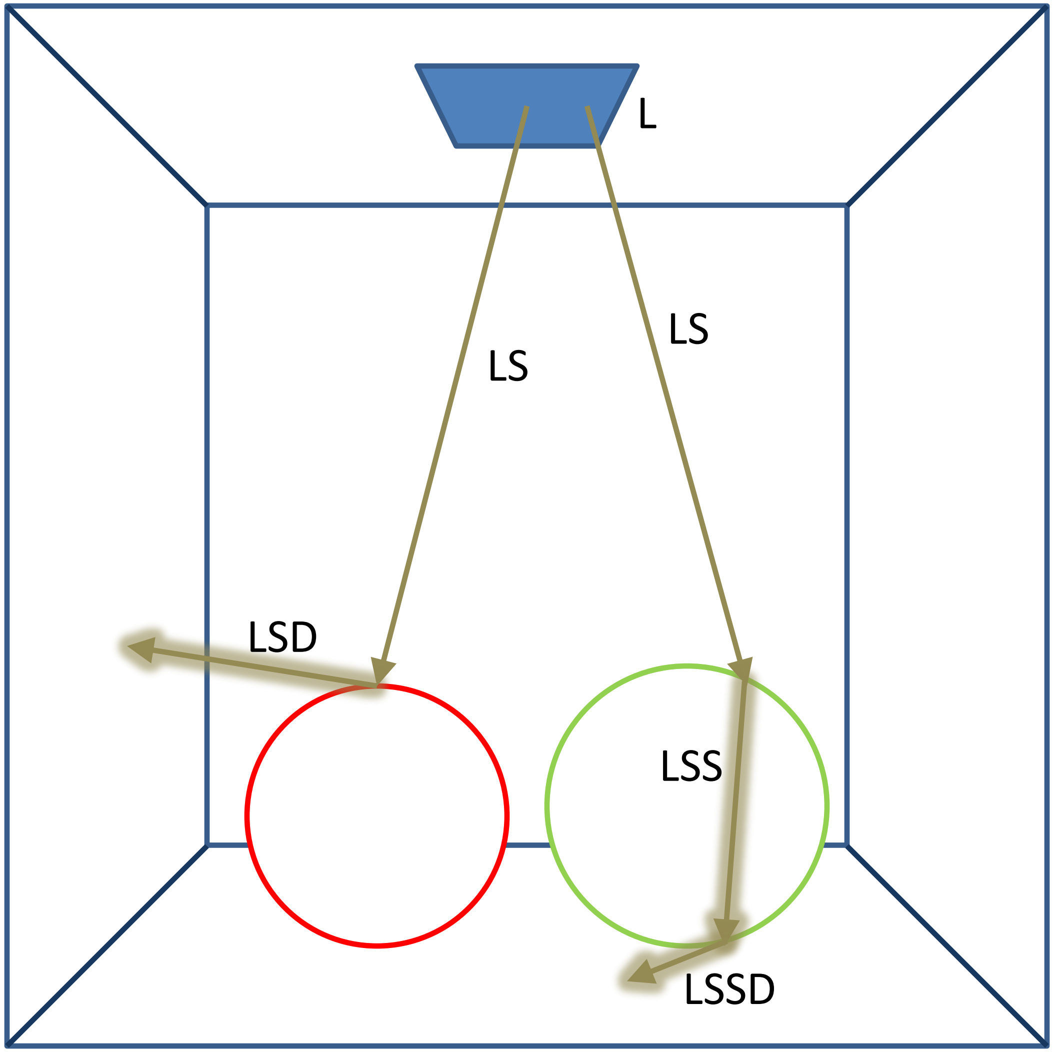


Average disk of a node with center "c"



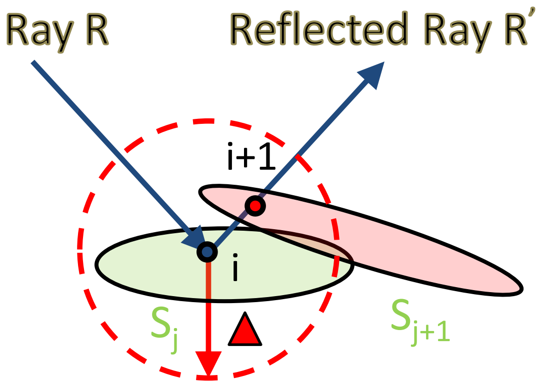
Tracing Caustic Photons

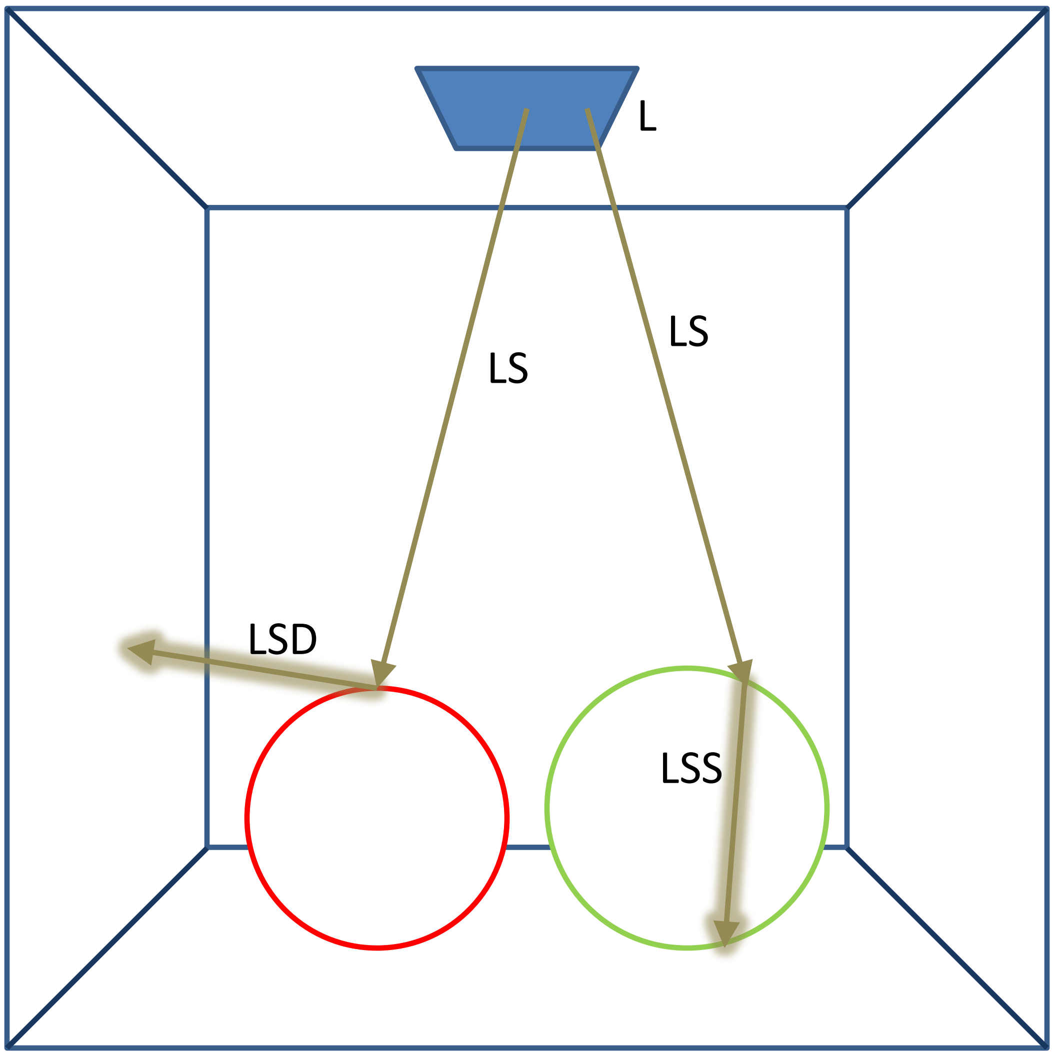




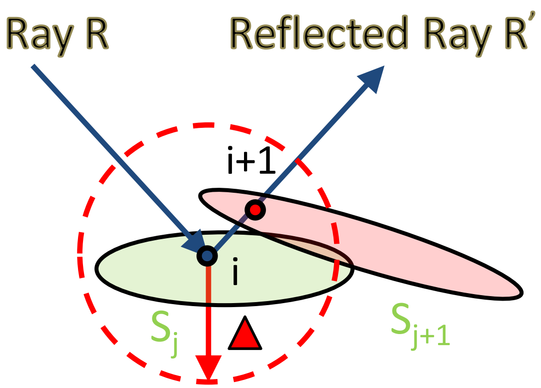
Rhushabh Goradia, Sharat Chandran

Ray-Splat Intersections: Minimum Delta (▲)

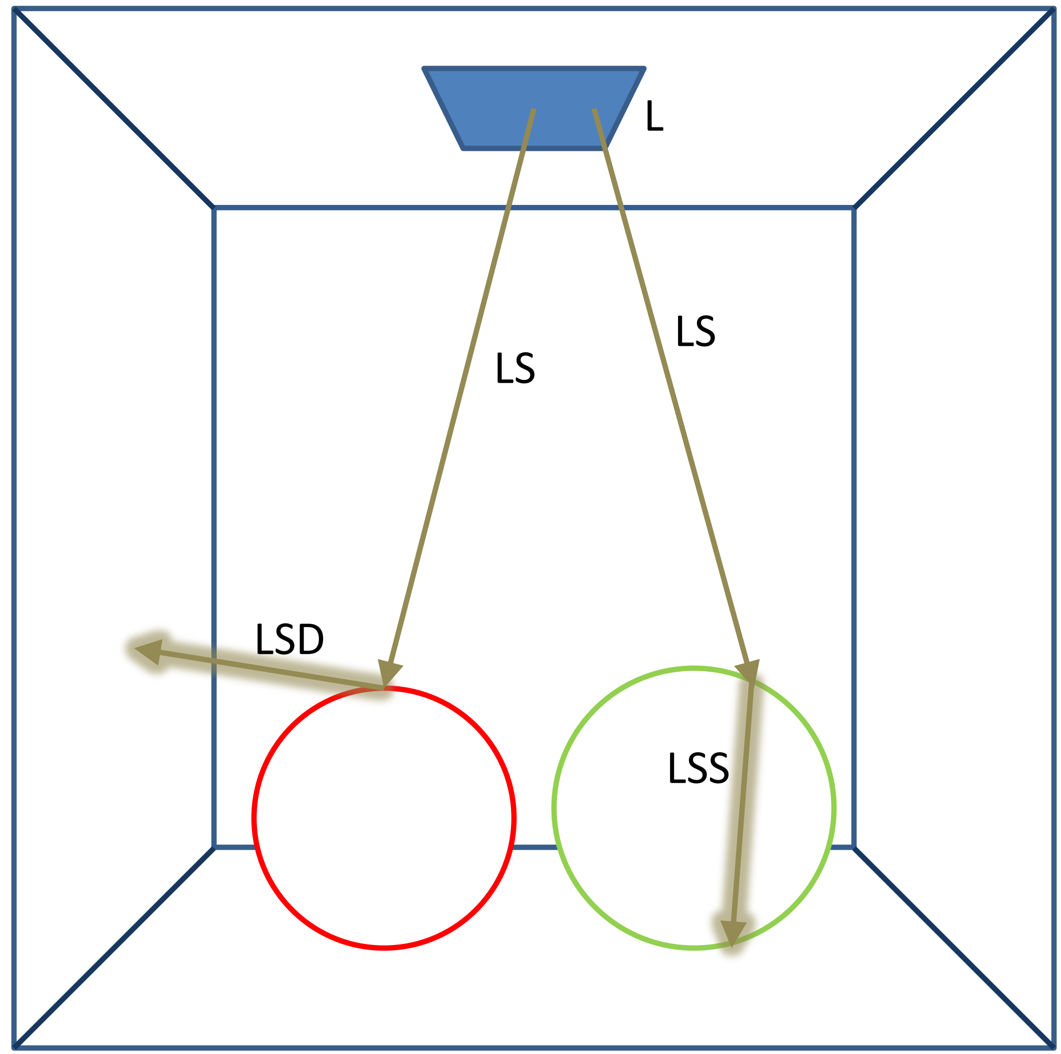




Ray-Splat Intersections: Minimum Delta (▲)

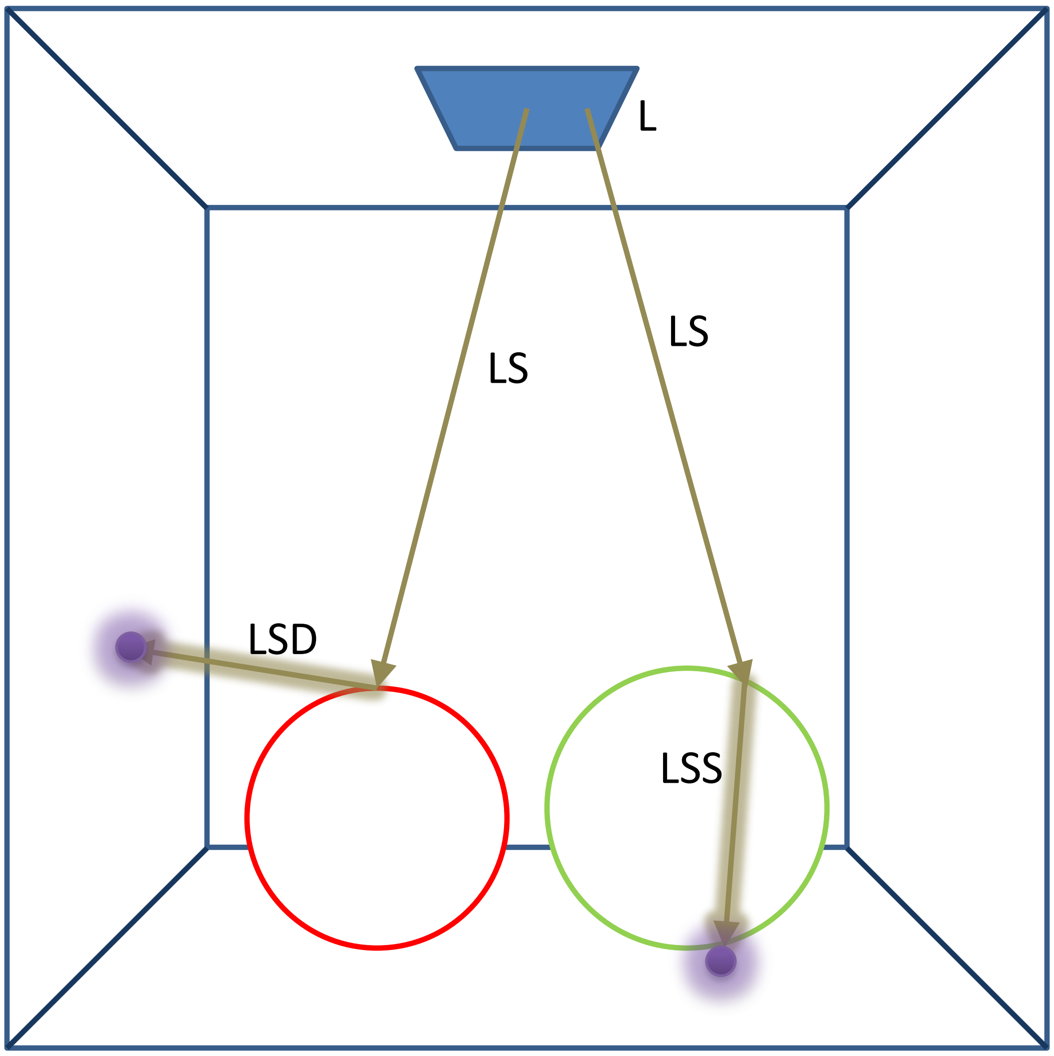


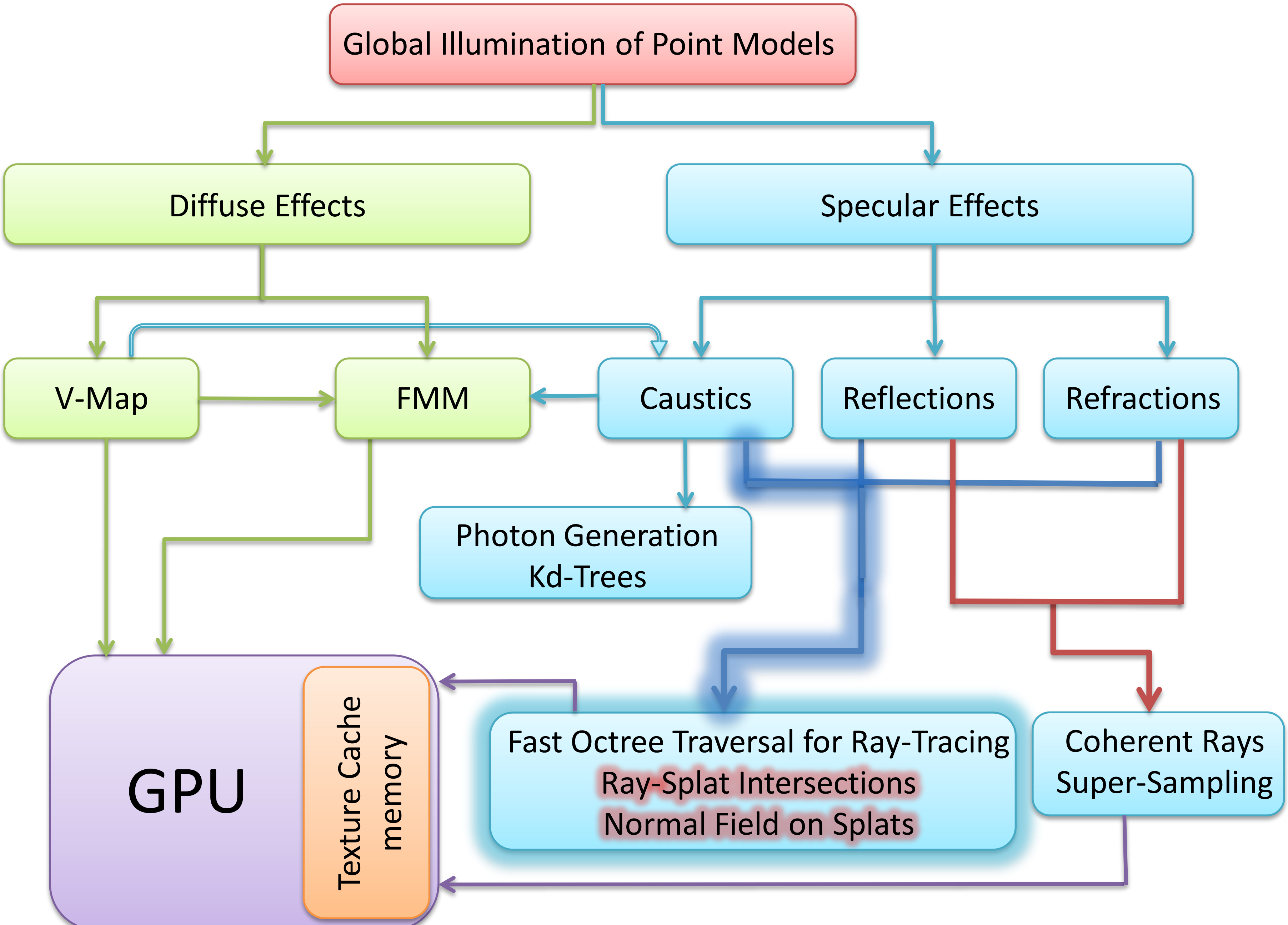
- We set ▲ to 0.05



Ray-Splat Intersections

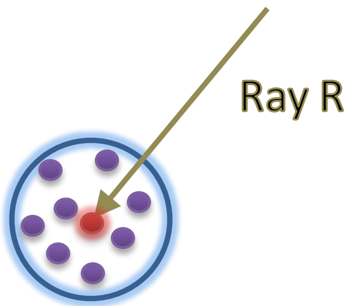
- We perform a standard Ray-Disk Intersection to find the intersecting splat

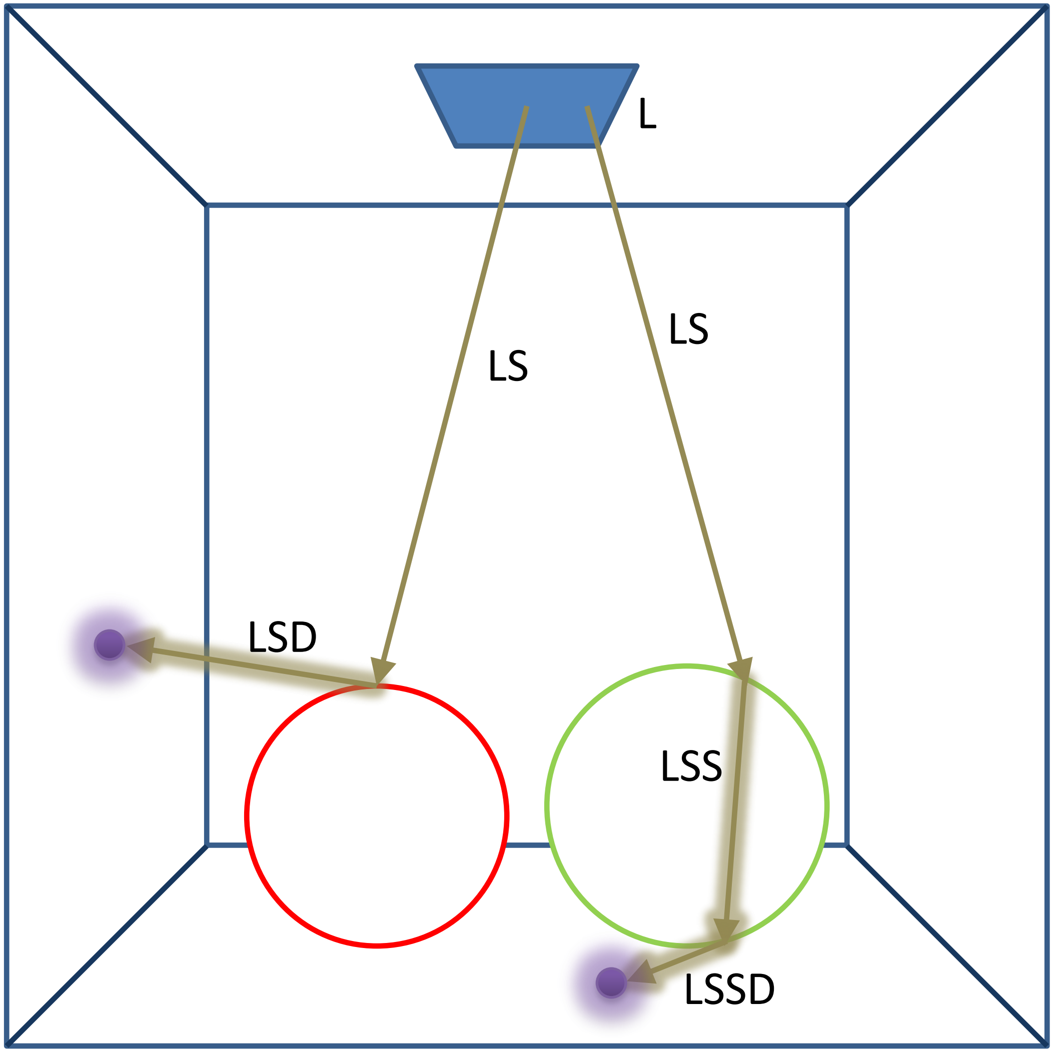



Global Illumination of Point Models
Diffuse Effects
Specular Effects
V-Map
FMM
Caustics
Reflections
Refractions
Photon Generation
Kd-Trees
GPU
Texture Cache
memory
Fast Octree Traversal for Ray-Tracing
Ray-Splat Intersections
Normal Field on Splats
Coherent Rays
Super-Sampling

Storing Caustic Photons: KD-Tree

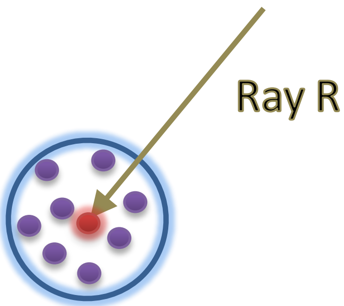
- Store the view-independent caustic map as a *kd-tree*



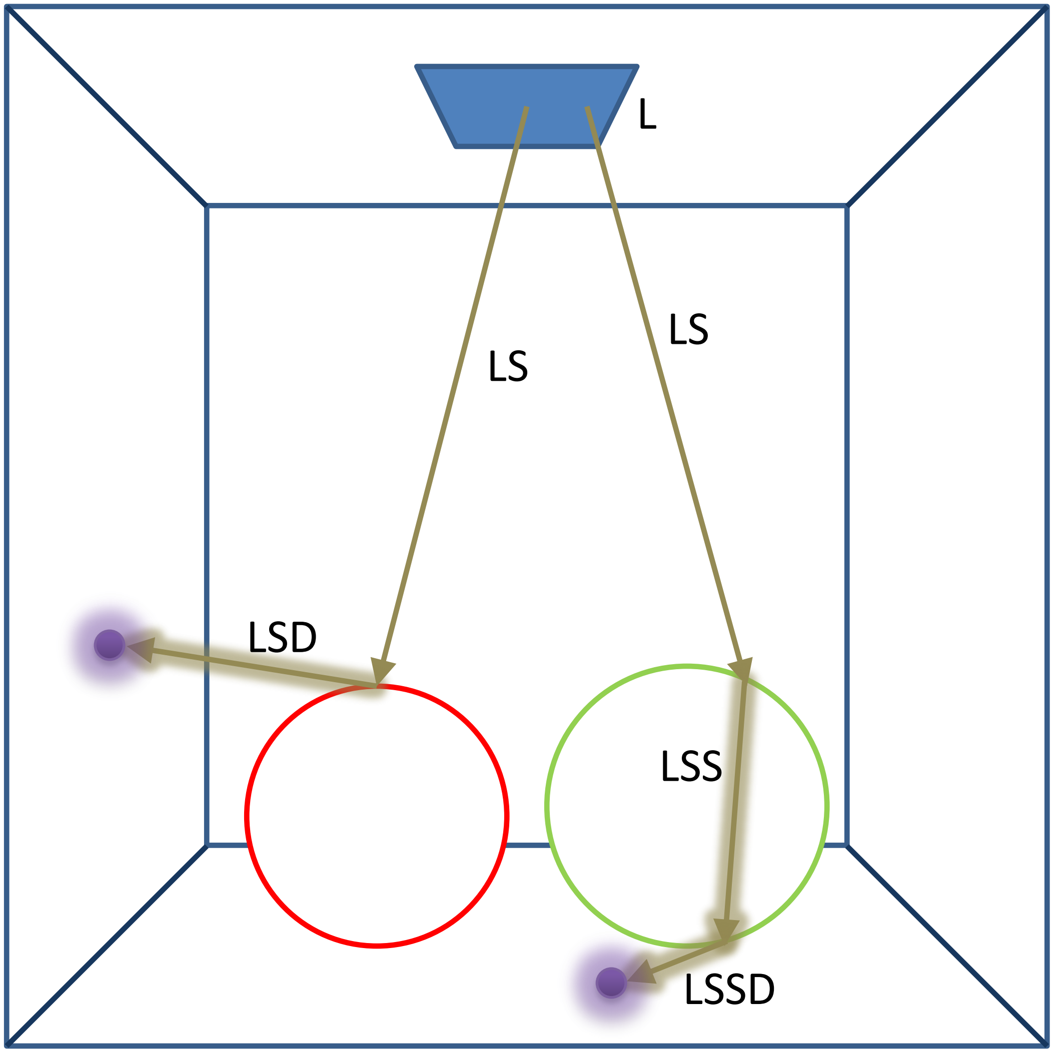


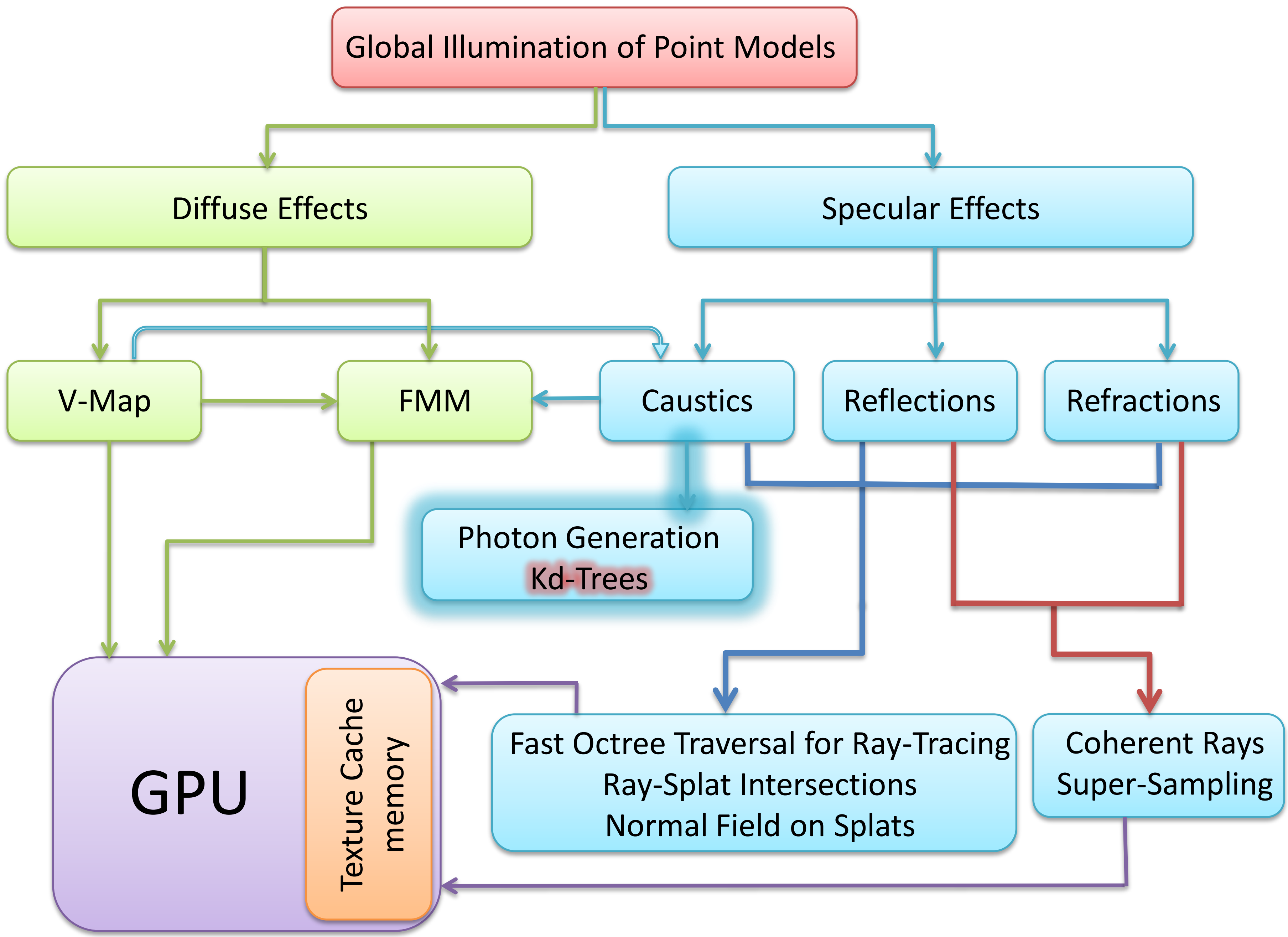
Storing Caustic Photons: KD-Tree

- Store the view-independent caustic map as a *kd-tree*

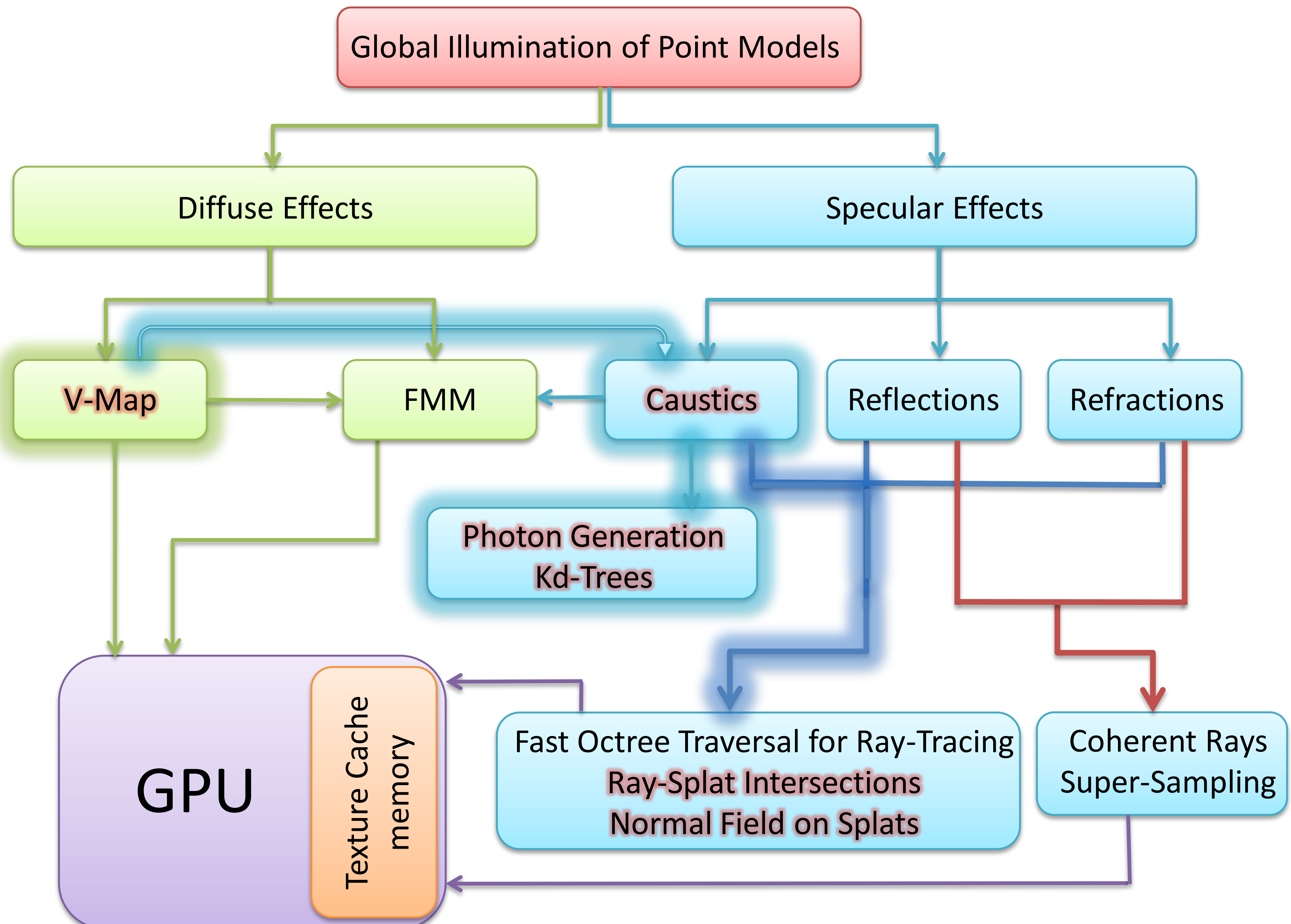


- Searching for nearest photons in the *kd-tree* is a run time step, while storing them is pre-computation
- We use the Approximate Near Neighbor Search (ANN) library for searching and constructing the *kd-tree*



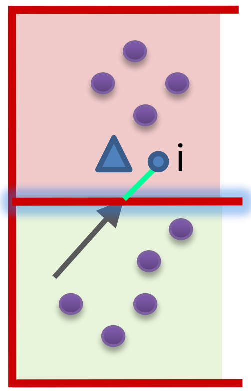


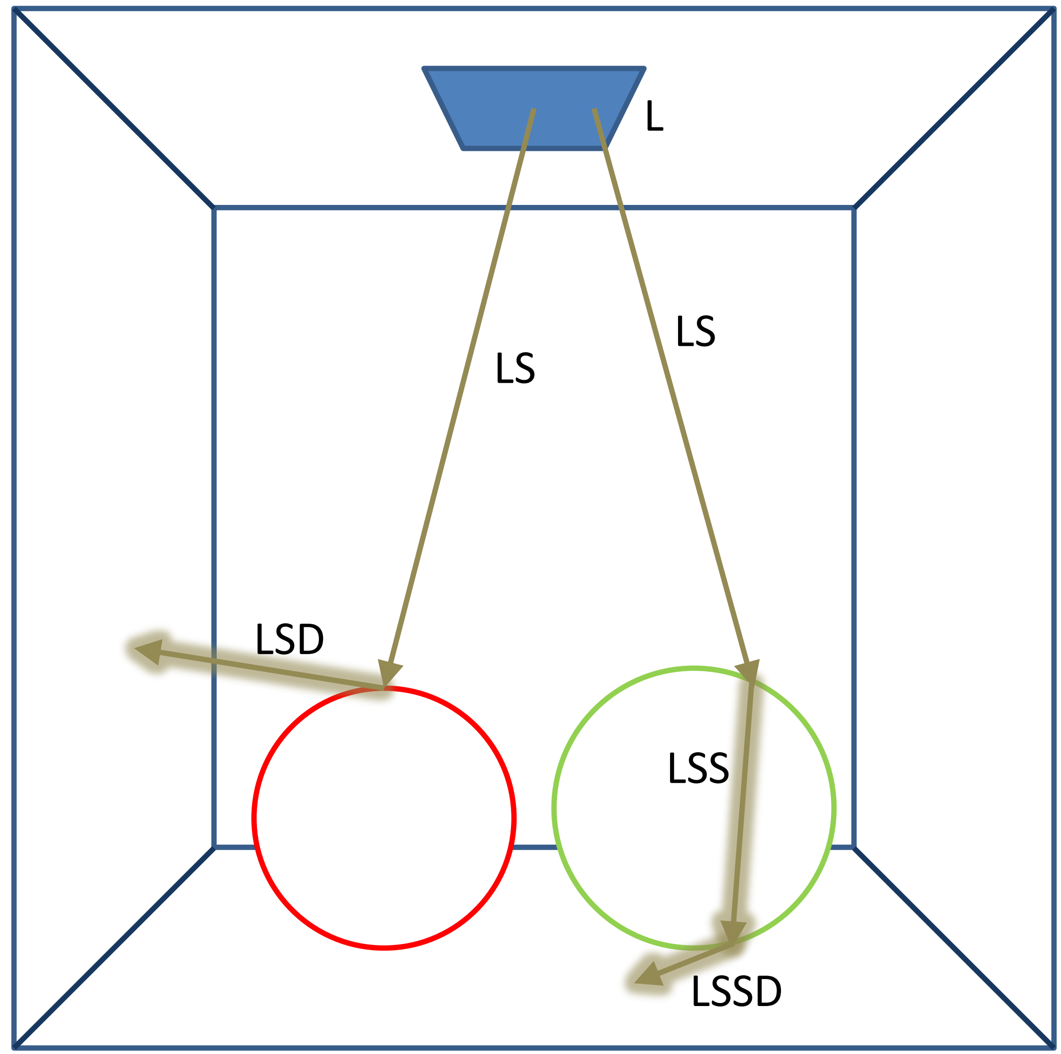
Global Illumination of Point Models
Diffuse Effects
Specular Effects
V-Map
FMM
Caustics
Reflections
Refractions
Photon Generation
Kd-Trees
GPU
Texture Cache memory
Fast Octree Traversal for Ray-Tracing
Ray-Splat Intersections
Normal Field on Splats
Coherent Rays
Super-Sampling



Global Illumination of Point Models
Diffuse Effects
Specular Effects
V-Map
FMM
Caustics
Reflections
Refractions
Photon Generation
Kd-Trees
GPU
Texture Cache
memory
Fast Octree Traversal for Ray-Tracing
Ray-Splat Intersections
Normal Field on Splats
Coherent Rays
Super-Sampling

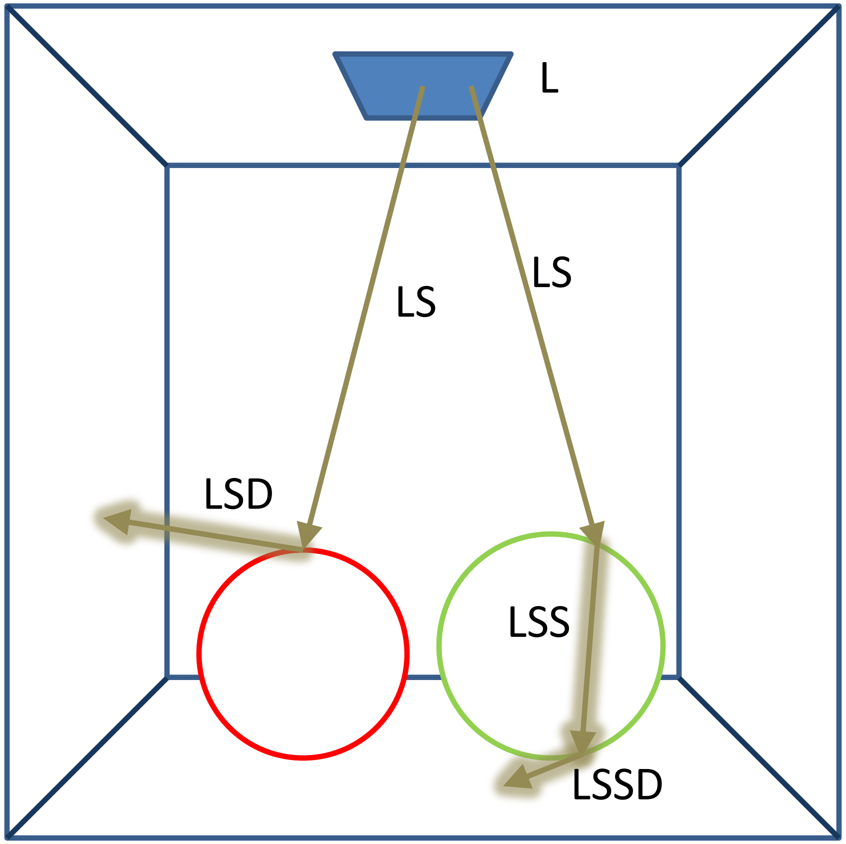
Tracing Caustic Photons

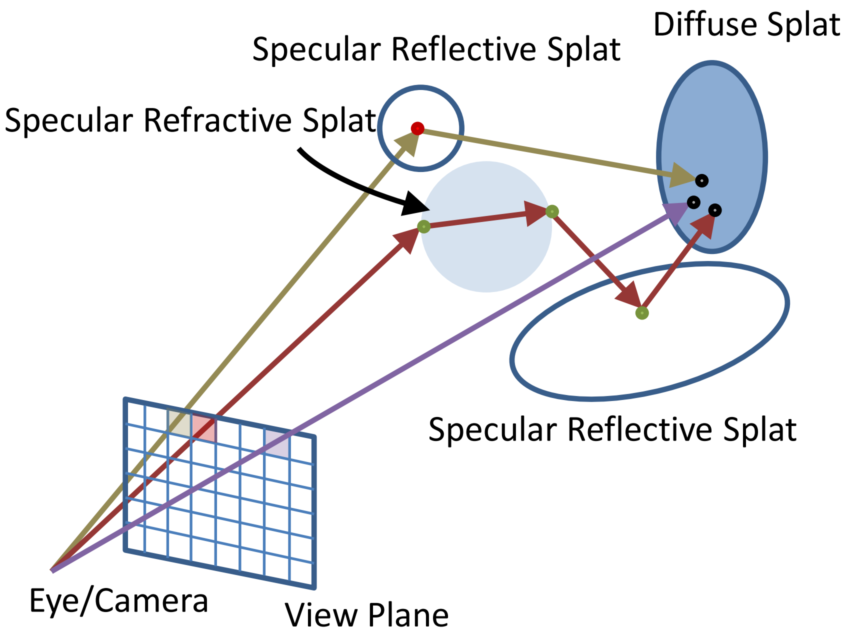


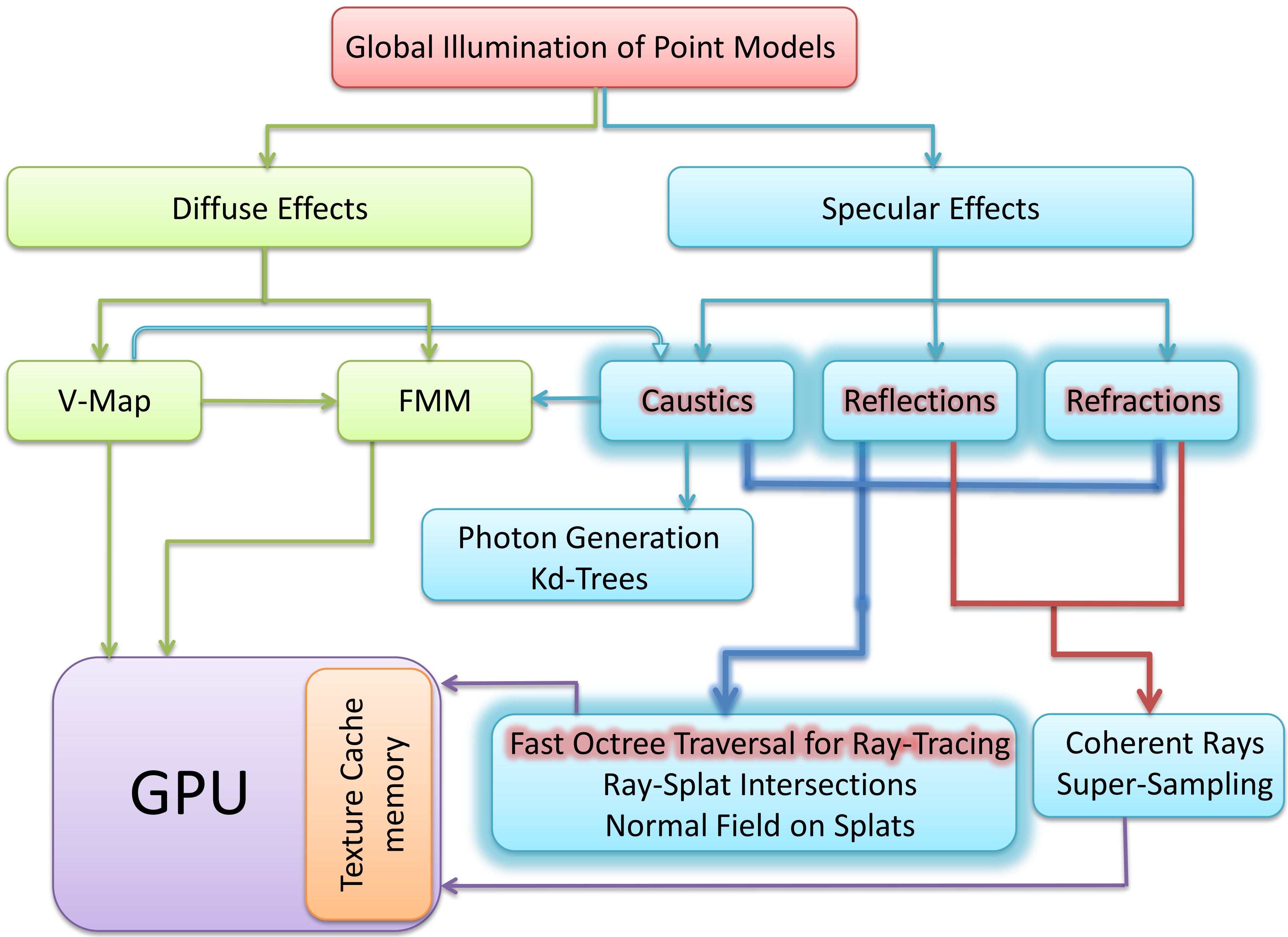


Rhushabh Goradia, Sharat Chandran

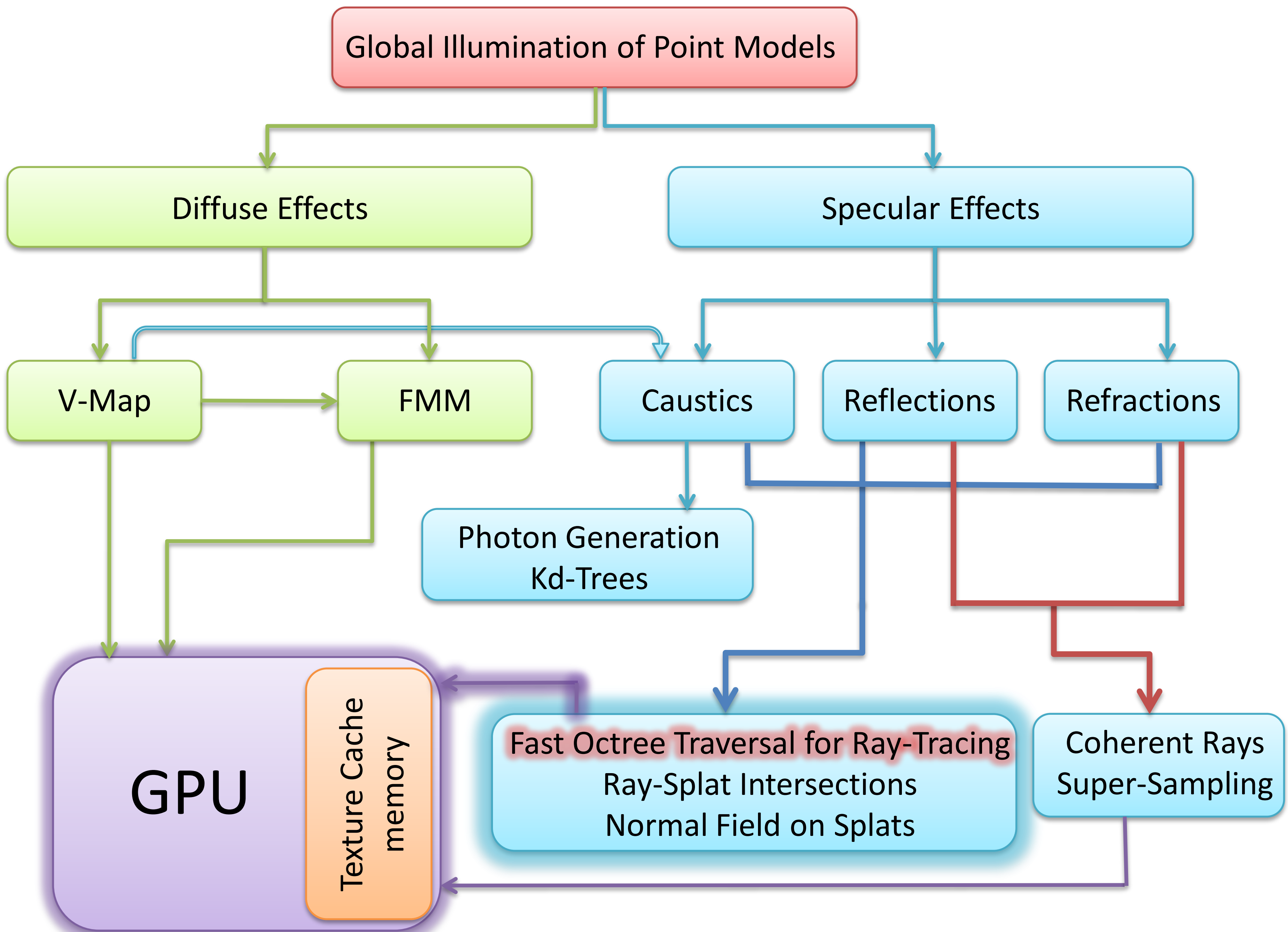
Tracing Photons: Similar To Ray-trace Rendering





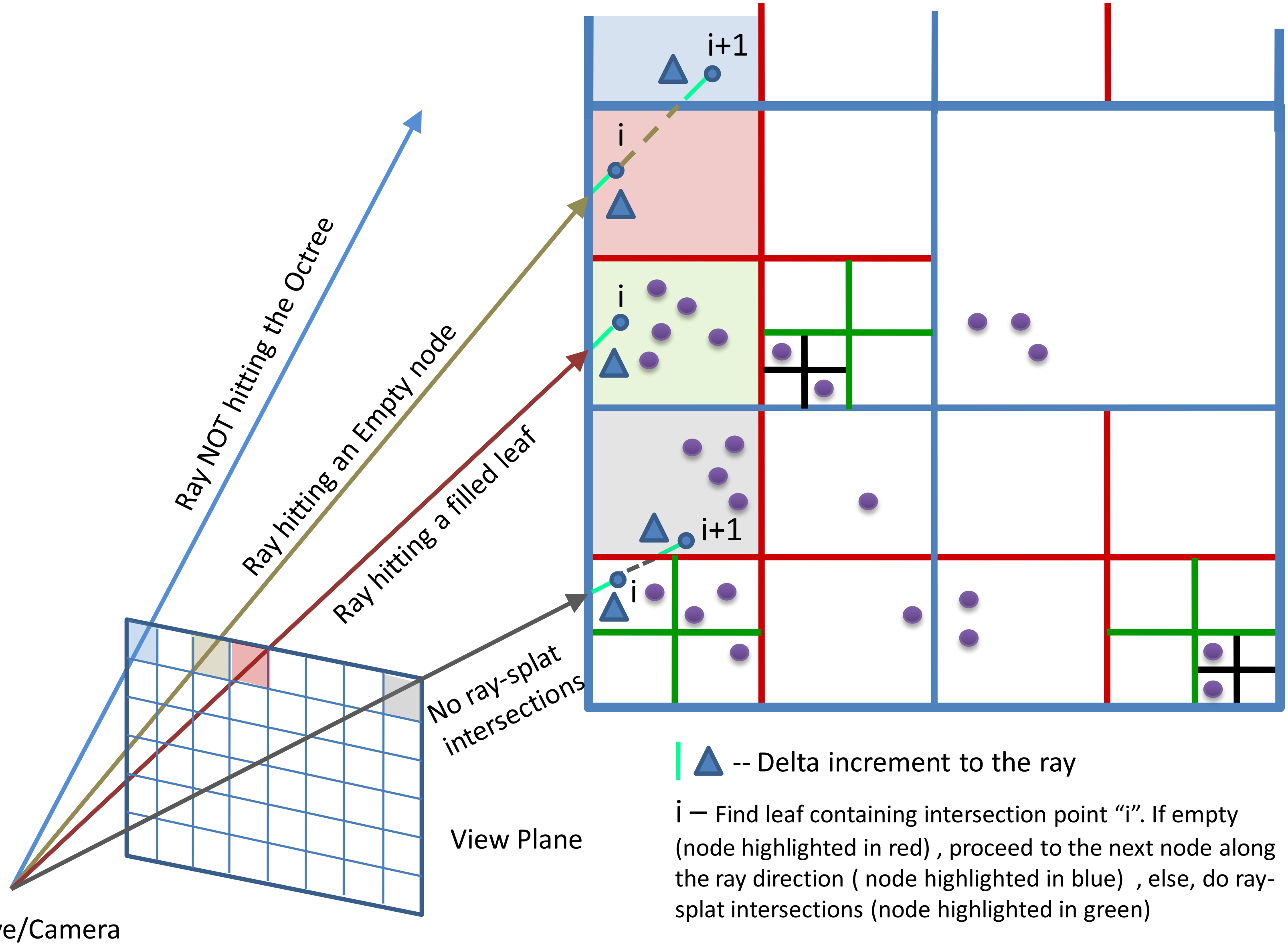


Global Illumination of Point Models
Diffuse Effects
Specular Effects
V-Map
FMM
Caustics
Reflections
Refractions
Photon Generation
Kd-Trees
GPU
Texture Cache
memory
Fast Octree Traversal for Ray-Tracing
Ray-Splat Intersections
Normal Field on Splats
Coherent Rays
Super-Sampling



Global Illumination of Point Models
Diffuse Effects
Specular Effects
V-Map
FMM
Caustics
Reflections
Refractions
Photon Generation
Kd-Trees
GPU
Texture Cache
memory
Fast Octree Traversal for Ray-Tracing
Ray-Splat Intersections
Normal Field on Splats
Coherent Rays
Super-Sampling

Ray-trace Rendering: Octree Traversal

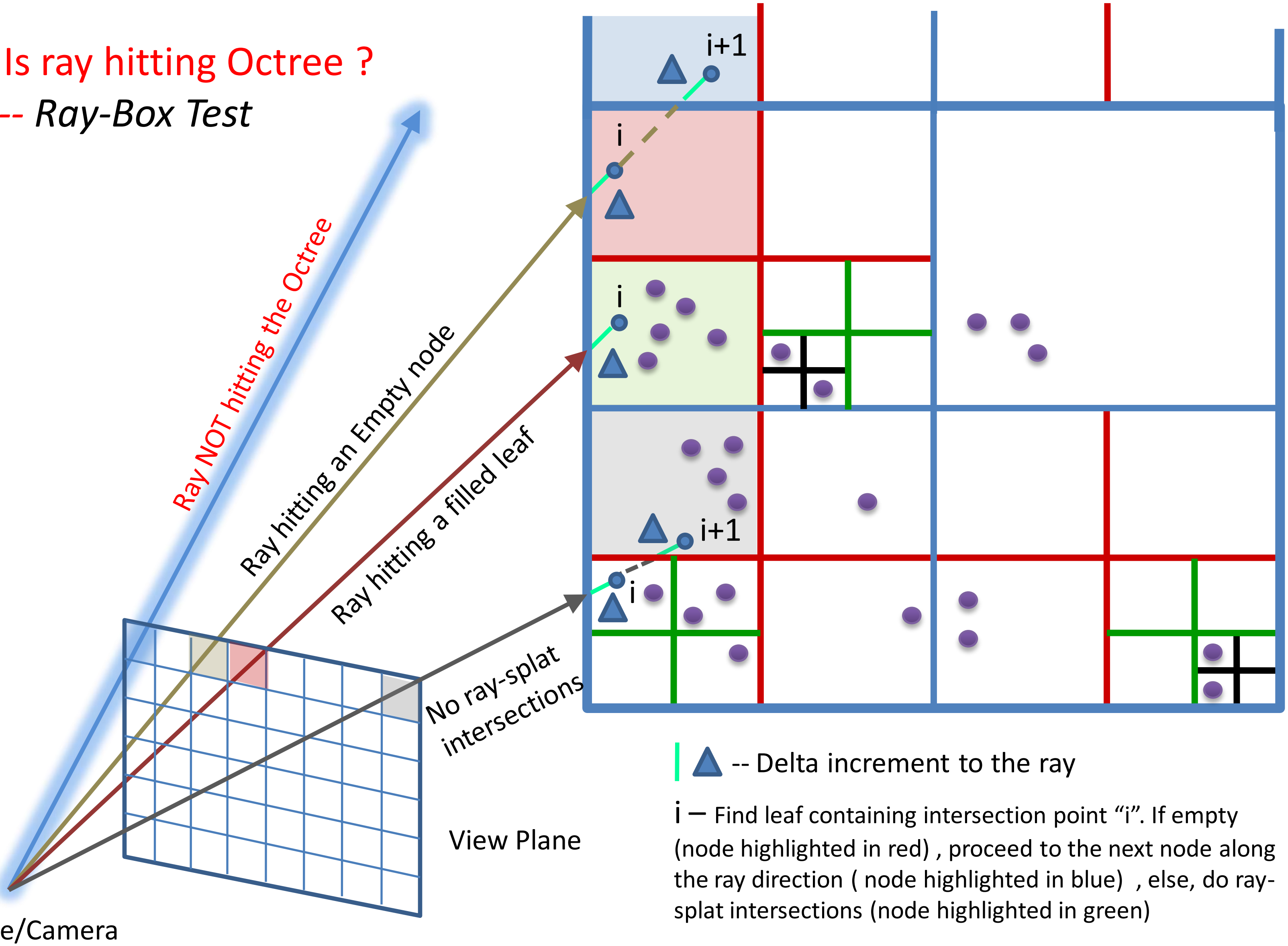


-- Delta increment to the ray

i – Find leaf containing intersection point "i". If empty (node highlighted in red) , proceed to the next node along the ray direction (node highlighted in blue) , else, do ray-splat intersections (node highlighted in green)

Ray-trace Rendering: Octree Traversal

- Is ray hitting Octree ?
  -- *Ray-Box Test*

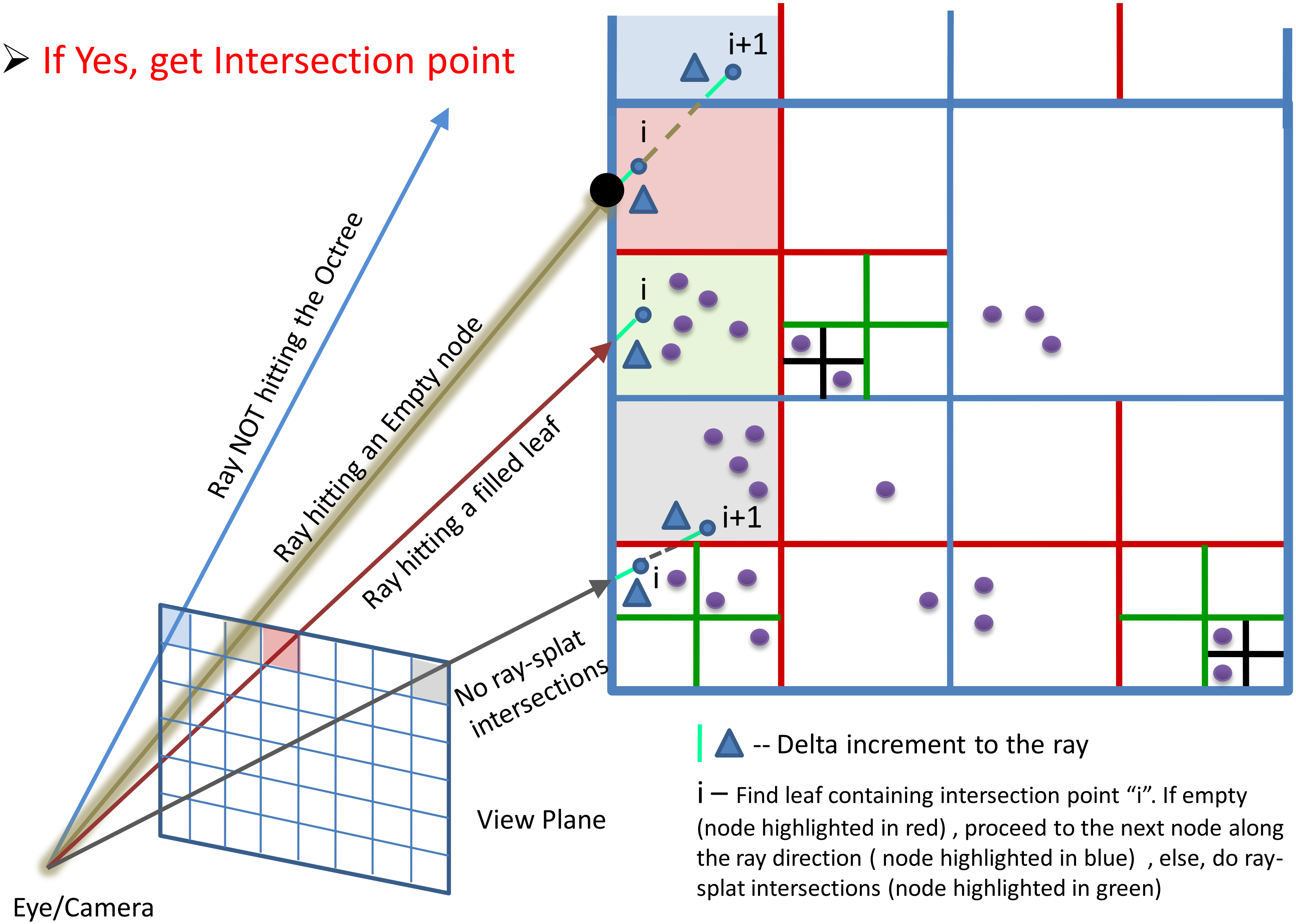


-- Delta increment to the ray

i – Find leaf containing intersection point "i". If empty (node highlighted in red) , proceed to the next node along the ray direction (node highlighted in blue) , else, do ray-splat intersections (node highlighted in green)

Ray-trace Rendering: Octree Traversal

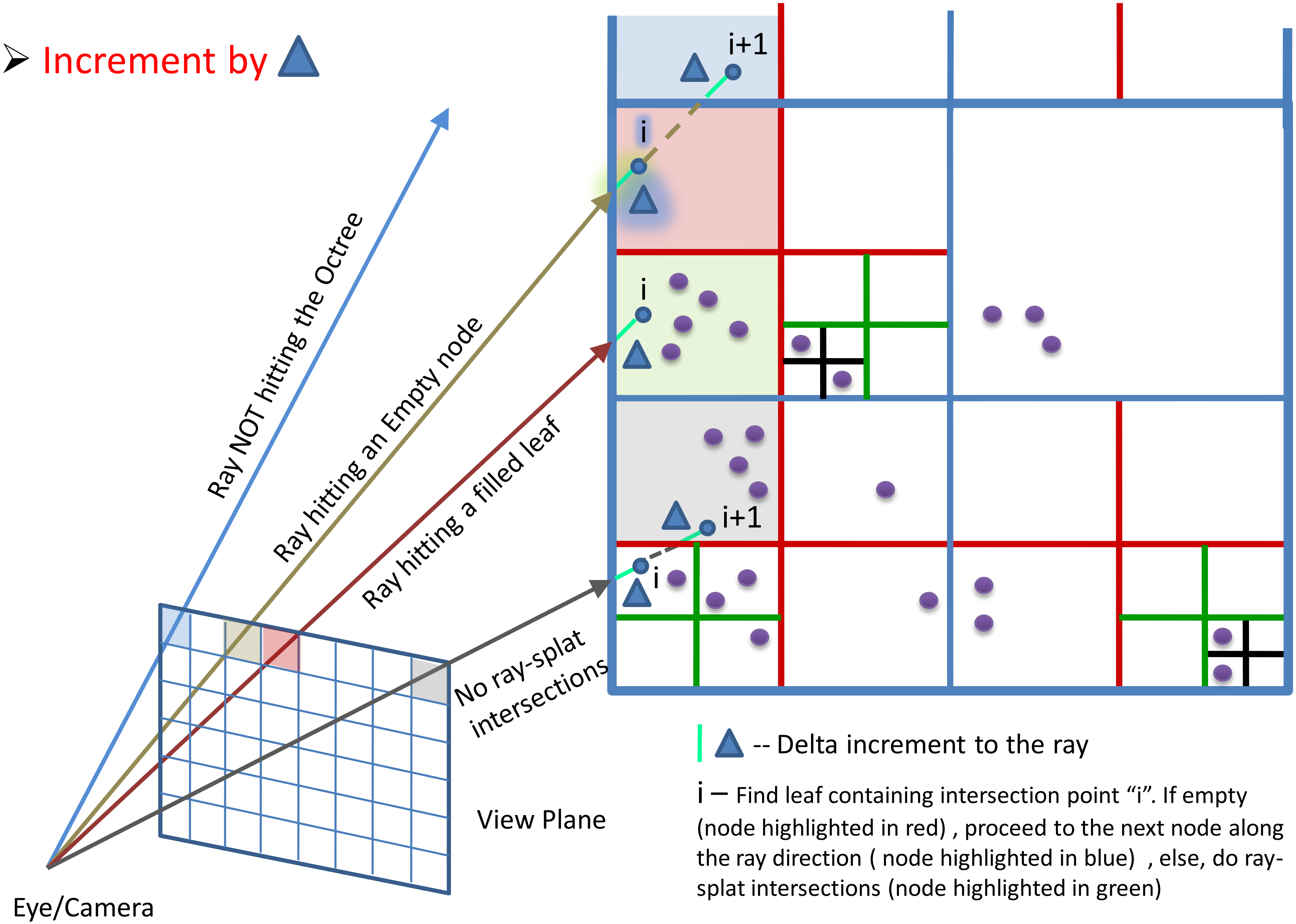
- If Yes, get Intersection point



i – Find leaf containing intersection point "i". If empty (node highlighted in red) , proceed to the next node along the ray direction (node highlighted in blue) , else, do ray-splat intersections (node highlighted in green)

Ray-trace Rendering: Octree Traversal

- Increment by △

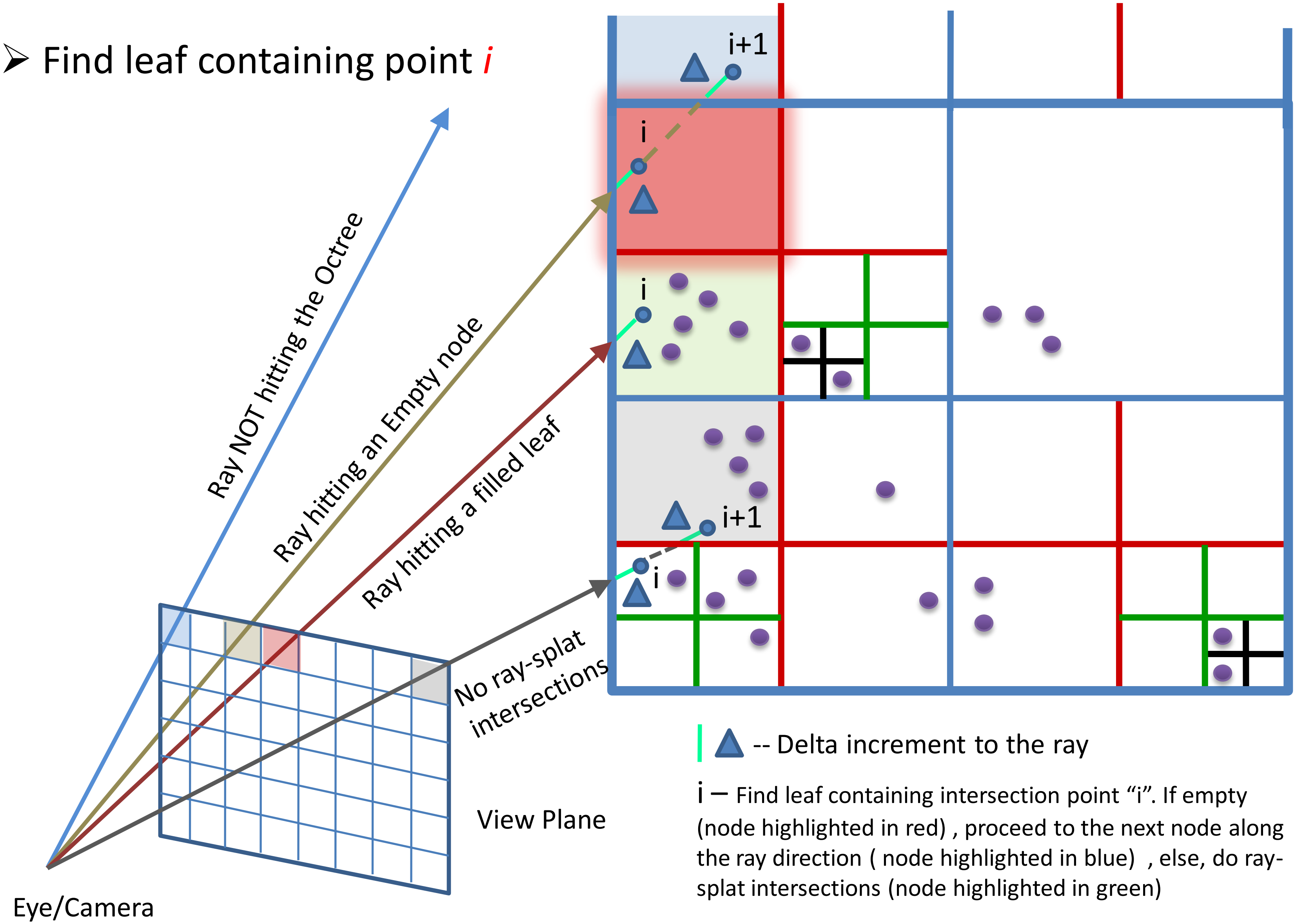


△ -- Delta increment to the ray

i – Find leaf containing intersection point "i". If empty (node highlighted in red) , proceed to the next node along the ray direction (node highlighted in blue) , else, do ray-splat intersections (node highlighted in green)

Ray-trace Rendering: Octree Traversal

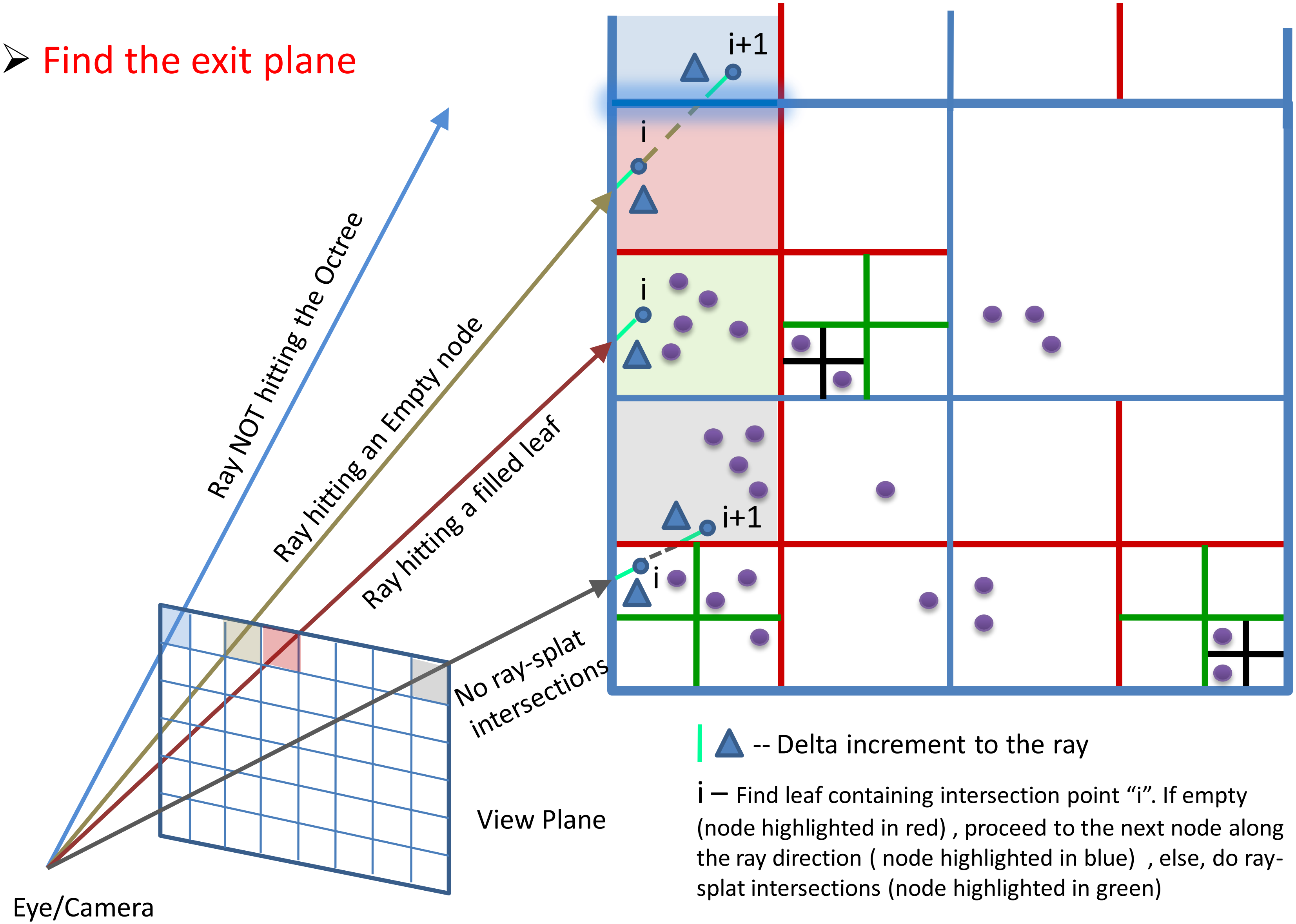
- Find leaf containing point *i*



i – Find leaf containing intersection point "i". If empty (node highlighted in red) , proceed to the next node along the ray direction (node highlighted in blue) , else, do ray-splat intersections (node highlighted in green)

Ray-trace Rendering: Octree Traversal

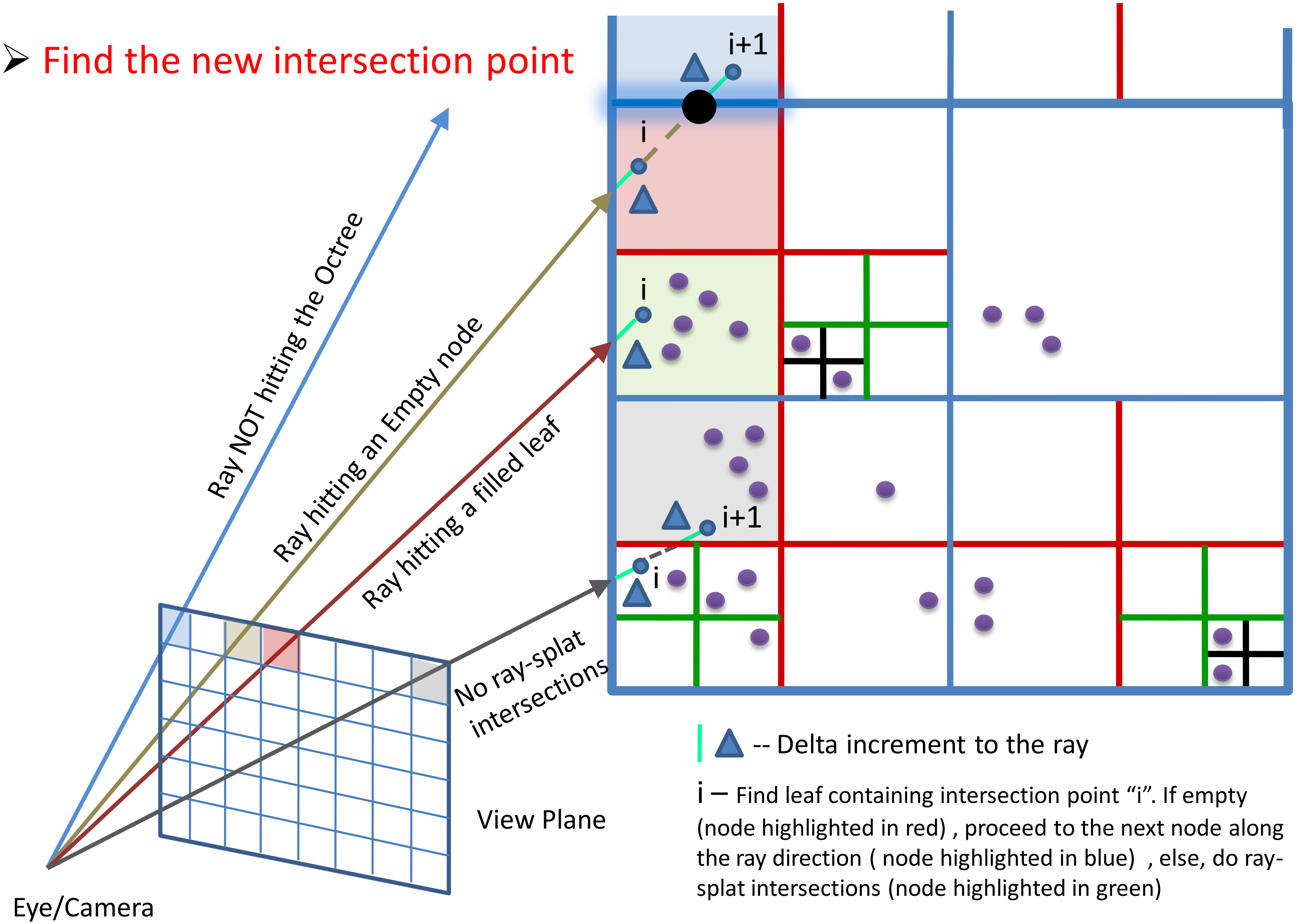
➢ Find the exit plane



i – Find leaf containing intersection point "i". If empty (node highlighted in red) , proceed to the next node along the ray direction (node highlighted in blue) , else, do ray-splat intersections (node highlighted in green)

Ray-trace Rendering: Octree Traversal

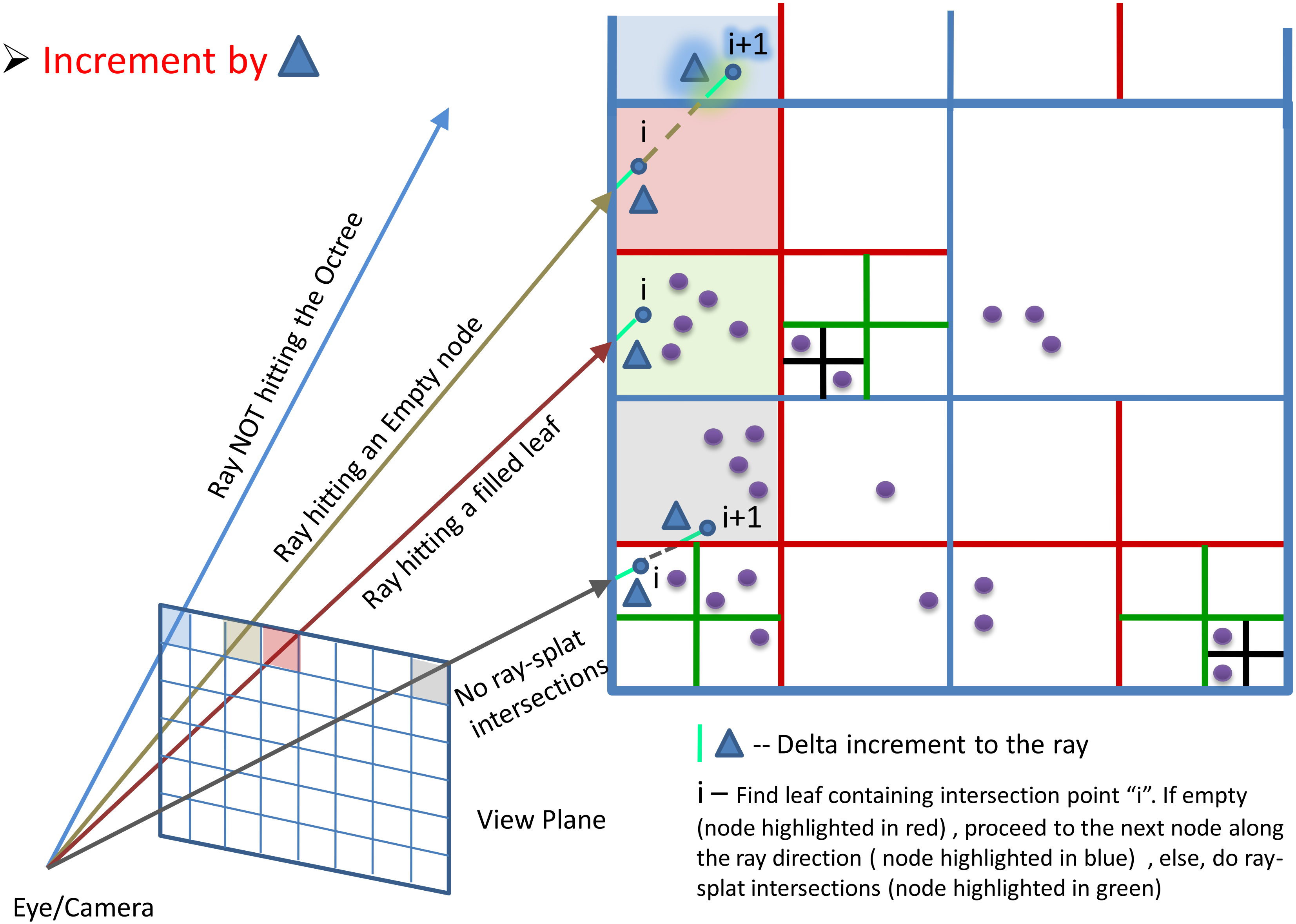
- Find the new intersection point



i – Find leaf containing intersection point "i". If empty (node highlighted in red) , proceed to the next node along the ray direction (node highlighted in blue) , else, do ray-splat intersections (node highlighted in green)

Ray-trace Rendering: Octree Traversal

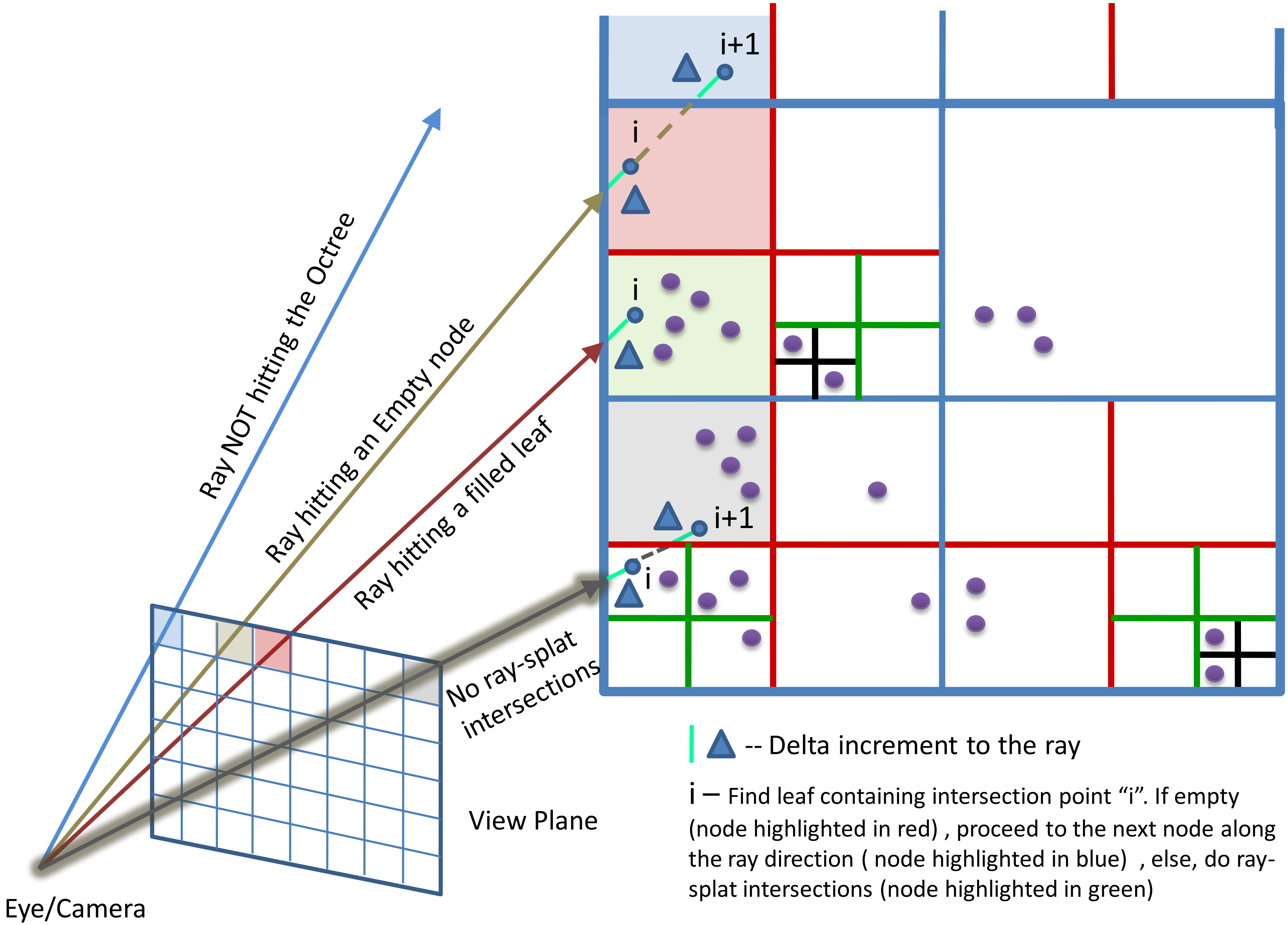
- Increment by △



△ -- Delta increment to the ray

i – Find leaf containing intersection point "i". If empty (node highlighted in red) , proceed to the next node along the ray direction (node highlighted in blue) , else, do ray-splat intersections (node highlighted in green)

Ray-trace Rendering: Octree Traversal

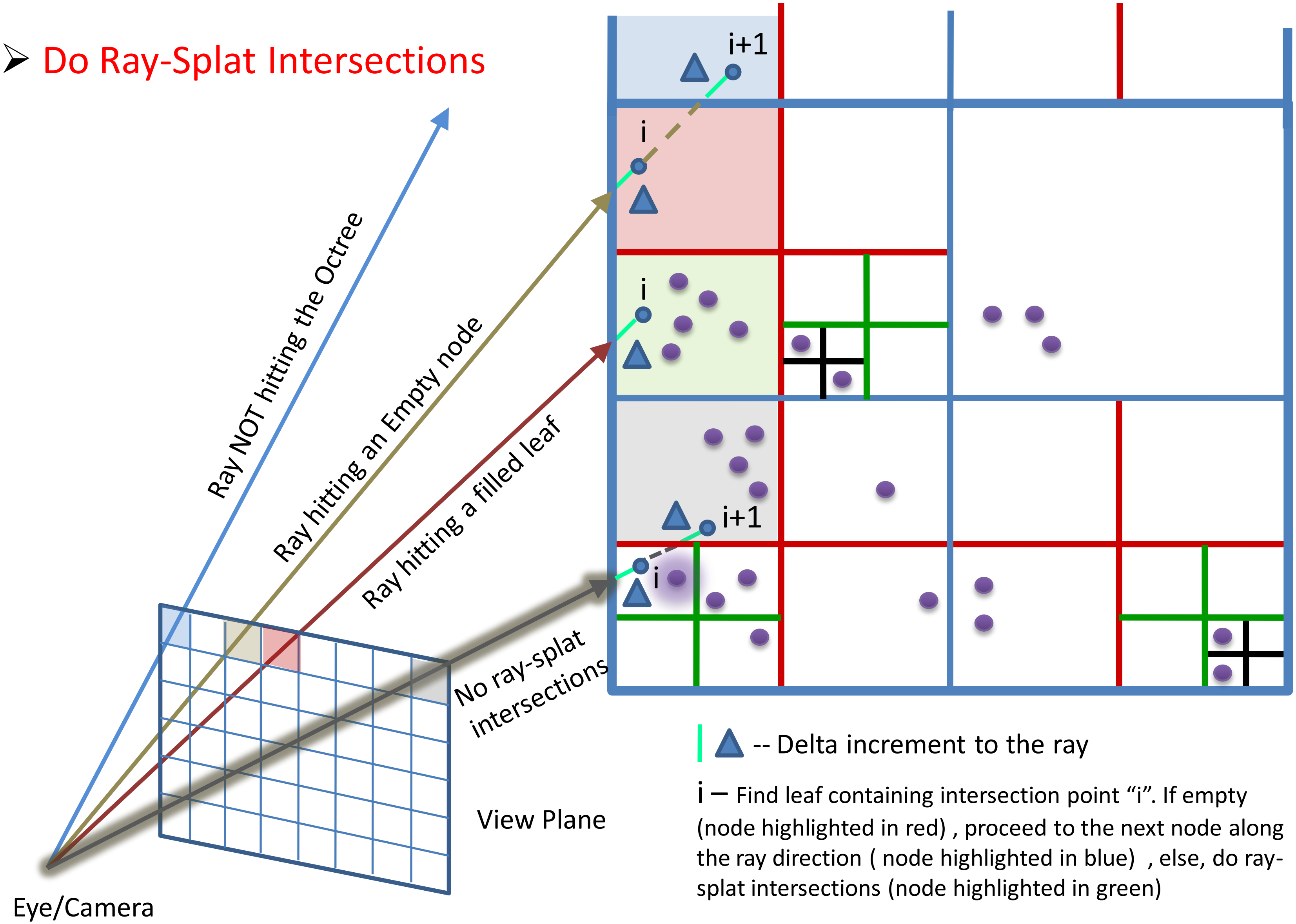


-- Delta increment to the ray

i – Find leaf containing intersection point "i". If empty (node highlighted in red) , proceed to the next node along the ray direction (node highlighted in blue) , else, do ray-splat intersections (node highlighted in green)

Ray-trace Rendering: Octree Traversal

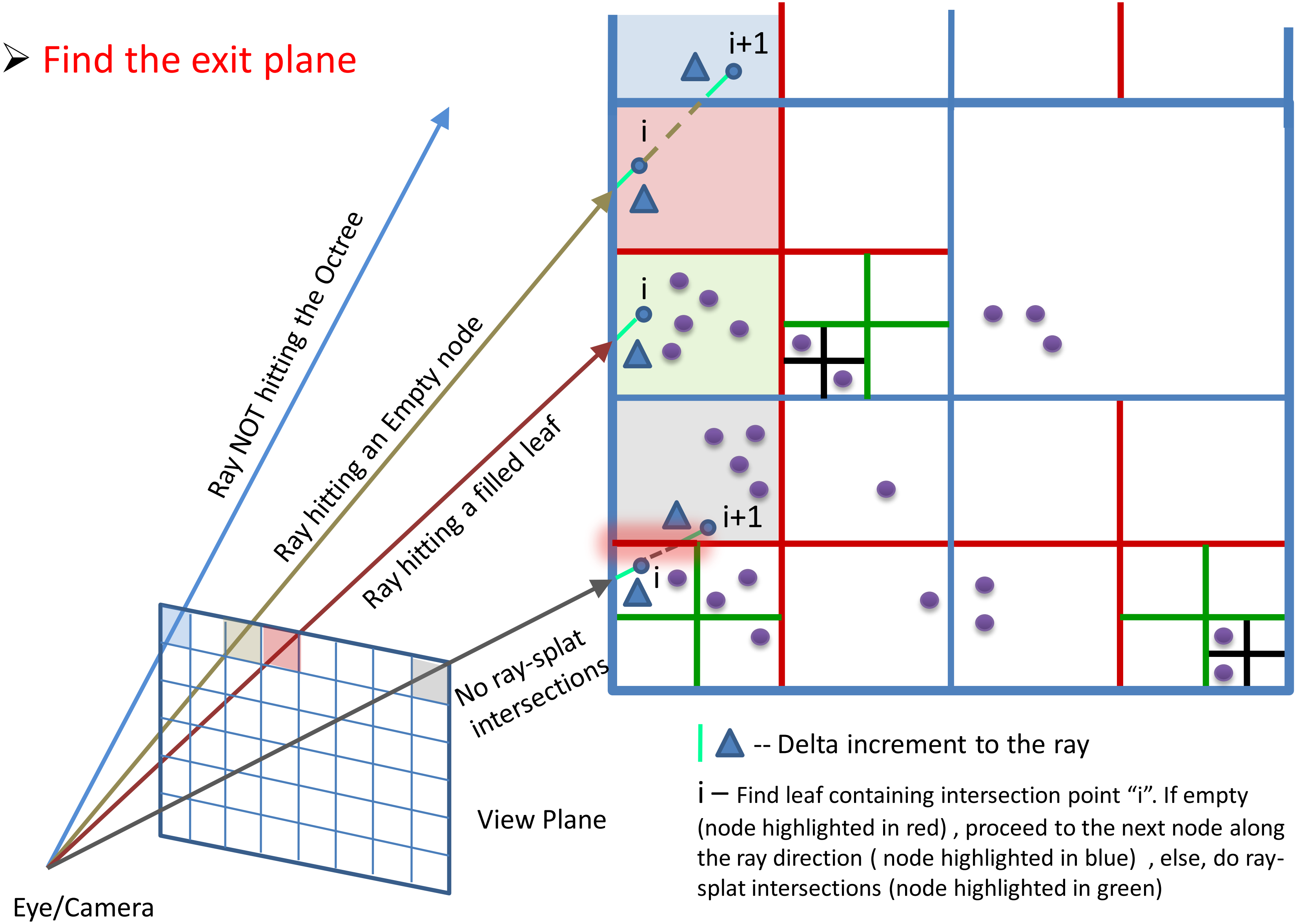
- Do Ray-Splat Intersections



i – Find leaf containing intersection point "i". If empty (node highlighted in red) , proceed to the next node along the ray direction (node highlighted in blue) , else, do ray-splat intersections (node highlighted in green)

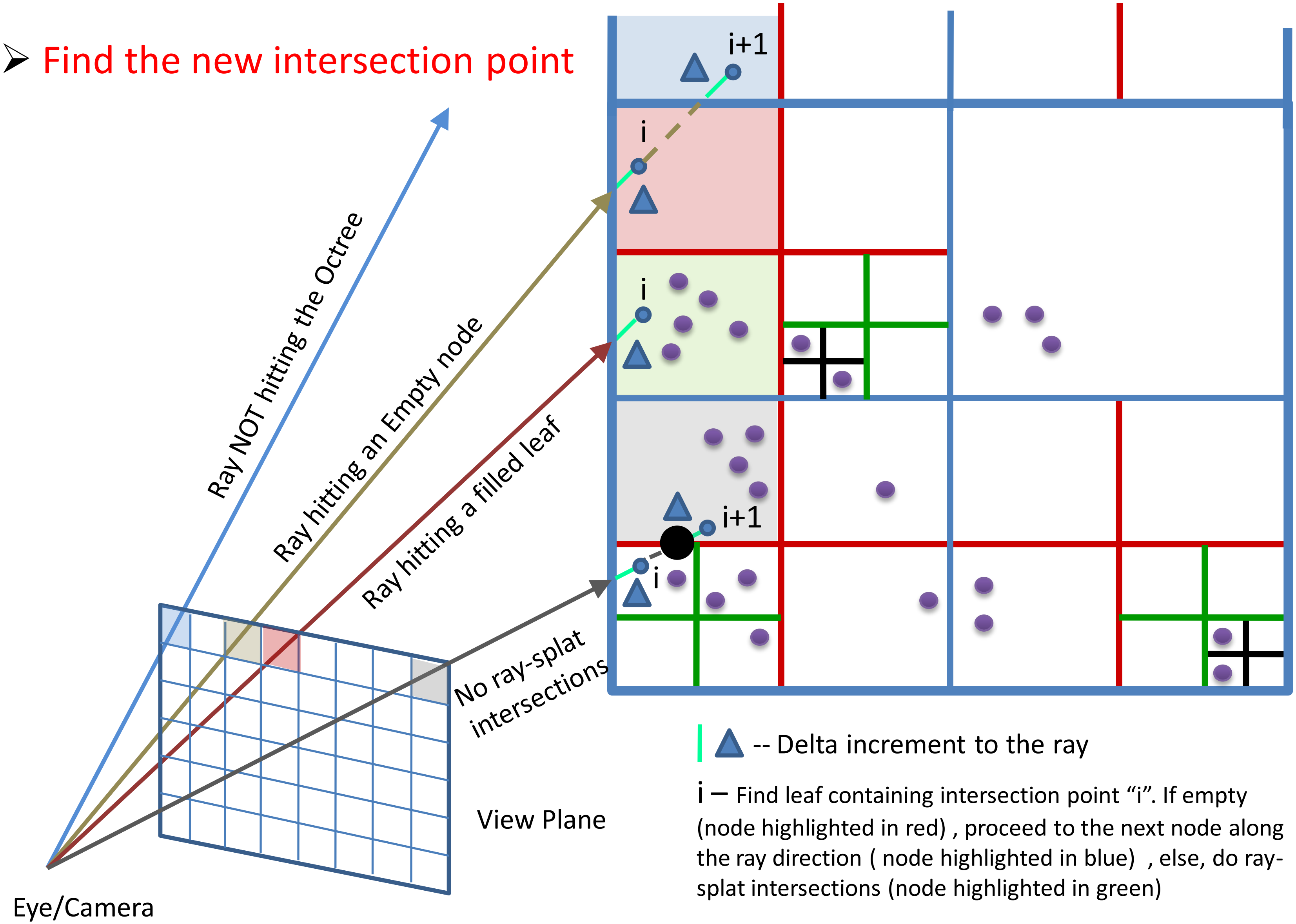
Ray-trace Rendering: Octree Traversal

- Find the exit plane



Ray-trace Rendering: Octree Traversal

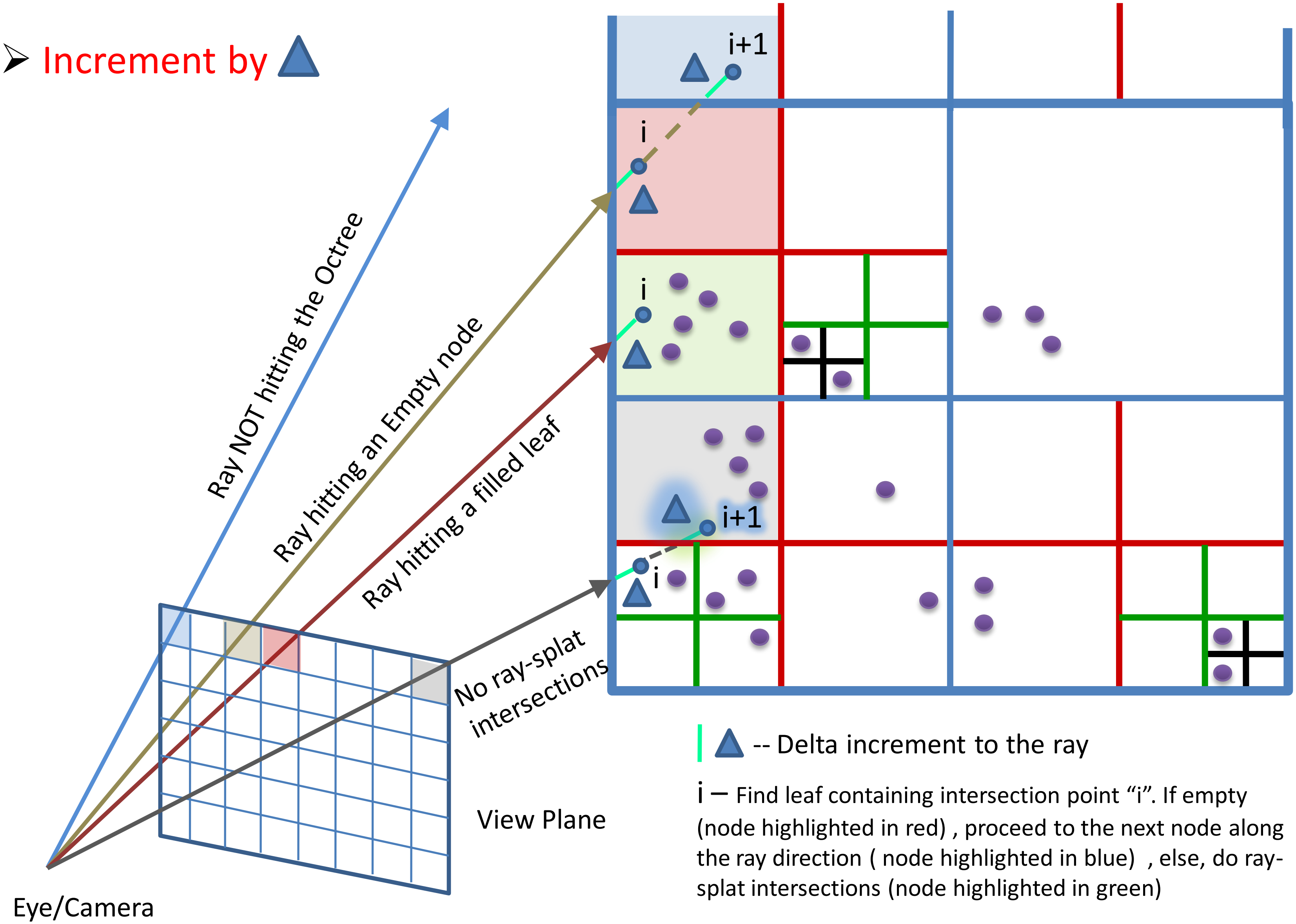
➢ Find the new intersection point



i – Find leaf containing intersection point "i". If empty (node highlighted in red) , proceed to the next node along the ray direction (node highlighted in blue) , else, do ray-splat intersections (node highlighted in green)

Ray-trace Rendering: Octree Traversal

- Increment by △



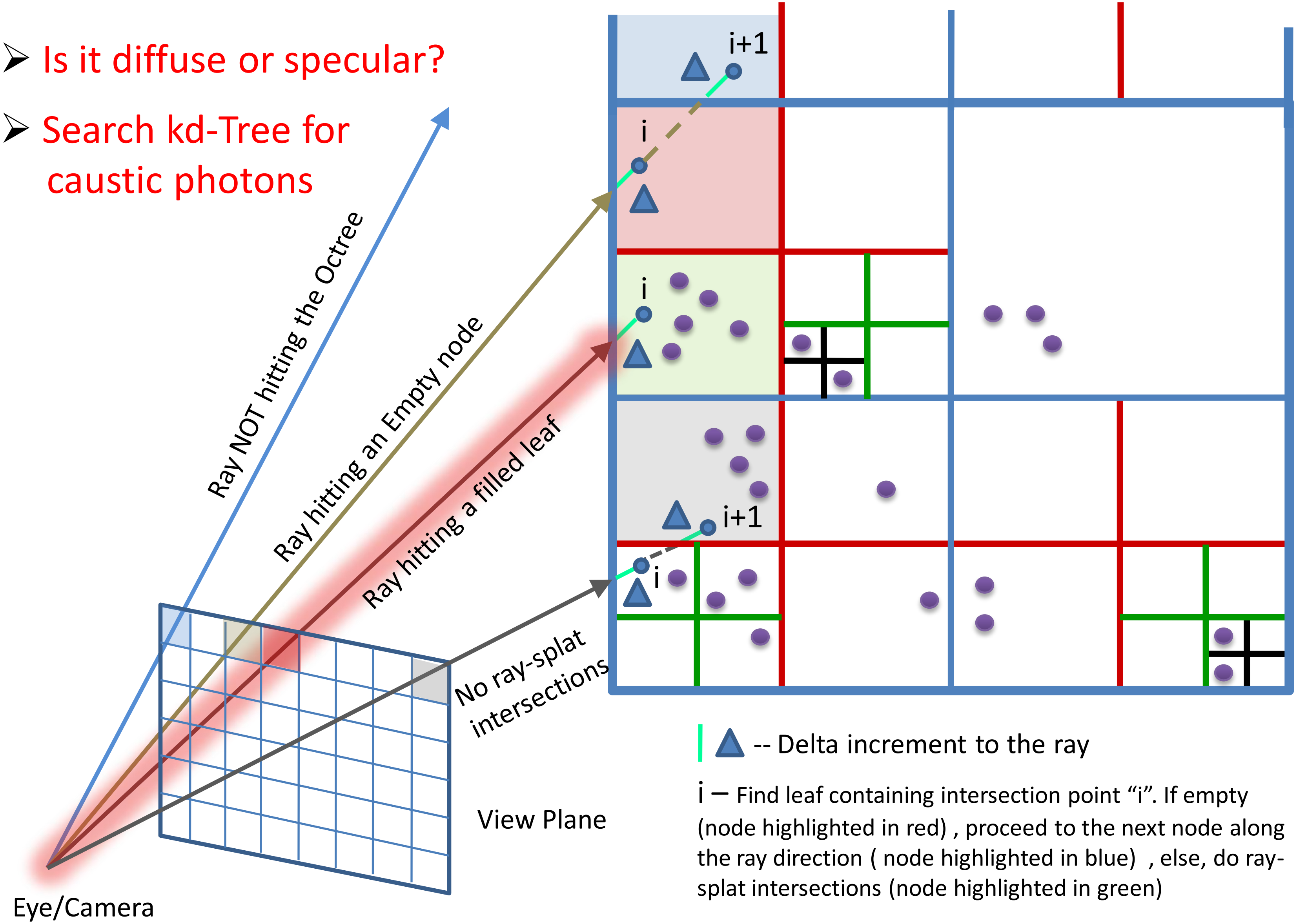
△ -- Delta increment to the ray

i – Find leaf containing intersection point "i". If empty (node highlighted in red) , proceed to the next node along the ray direction (node highlighted in blue) , else, do ray-splat intersections (node highlighted in green)

Rhushabh Goradia, Sharat Chandran

Ray-trace Rendering: Octree Traversal

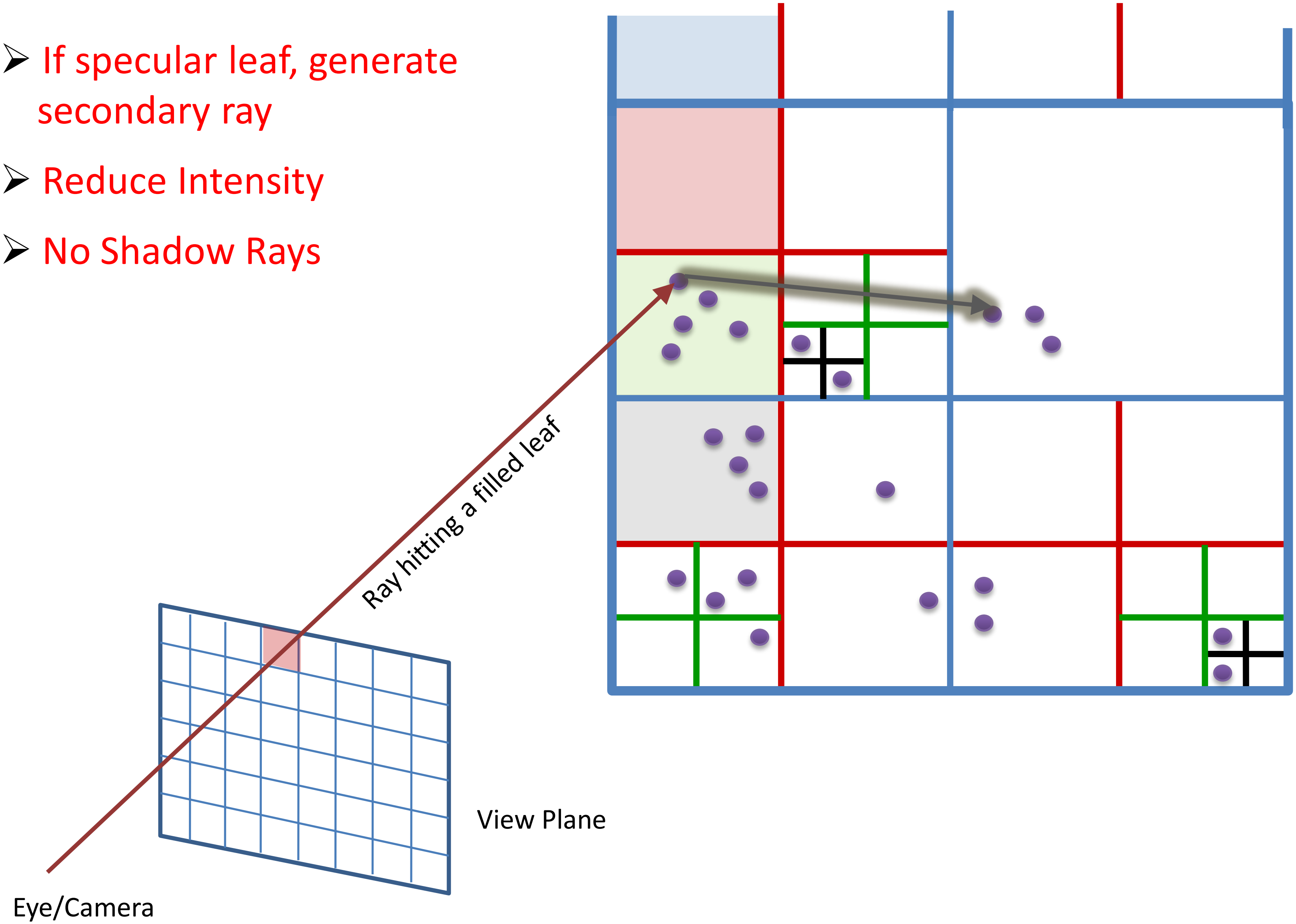
- Is it diffuse or specular?
- Search kd-Tree for caustic photons



i – Find leaf containing intersection point "i". If empty (node highlighted in red) , proceed to the next node along the ray direction (node highlighted in blue) , else, do ray-splat intersections (node highlighted in green)

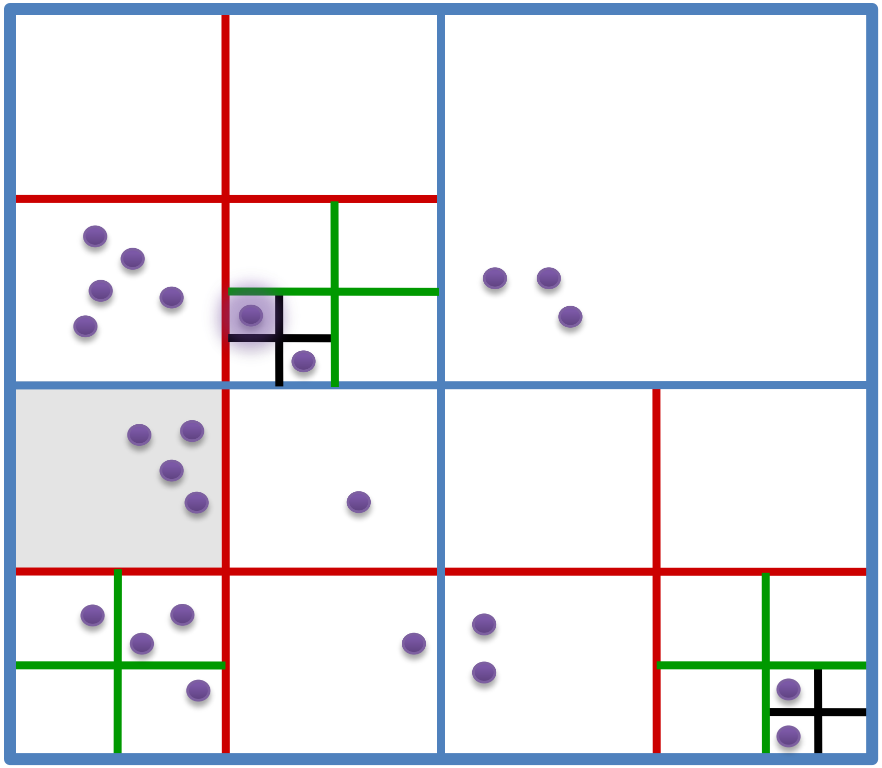
Ray-trace Rendering: Octree Traversal

- If specular leaf, generate secondary ray
- Reduce Intensity
- No Shadow Rays



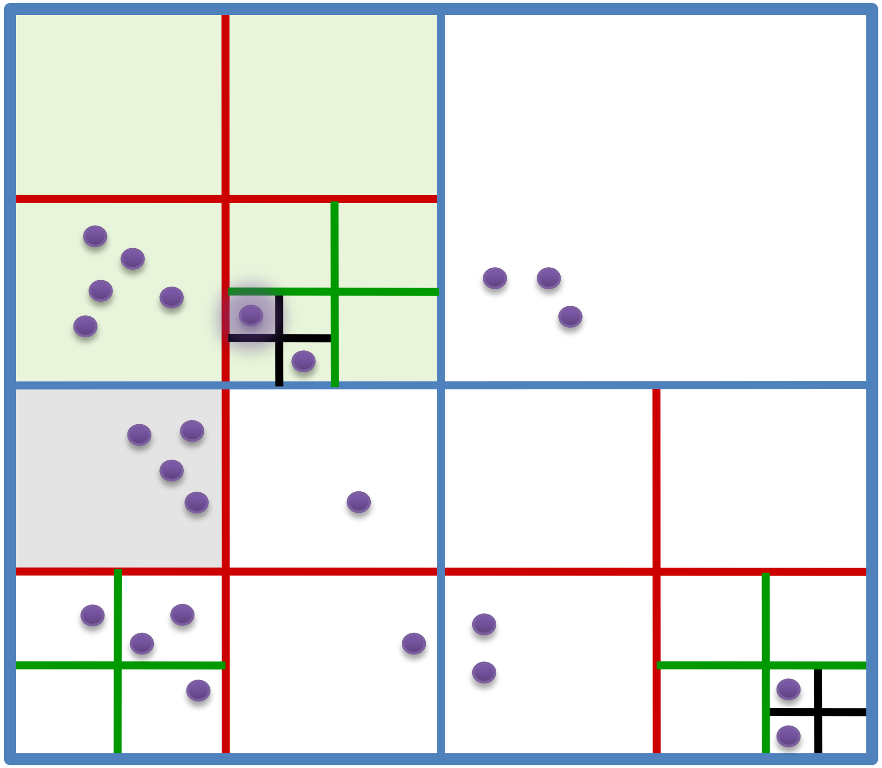
Ray-trace Rendering: Octree Traversal

- We now how we traverse the ray through the leaves of the octree
- *How do we get to the leaf containing the intersection point?*



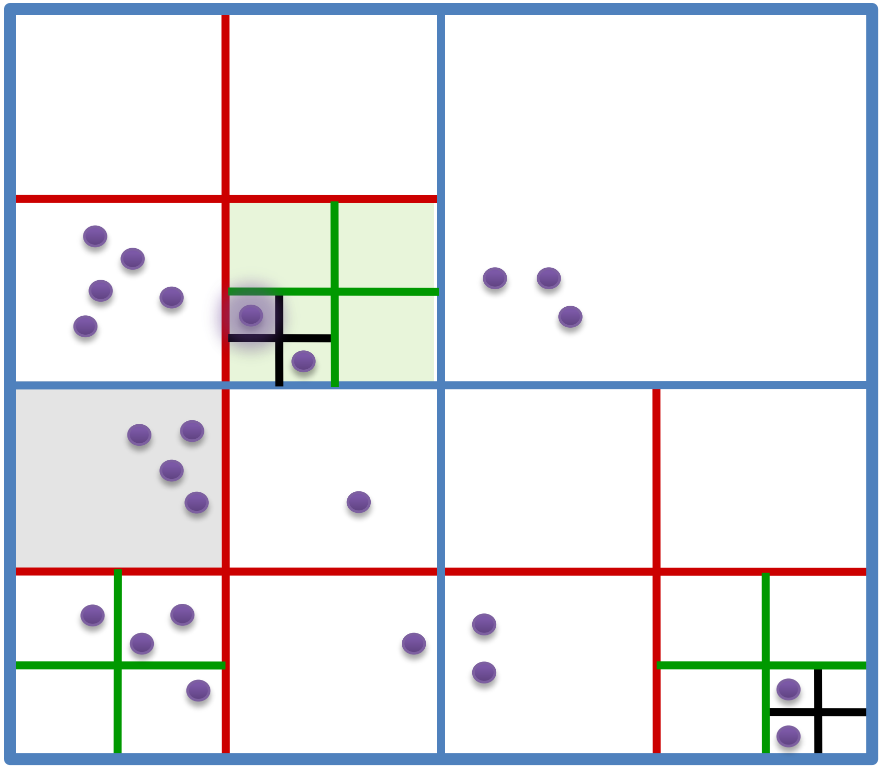
Ray-trace Rendering: Octree Traversal

- We now how we traverse the ray through the leaves of the octree
- *How do we get to the leaf containing the intersection point?*



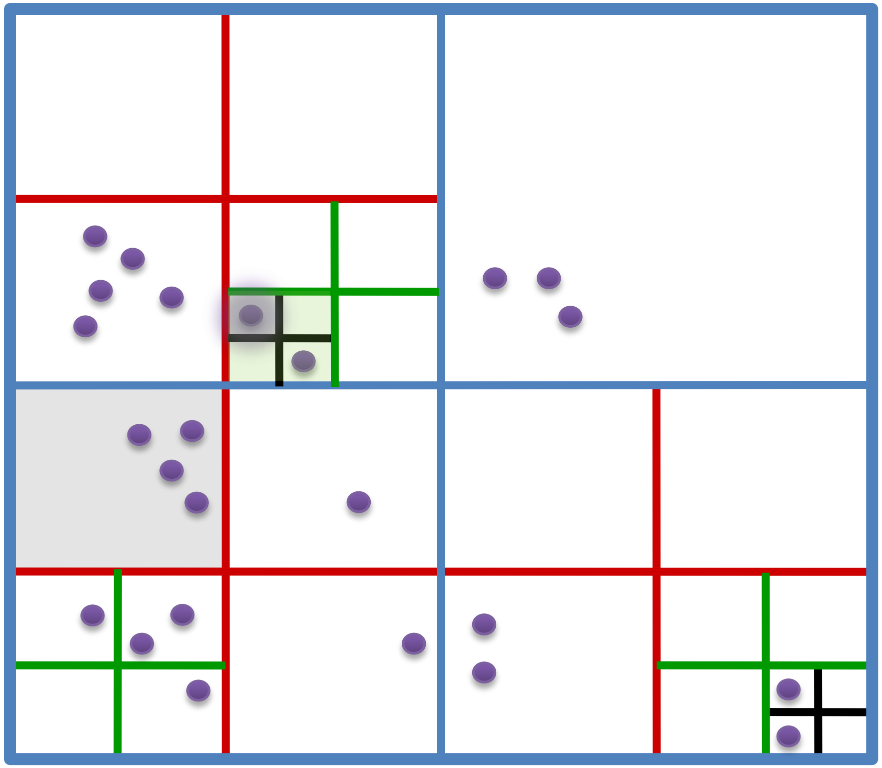
Ray-trace Rendering: Octree Traversal

- We now how we traverse the ray through the leaves of the octree
- *How do we get to the leaf containing the intersection point?*



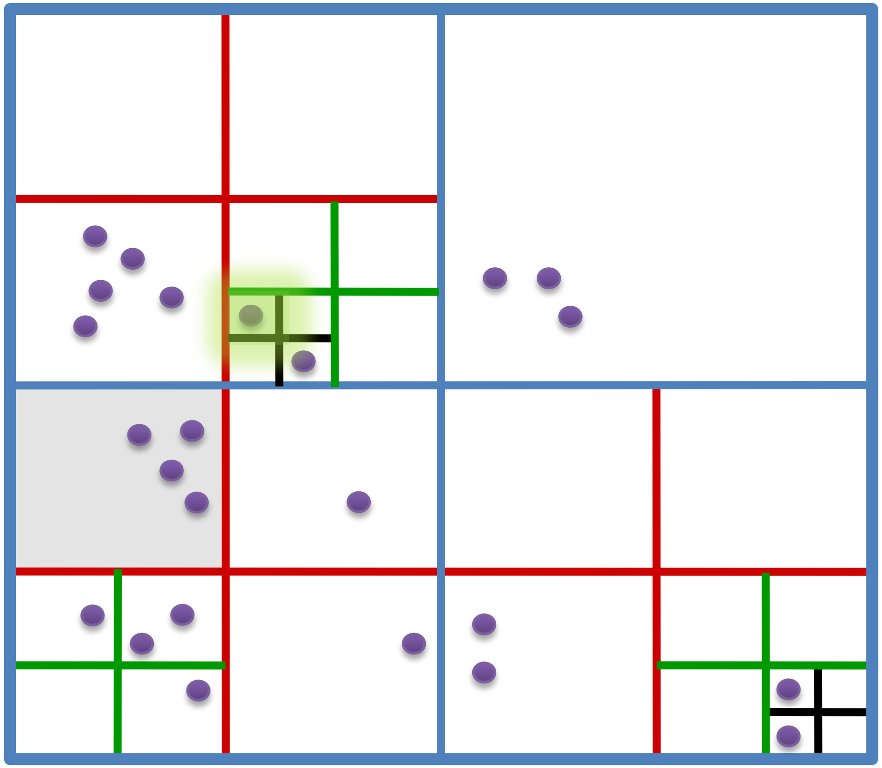
Ray-trace Rendering: Octree Traversal

- We now how we traverse the ray through the leaves of the octree
- *How do we get to the leaf containing the intersection point?*



Ray-trace Rendering: Octree Traversal

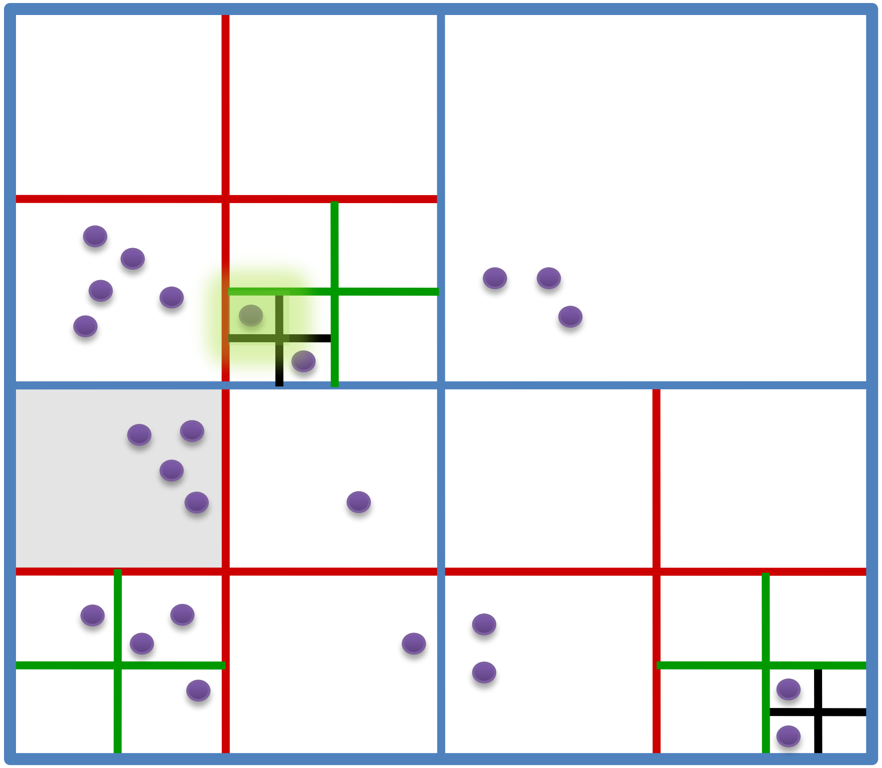
- We now how we traverse the ray through the leaves of the octree
- *How do we get to the leaf containing the intersection point?*



Ray-trace Rendering: Octree Traversal

- We now how we traverse the ray through the leaves of the octree
- *How do we get to the leaf containing the intersection point?*

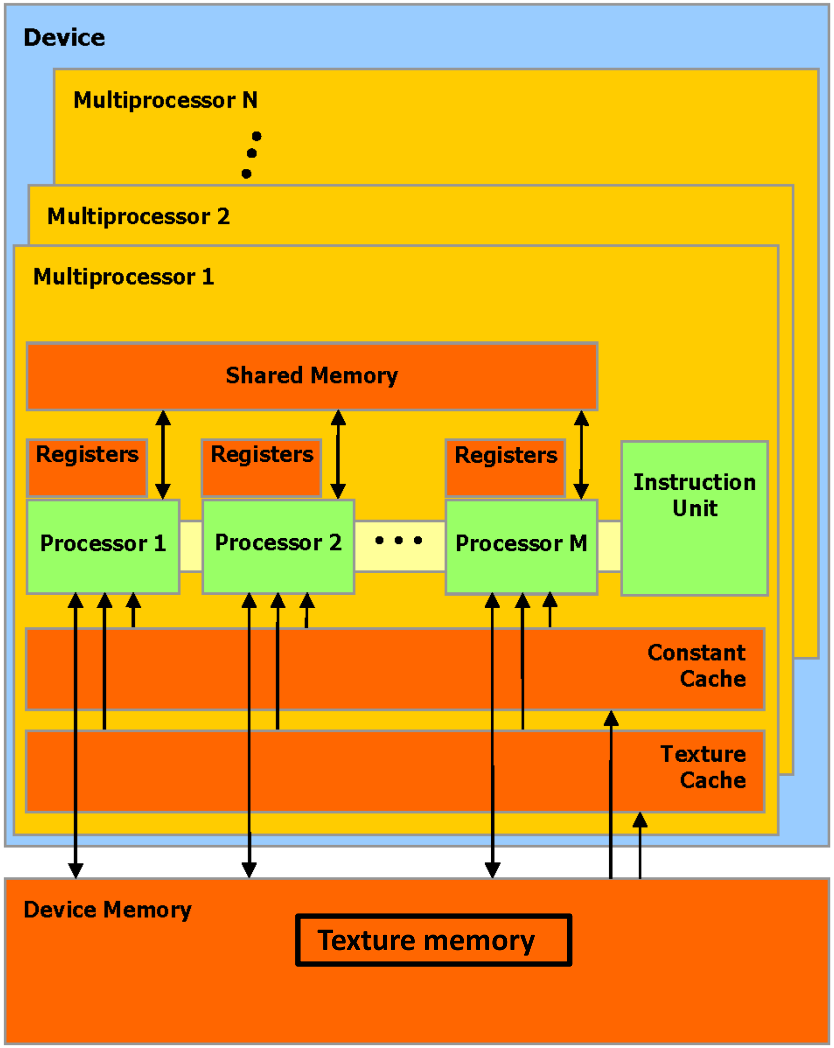
- No condition checks per level
- Use SFC ordering



Ray-trace Rendering: Octree Traversal

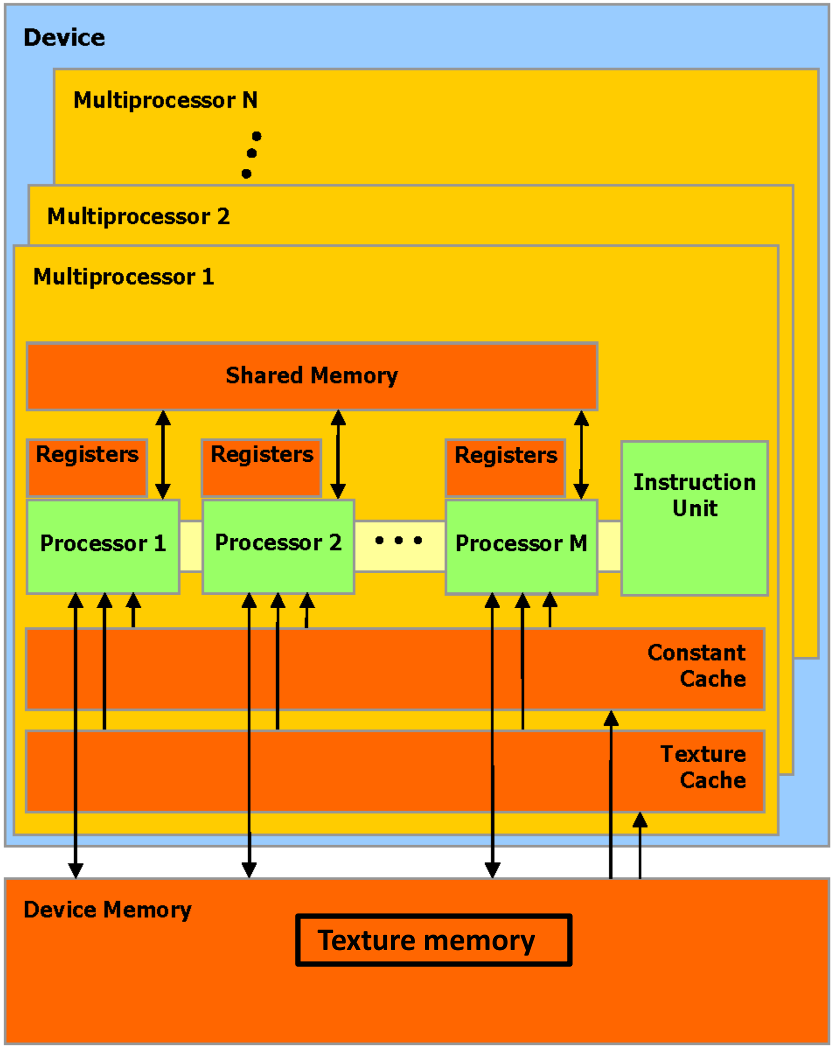
- We now how we traverse the ray through the leaves of the octree
- *How do we get to the leaf containing the intersection point?*
- *….. On GPU !*

Memory Map on CUDA-GPU



Memory Map on CUDA-GPU

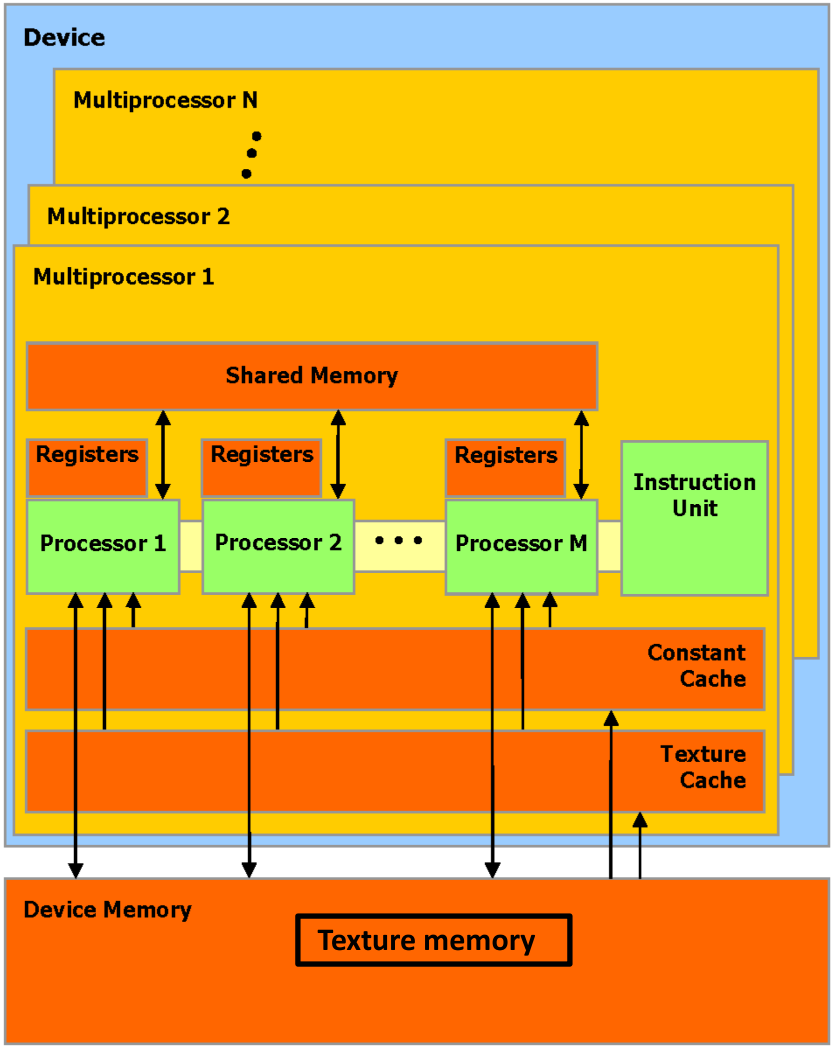
- Store Octree as a texture on GPU
- Improve texture cache-hits



Rhushabh Goradia, Sharat Chandran

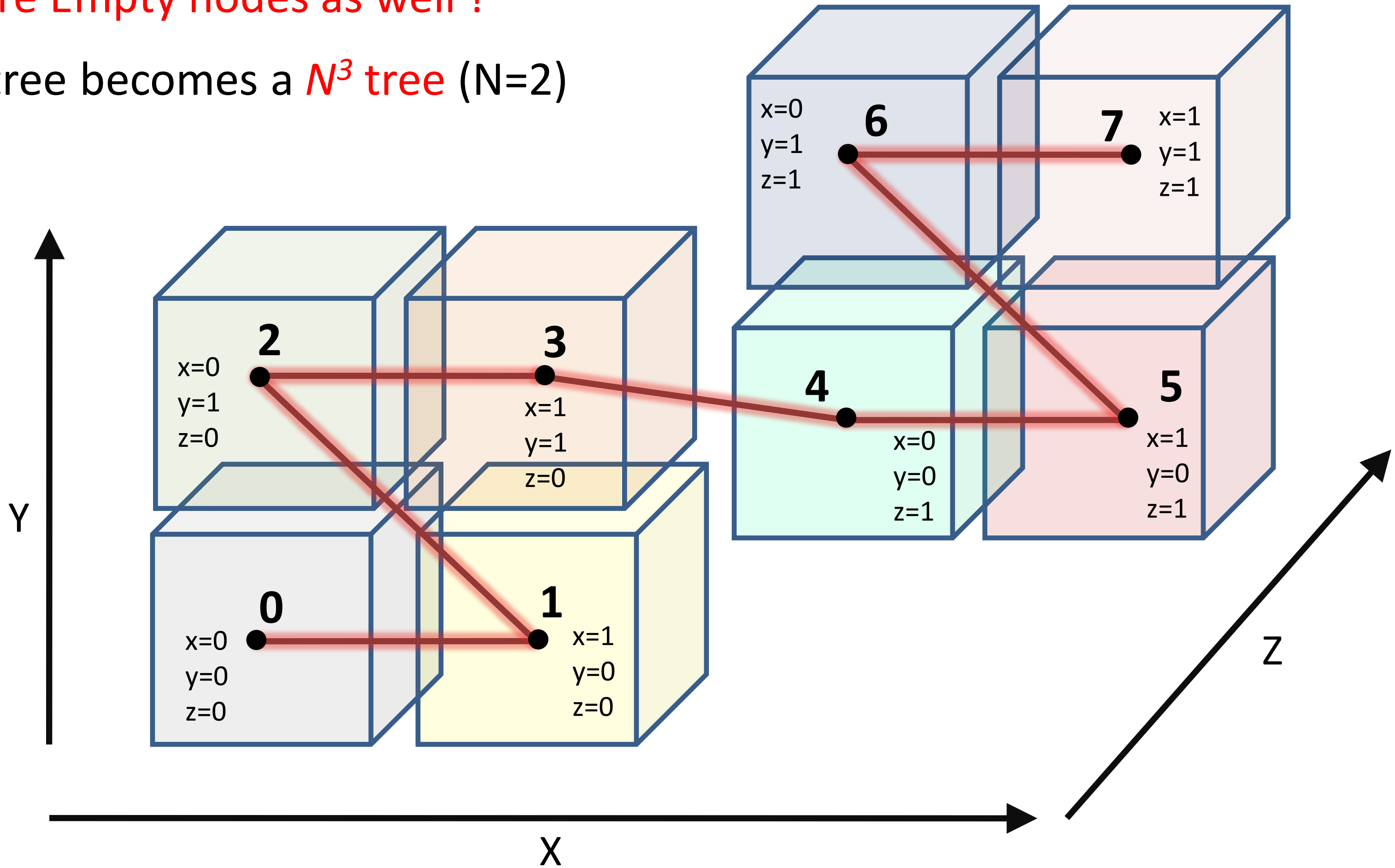
Memory Map on CUDA-GPU

- Store Octree as a texture on GPU
- Improve texture cache-hits
- Store Octree as an array on CPU

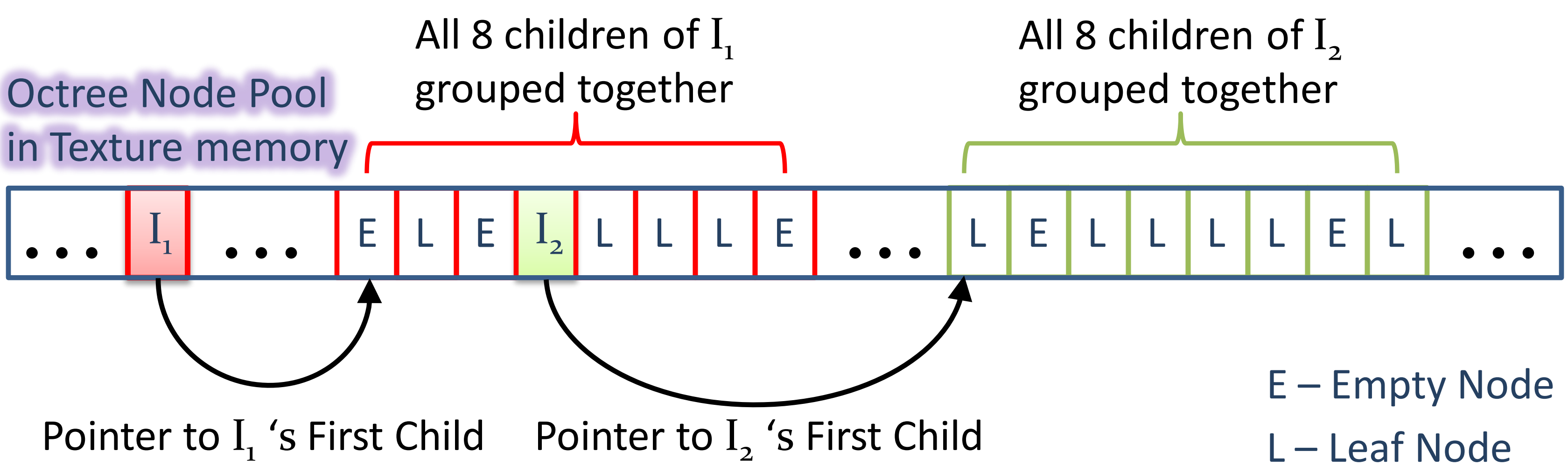


Ray-trace Rendering: Octree Traversal

- Store Empty nodes as well !
- Octree becomes a N^3 tree (N=2)



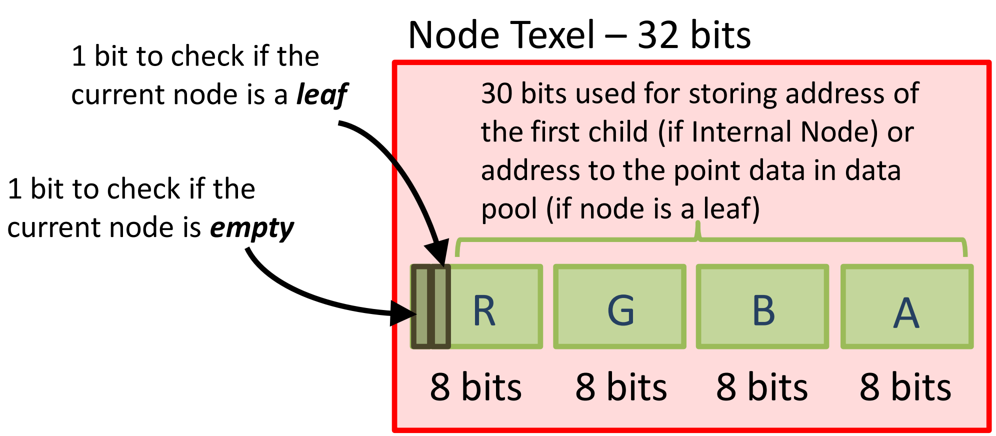
Ray-trace Rendering: Octree Traversal



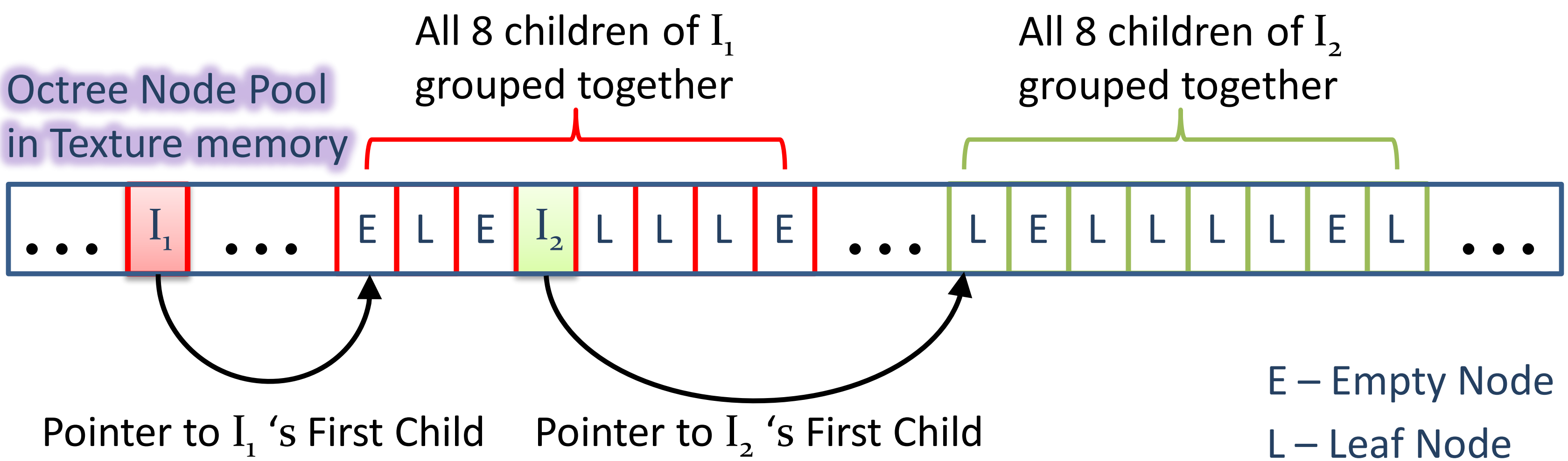
E – Empty Node

L – Leaf Node

I – Internal Node



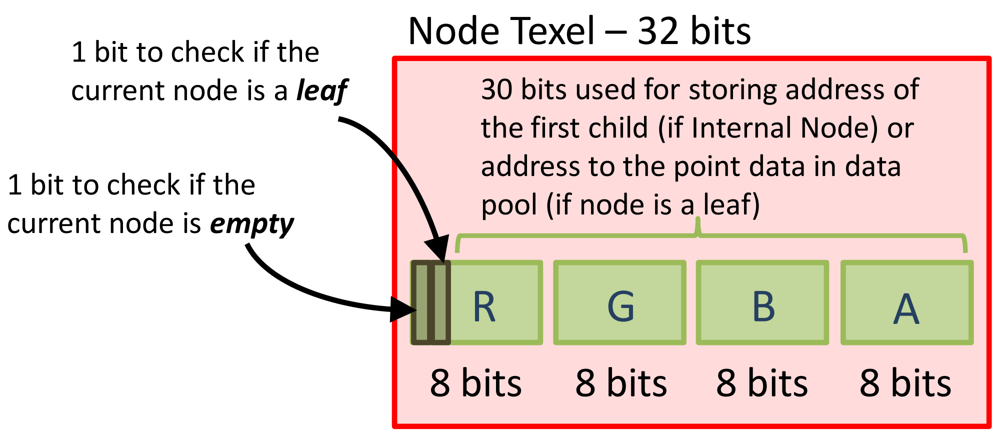
Ray-trace Rendering: Octree Traversal



E – Empty Node

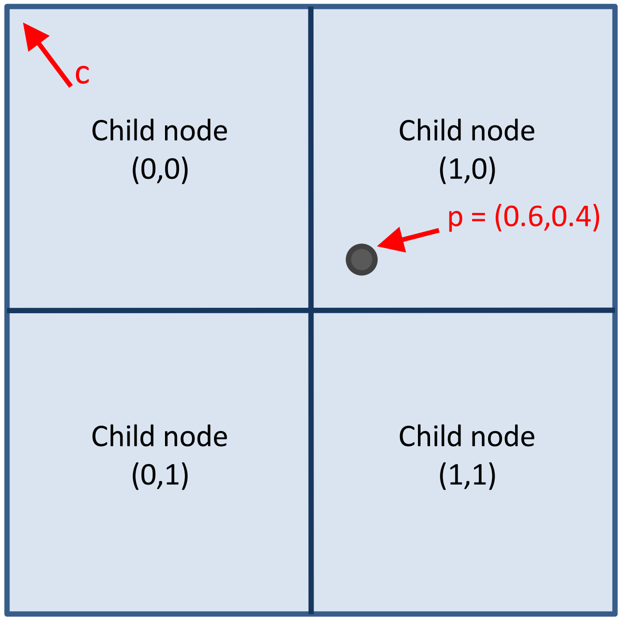
L – Leaf Node

I – Internal Node



- No parent pointer required
- Point data stored as another *1D texture*

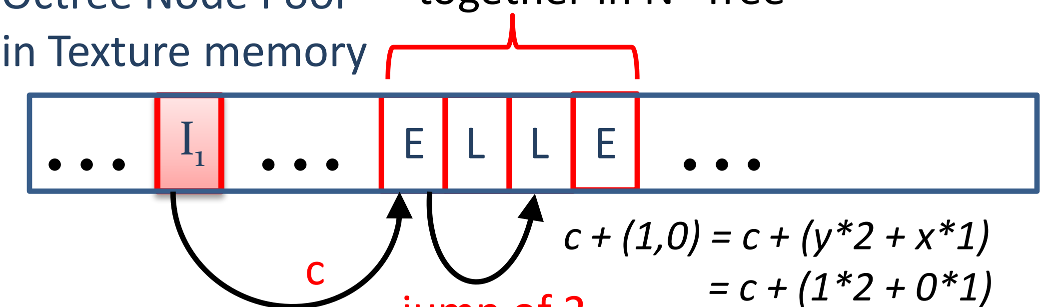
Ray-trace Rendering: Octree Traversal



$n = c + int(p * N)$
$= c + int((0.6*2, 0.4*2))$
$= c + int((1.2,0.8))$
$= c + (1,0)$

All 4 children of I_1 grouped together in N^2 Tree

Octree Node Pool in Texture memory



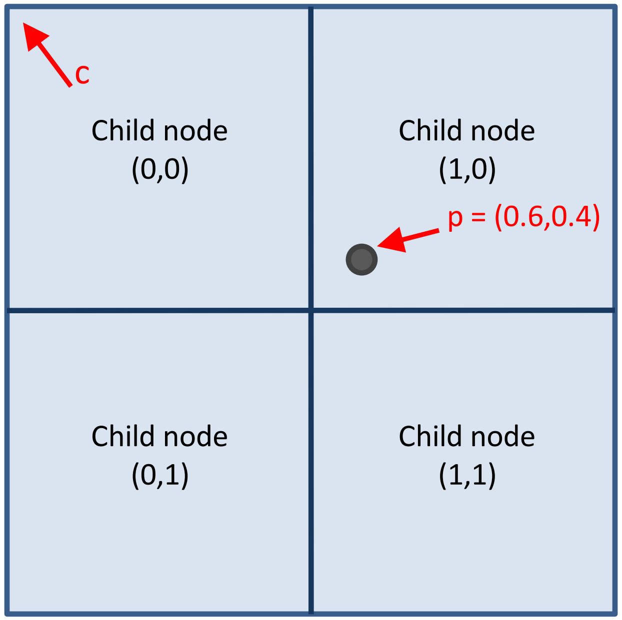
Pointer to I_1 's First Child

$c + (1,0) = c + (y*2 + x*1)$
$= c + (1*2 + 0*1)$
$= c + 2$
$=$ *Third Child*

$$n = c + \text{int}(p * N)$$

E – Empty Node   L – Leaf Node   I – Internal Node

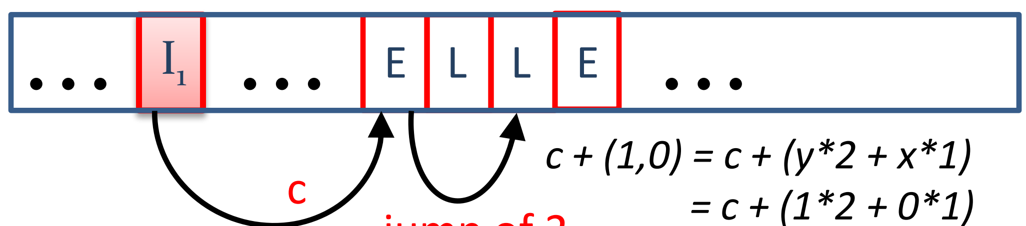
Ray-trace Rendering: Octree Traversal



*n = c + int(p * N)*
*= c + int((0.6*2, 0.4*2))*
= c + int((1.2,0.8))
= c + (1,0)

Octree Node Pool in Texture memory

All 4 children of I_1 grouped together in N^2 Tree



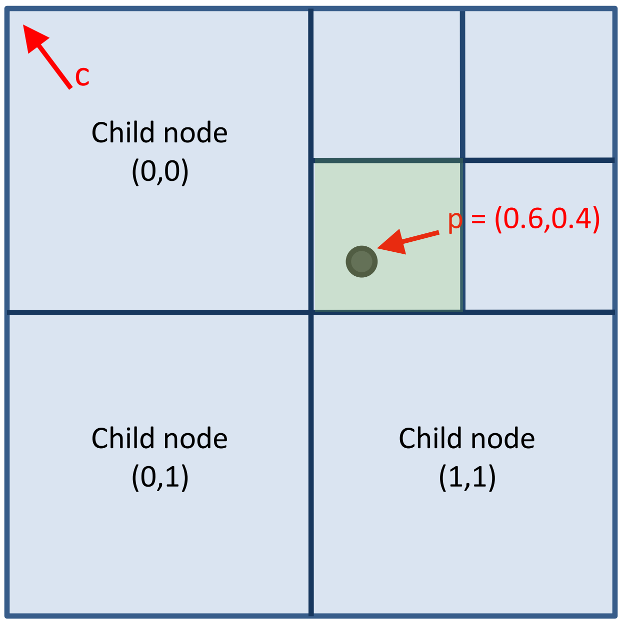
Pointer to I_1 's First Child

*c + (1,0) = c + (y*2 + x*1)*
*= c + (1*2 + 0*1)*
= c + 2
= Third Child

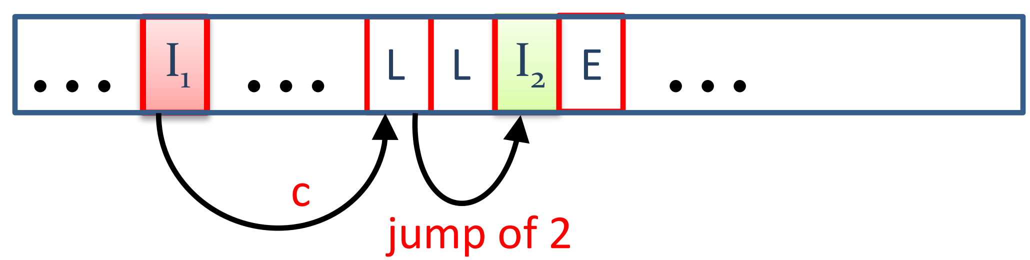
$$n = c + \mathrm{int}(p * N)$$

E – Empty Node L – Leaf Node I – Internal Node

Ray-trace Rendering: Octree Traversal



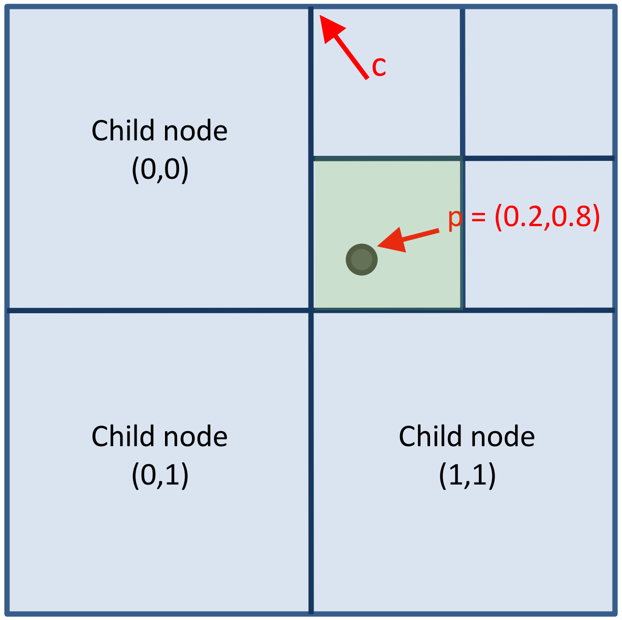
Octree Node Pool
in Texture memory



- Update *p*

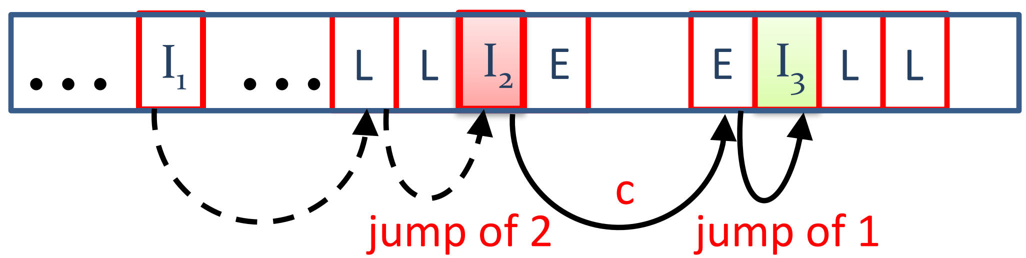
$$p = p * N - \mathrm{int}(p * N)$$

Ray-trace Rendering: Octree Traversal



*p = p * N + int(p * N)*
*= (0.6*2, 0.4*2) + int((0.6*2, 0.4*2))*
= (1.2,0.8) + int((1.2,0.8))
= (0.2,0.8)

Octree Node Pool
in Texture memory

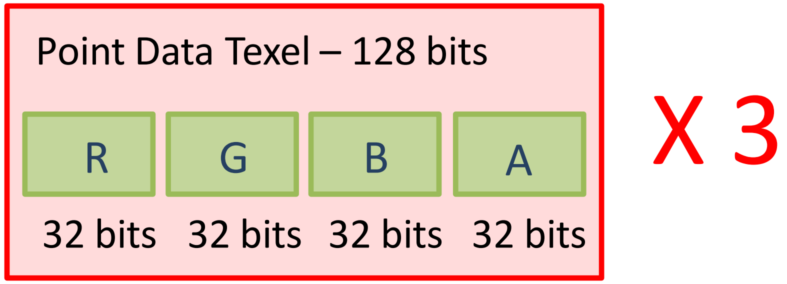


- Update *p*

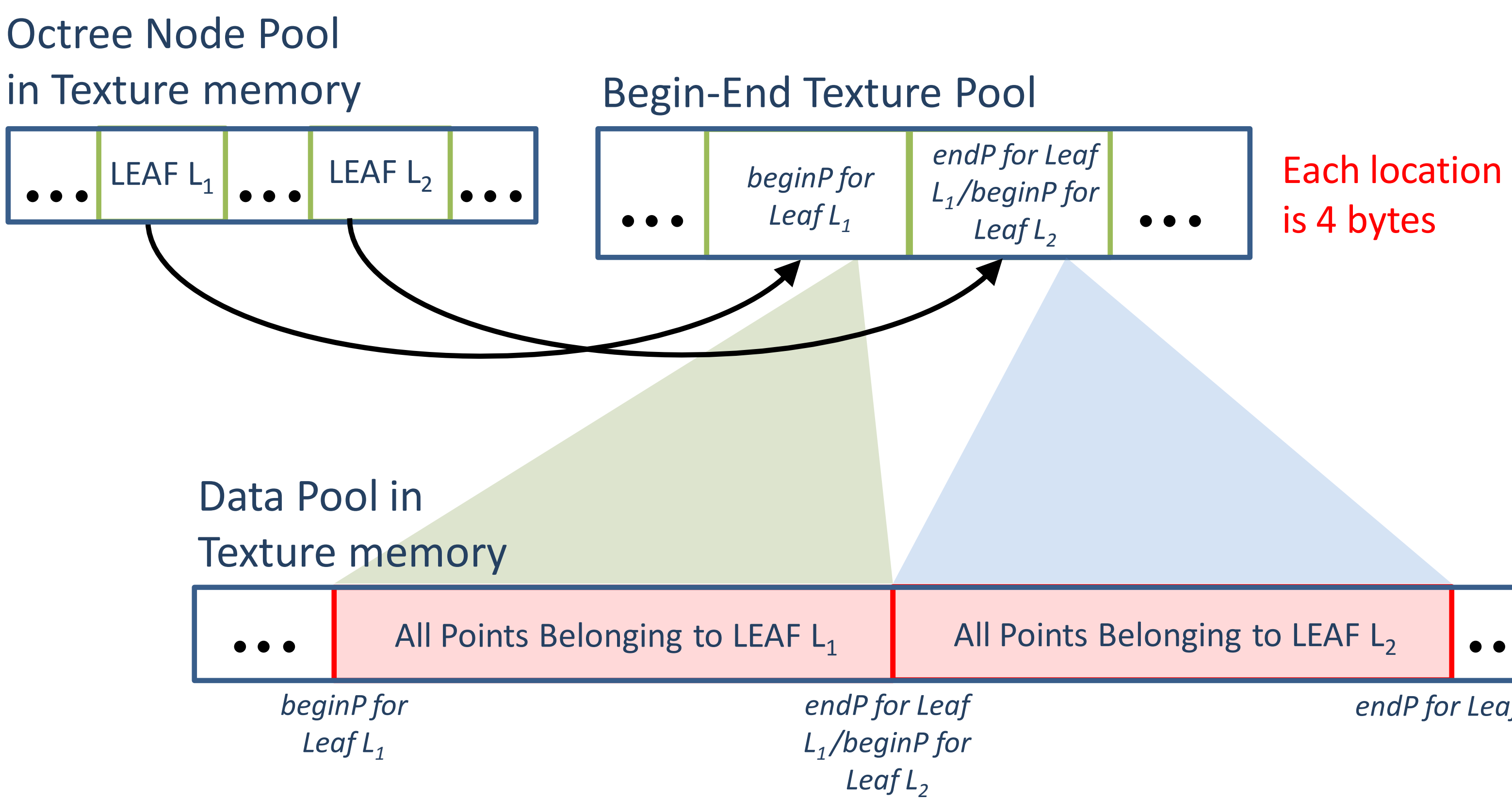
$$p = p * N - \text{int}(p * N)$$

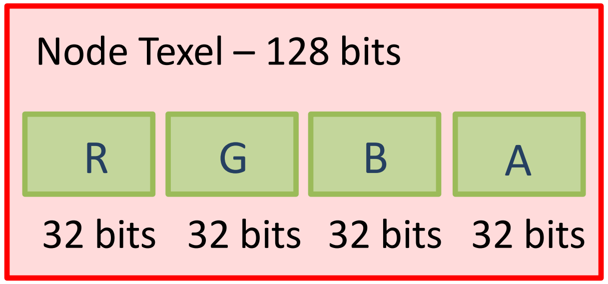
Ray-trace Rendering: Point Data Texture on GPU

- Every point has following attributes:
  - Co-ordinates (3 floats)
  - Normal (3 floats)
  - Color (3 floats)
  - Radius (1 float)
  - Material ID (1 float)
- Total of 44 bytes per point
- 3 Texels, each of 16 bytes

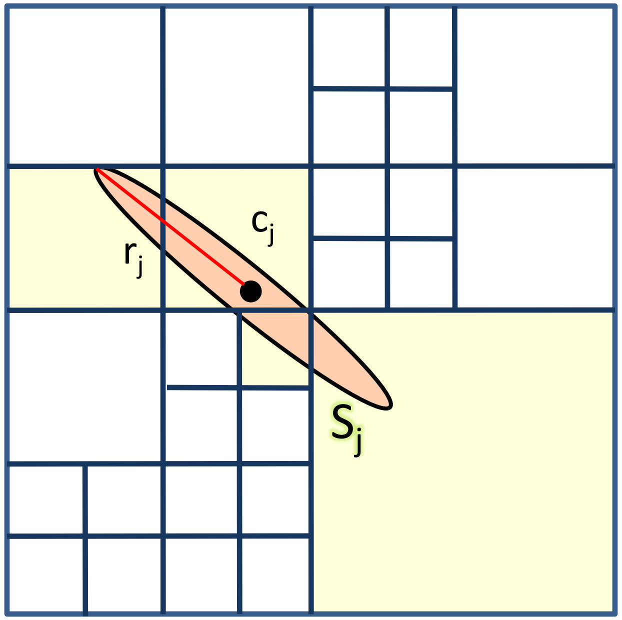


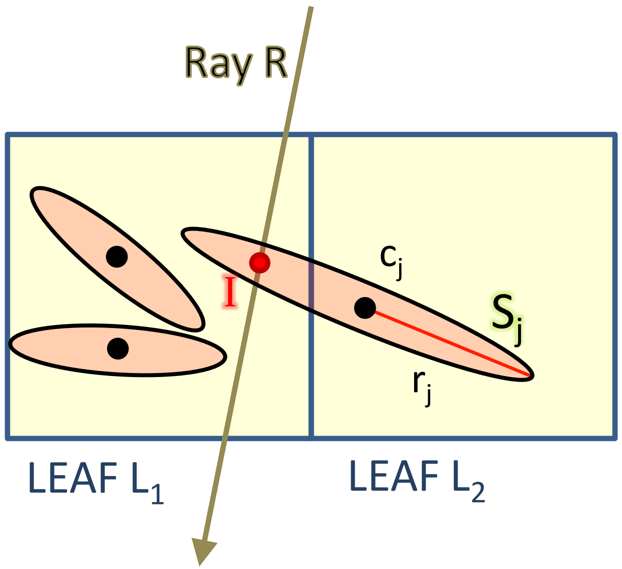
Ray-trace Rendering: Point Data Texture on GPU



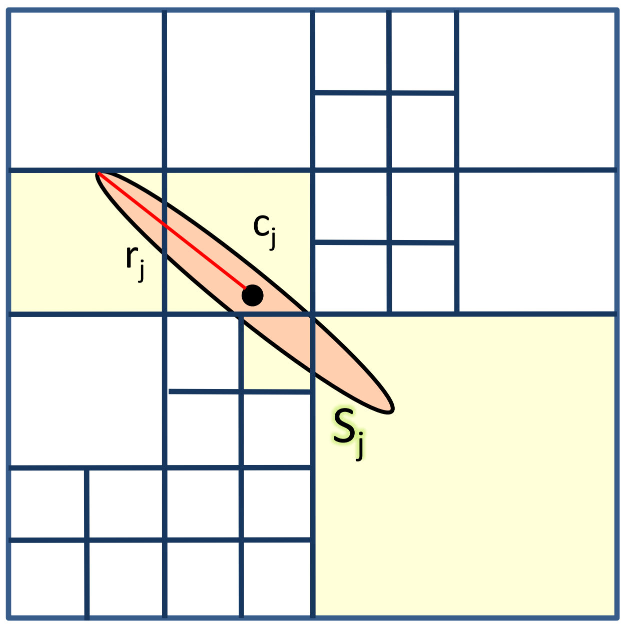


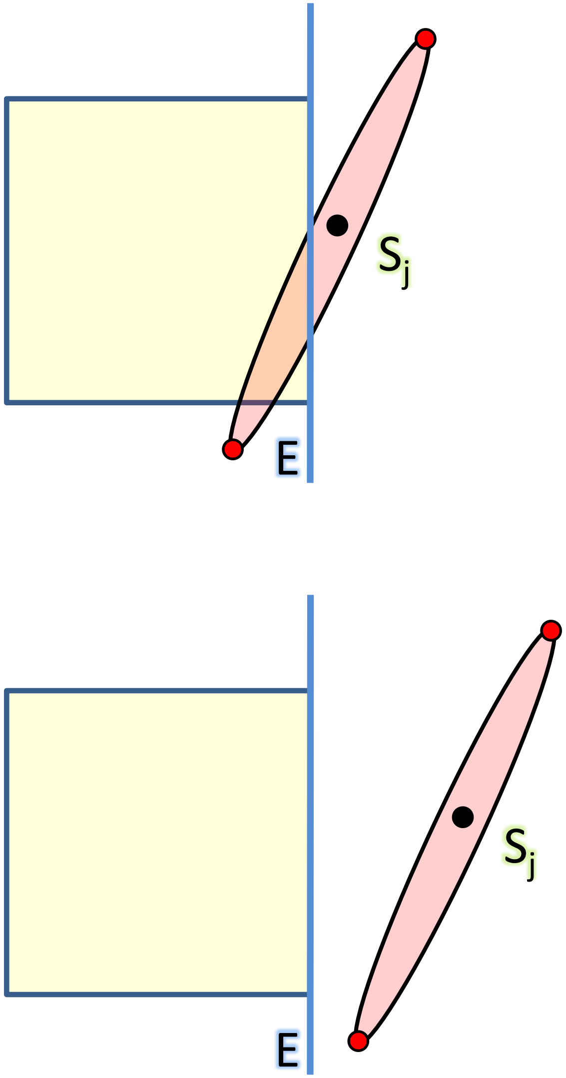
Ray-trace Rendering: Multiple Leaves Per Splat





Ray-trace Rendering: Multiple Leaves Per Splat





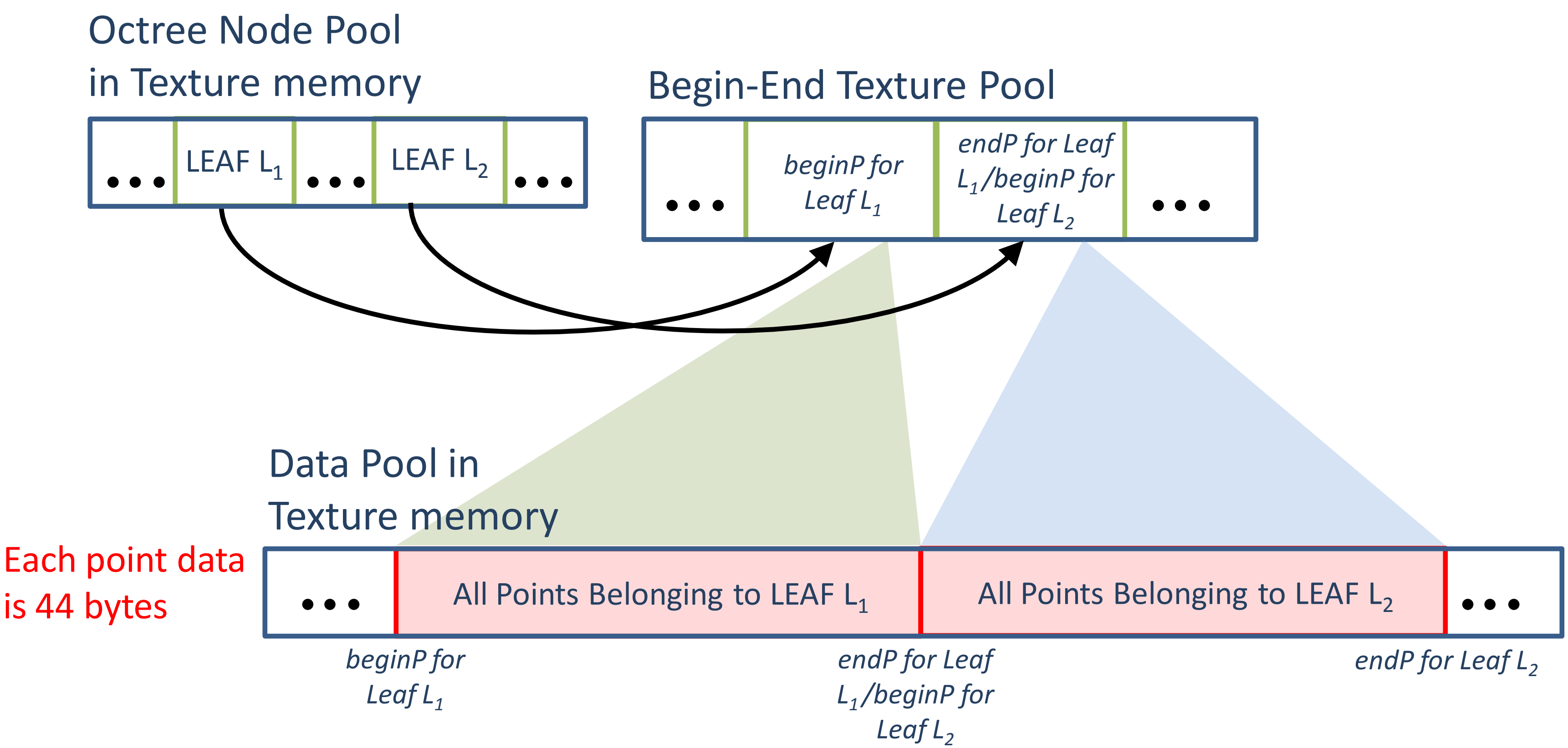
- Traverse top-down. Check if axis-aligned bounding box of splat intersects the node
- If leaf, perform a parameterized intersection test, using splat-aligned square bounding box

Ray-trace Rendering: Multiple Leaves Per Splat

- Adding splats to every intersecting leaf means adding 44 bytes of data for every *Extra Splat (ES)*
- Store only the address (4 bytes) of the *ES* !

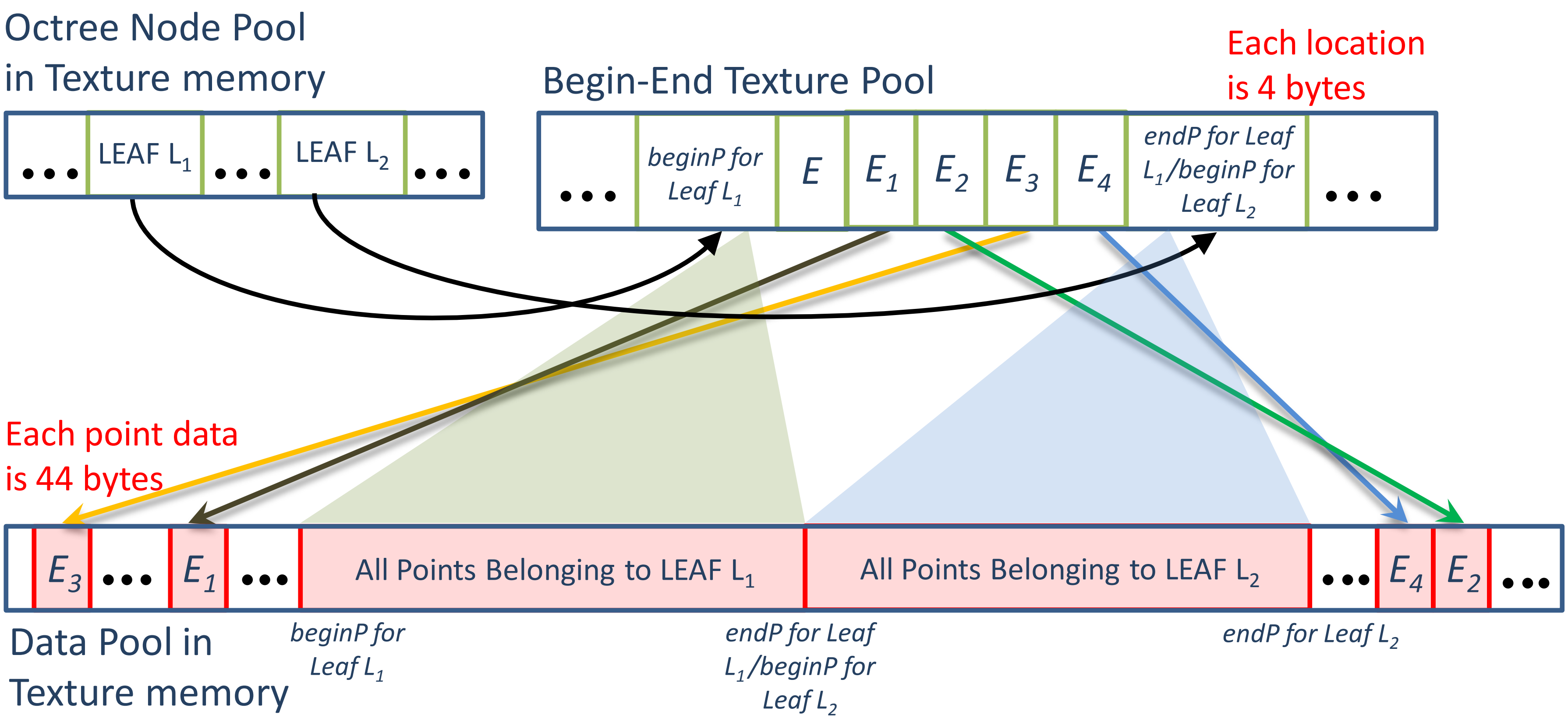
Ray-trace Rendering: Multiple Leaves Per Splat

- Adding splats to every intersecting leaf means adding 44 bytes of data for every *Extra Splat (ES)*
- Store only the address (4 bytes) of the *ES* !



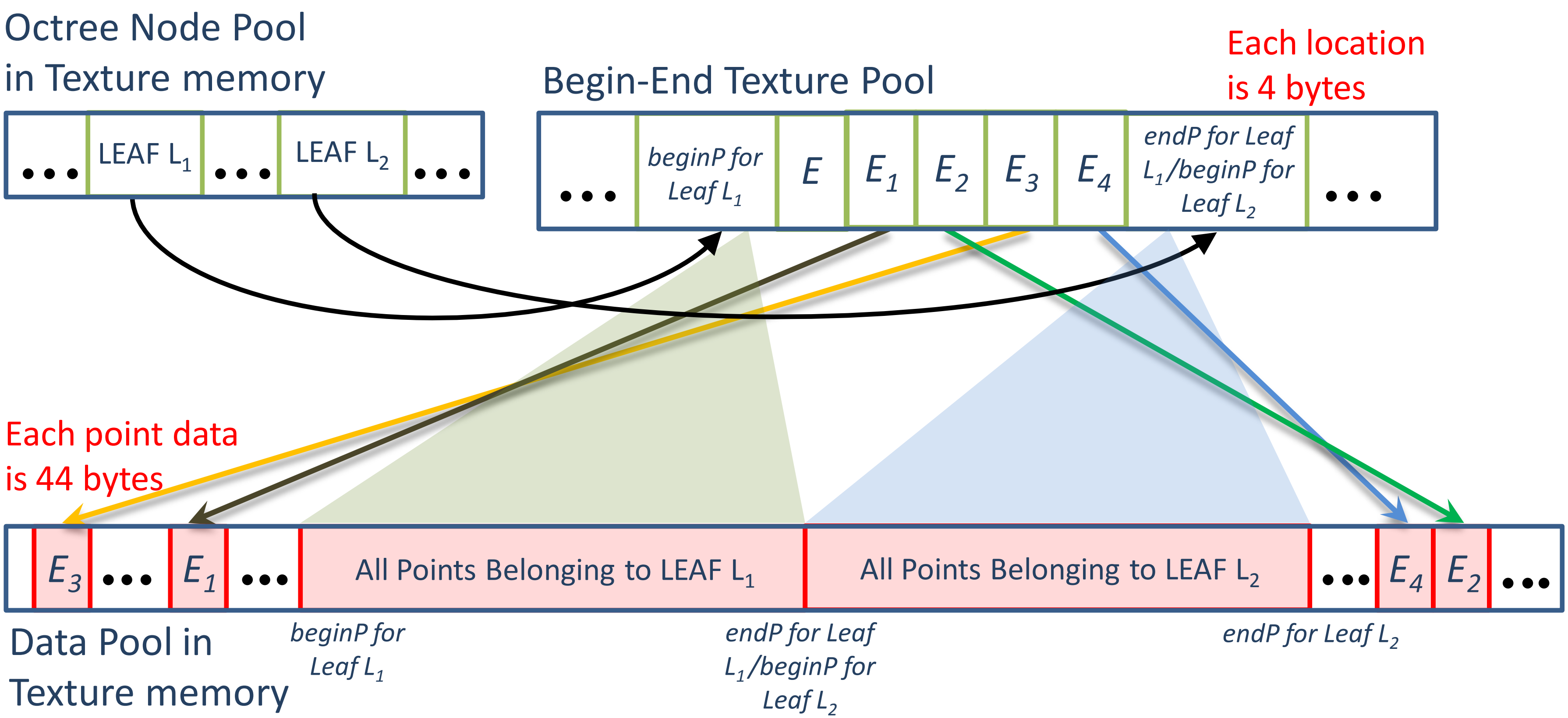
Ray-trace Rendering: Multiple Leaves Per Splat

- Make some space for *Extra Splats (ES)*

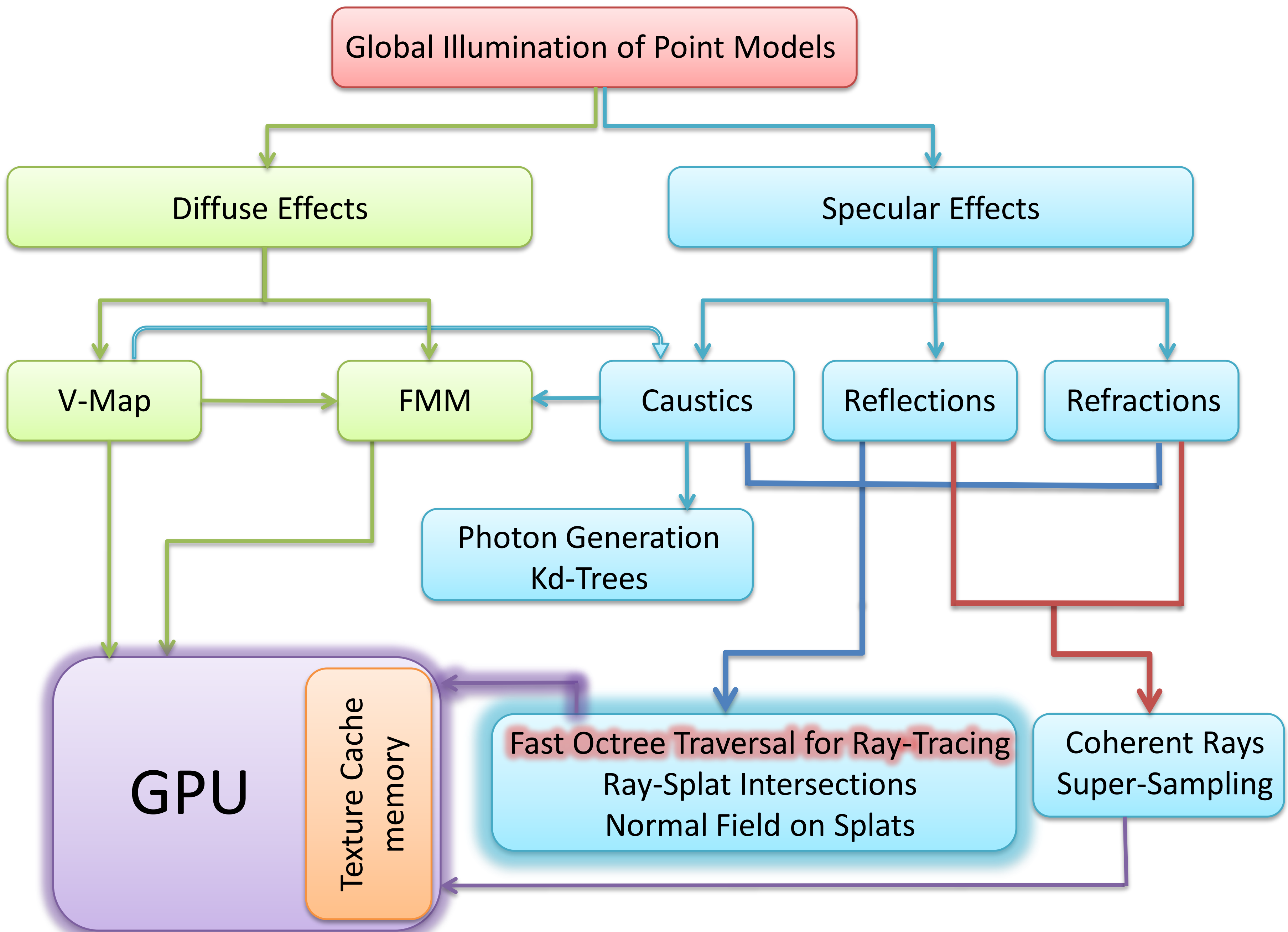


Ray-trace Rendering: Multiple Leaves Per Splat

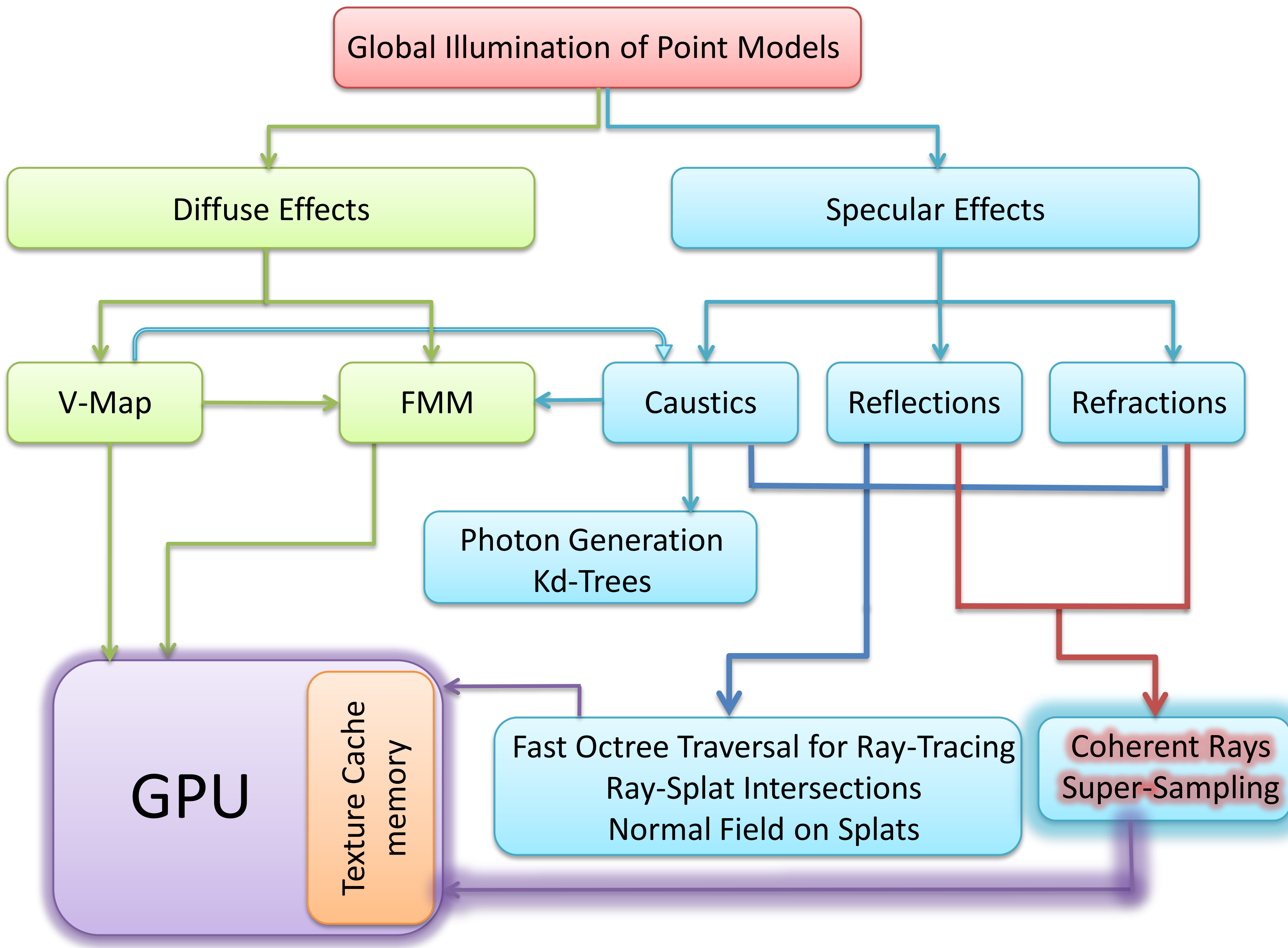
- Make some space for *Extra Splats (ES)*



- Pre-process step



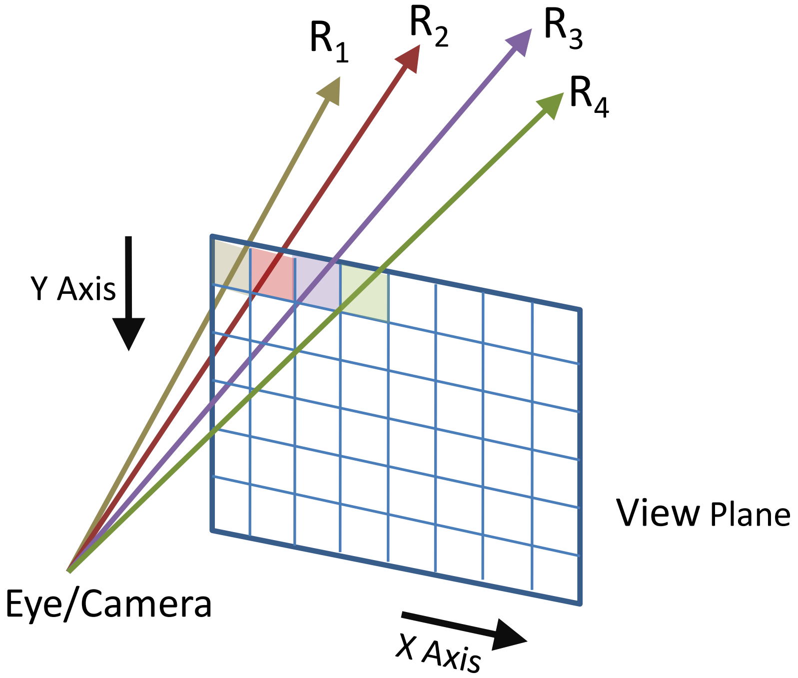
Global Illumination of Point Models
Diffuse Effects
Specular Effects
V-Map
FMM
Caustics
Reflections
Refractions
Photon Generation
Kd-Trees
GPU
Texture Cache
memory
Fast Octree Traversal for Ray-Tracing
Ray-Splat Intersections
Normal Field on Splats
Coherent Rays
Super-Sampling

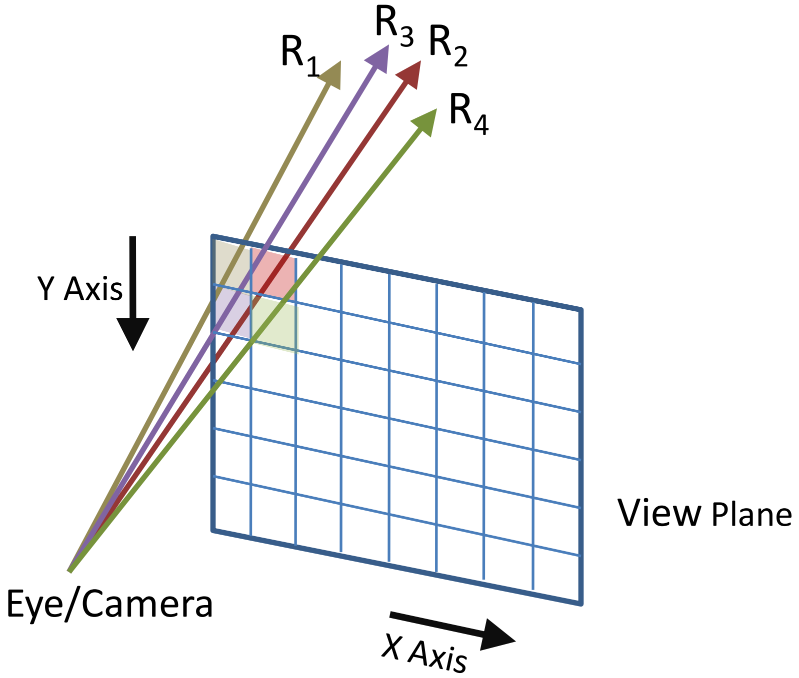


Global Illumination of Point Models
Diffuse Effects
Specular Effects
V-Map
FMM
Caustics
Reflections
Refractions
Photon Generation
Kd-Trees
GPU
Texture Cache memory
Fast Octree Traversal for Ray-Tracing
Ray-Splat Intersections
Normal Field on Splats
Coherent Rays
Super-Sampling

Ray-tracing: Coherent Rays & Super-Sampling

- Important to get *texture cache-hits* !
- Use Ray-Coherency and Super-Sampling *(8x4)*
- Gives well-behaved warp of 32 threads
- Also reduces *aliasing*





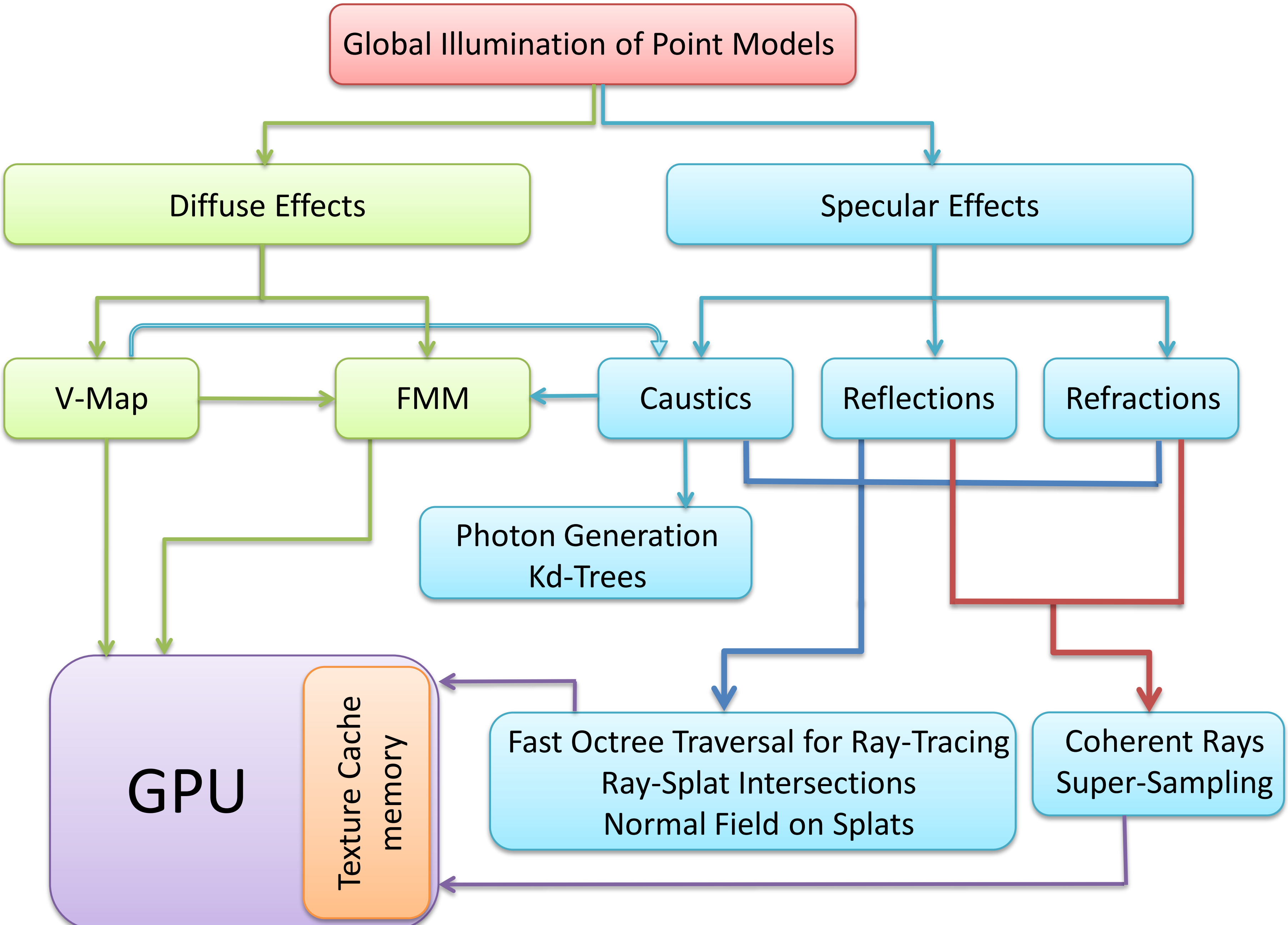
Fusing Together: Diffuse and Specular Effects

- FMM takes care of LD^+ path
- Ignores contributions from specular splats (LS^+D)
- Does not transfer energy to specular splats (LD^*S^+)
- LS^+D and LDS^+ handled by caustics and ray-tracing

Fusing Together: Diffuse and Specular Effects

- FMM takes care of *LD^+* path
- Ignores contributions from specular splats (*LS^+D*)
- Does not transfer energy to specular splats (*LD^*S^+*)
- ***LS^+D* and *LDS^+* handled by caustics and ray-tracing**

- This gives us the whole set-up of a complete global illumination package for point models!


Global Illumination of Point Models
Diffuse Effects
Specular Effects
V-Map
FMM
Caustics
Reflections
Refractions
Photon Generation
Kd-Trees
GPU
Texture Cache
memory
Fast Octree Traversal for Ray-Tracing
Ray-Splat Intersections
Normal Field on Splats
Coherent Rays
Super-Sampling

Plan

- Introduction – *Problem Definition*
- Diffuse Effects – *Overview*
- Specular Effects – *Details*
- Results
- Wrap-Up

Results

Work In Progress!

Plan

- Introduction – *Problem Definition*
- Diffuse Effects – *Overview*
- Specular Effects – *Details*
- Results
- Wrap-Up

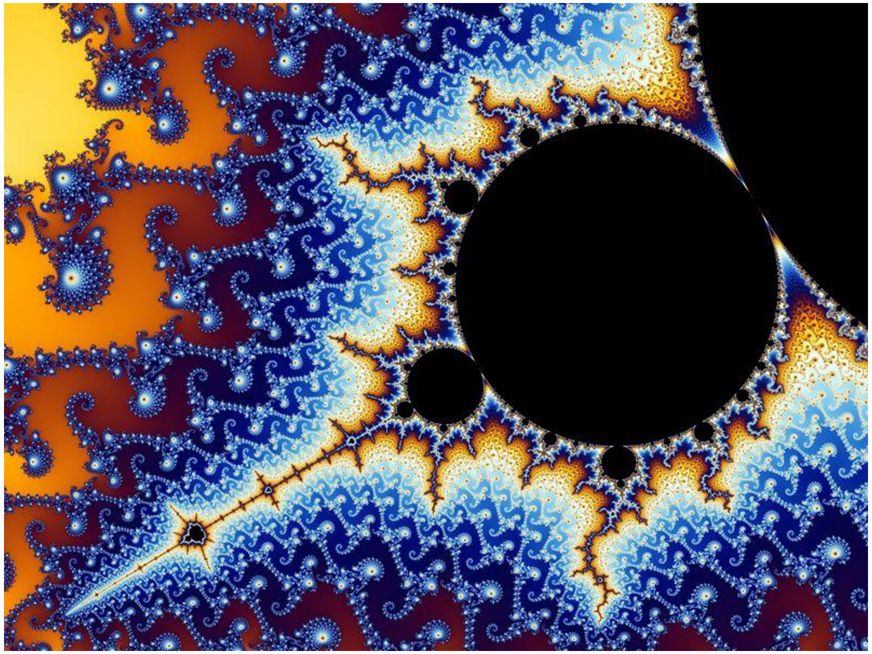
Final Notes

- Lack of connectivity makes GI on point models, difficult
- Point-to-Point visibility arguably one of the more difficult problem in rendering
- V-Map helps in resolving visibility between clusters at group level
- FMM uses V-Map for diffuse irradiance transfers
- Parallel versions of V-Map construction and FMM implemented using CUDA on GPU
- Achieves upto 20x speed-ups
- Further, V-Maps were used for tracing initial paths of caustic photons, saving time !
- Factors like Number of photons to be sent, where to send were in sync with emissive power of light source and areas occupied by the splats as seen from the light source
- Normal field generated for each specular splat gave accurate secondary ray direction

Final Notes

- We used an efficient yet naive octree traversal algorithm (on GPU) to do photon tracing
- Texture cache on GPU used for efficient octree storage and fast traversal
- Same octree traversal algorithm used for ray-trace rendering as well
- Kd-tree was used to organize the caustic photon map, enabling fast retrieval
- Point data organized into texture accessible by leaves of the octree
- Provision for multiple leaves per splat was added to avoid undesirable holes & artifacts
- Coherent rays and super-sampling was performed to aid a well behaved warp on GPU and avoid aliasing artifacts
- Diffuse and Specular effects are thus achieved in an unified global illumination package for point models

Thank You



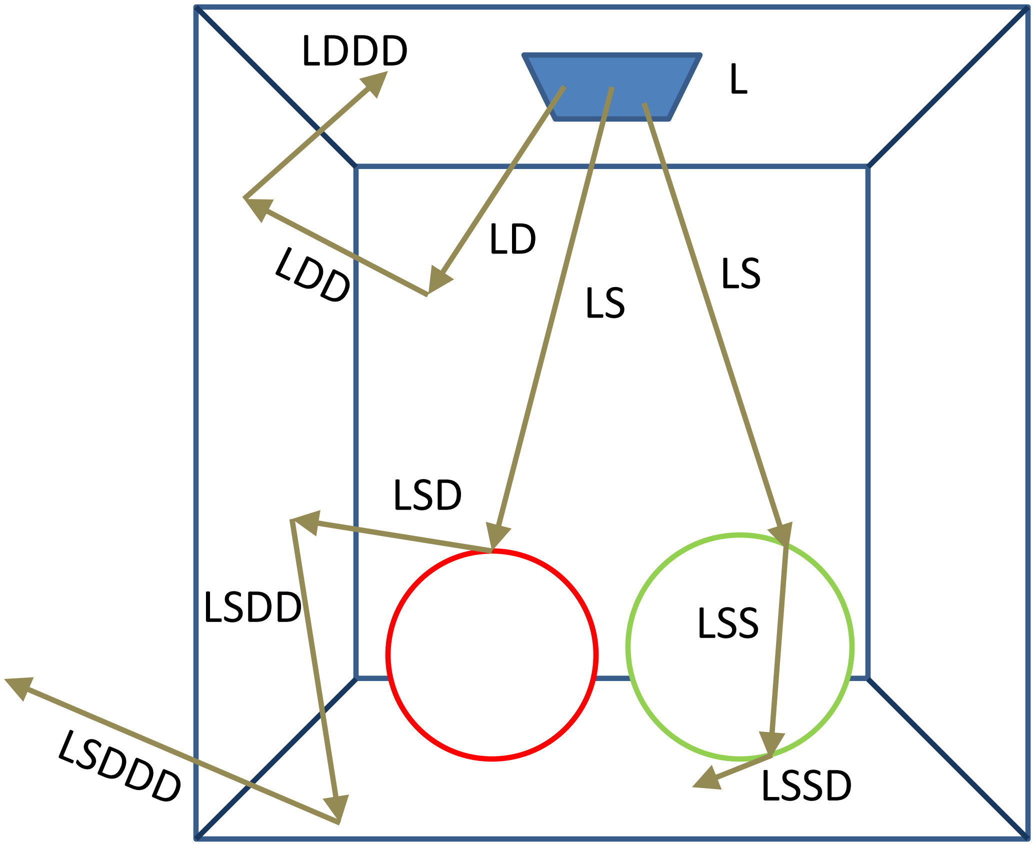
Fractal: Mandel Zoom - Satellite Antenna, Mandelbrot set

Photon Mapping

- A Two-pass GI method

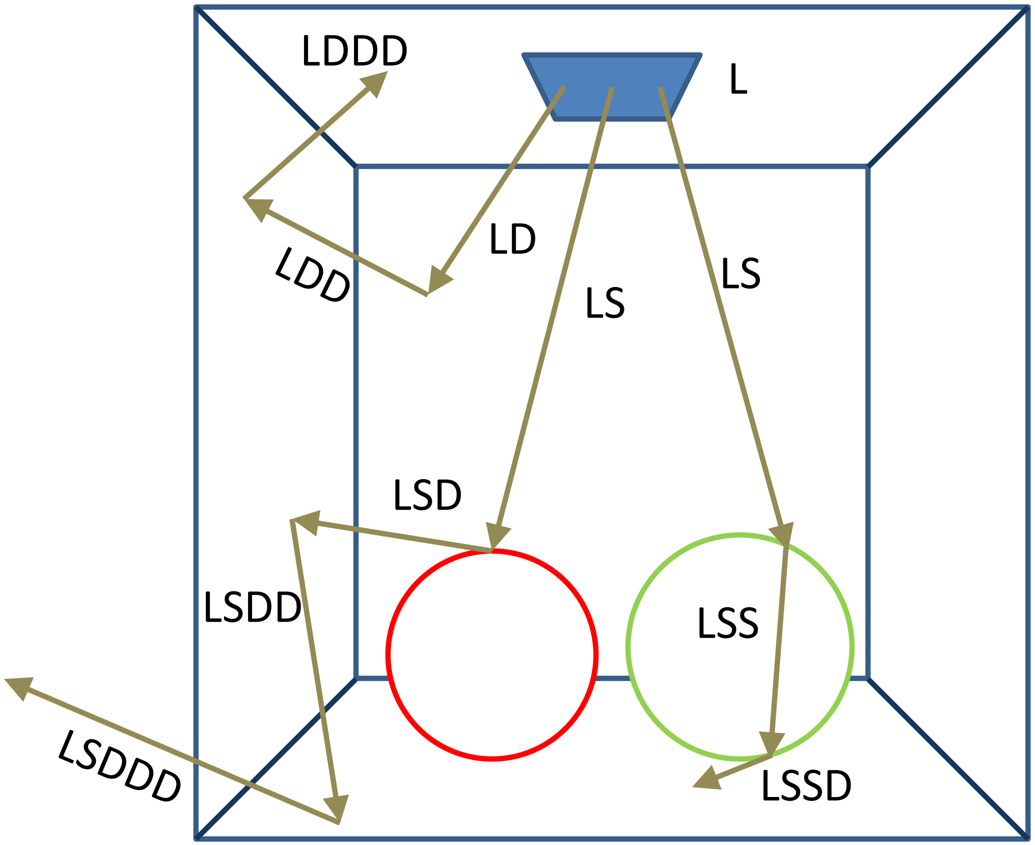
Photon Mapping

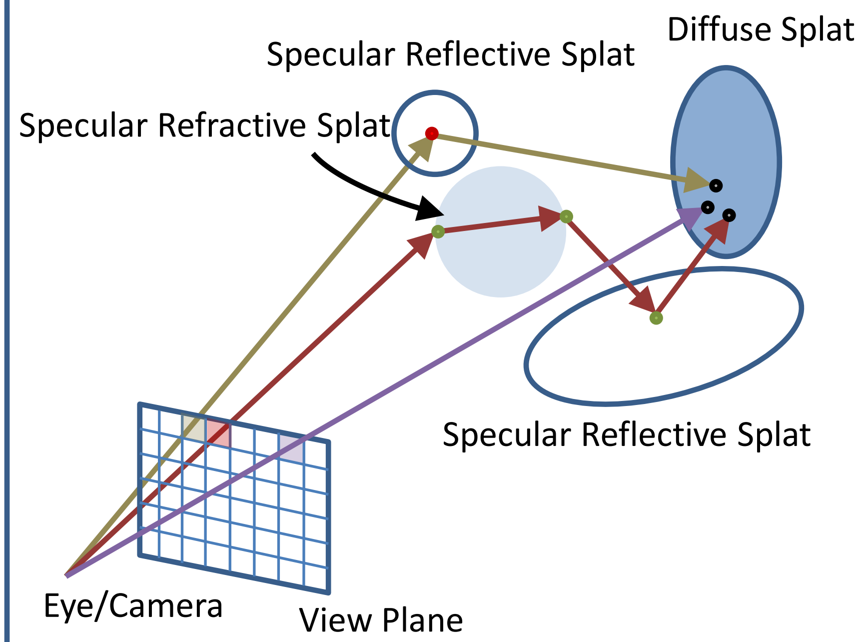
- A Two-pass GI method
  - First pass builds the photon map (Follow Heckbert's notation)
    - Emit photons from light sources into the scene
    - Store them in a photon map on hitting diffuse objects



Photon Mapping

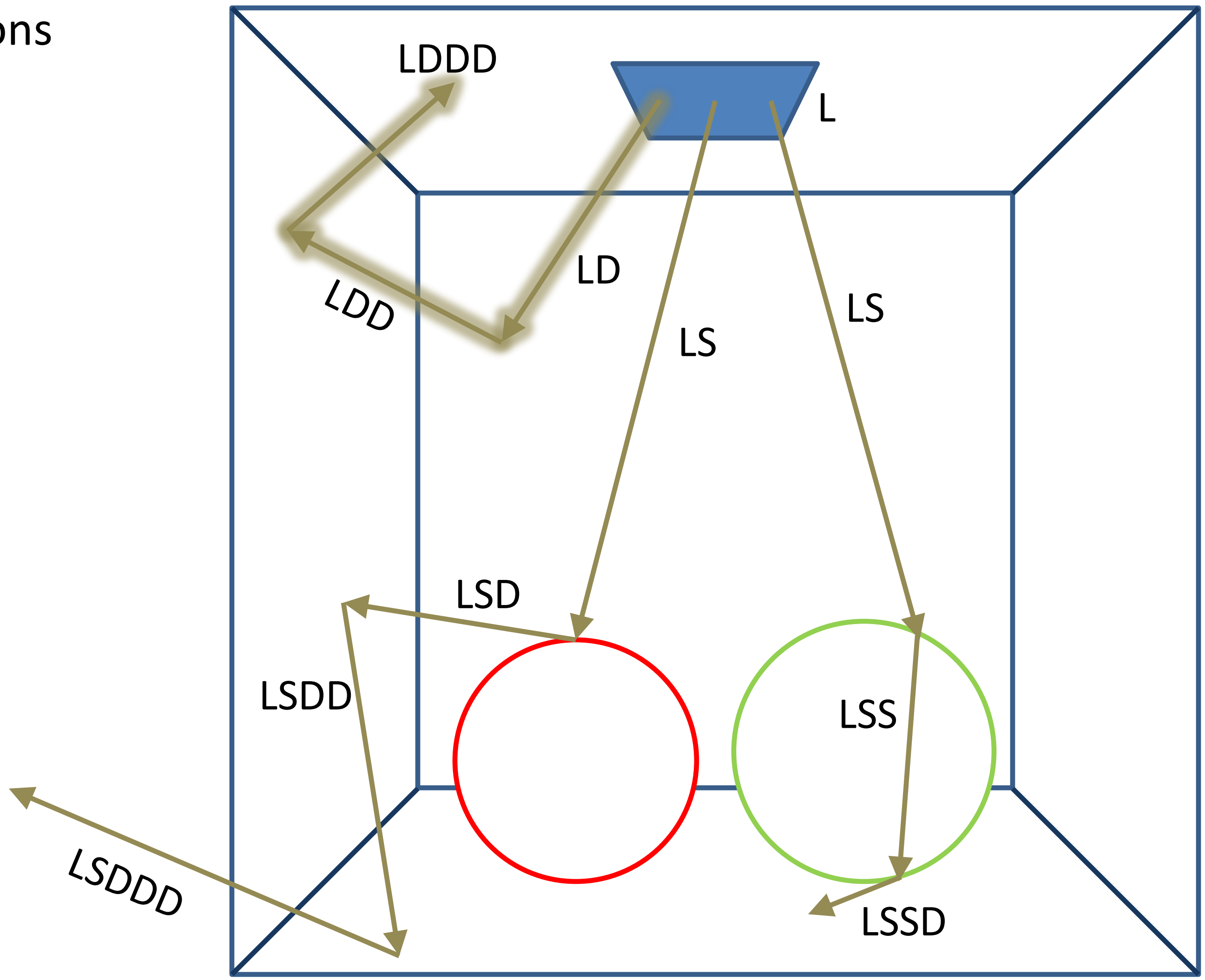
- A Two-pass GI method
  - Second pass, the ray-trace rendering pass
    - Make *kNN* queries on the photon map
    - Extract information about the radiance values





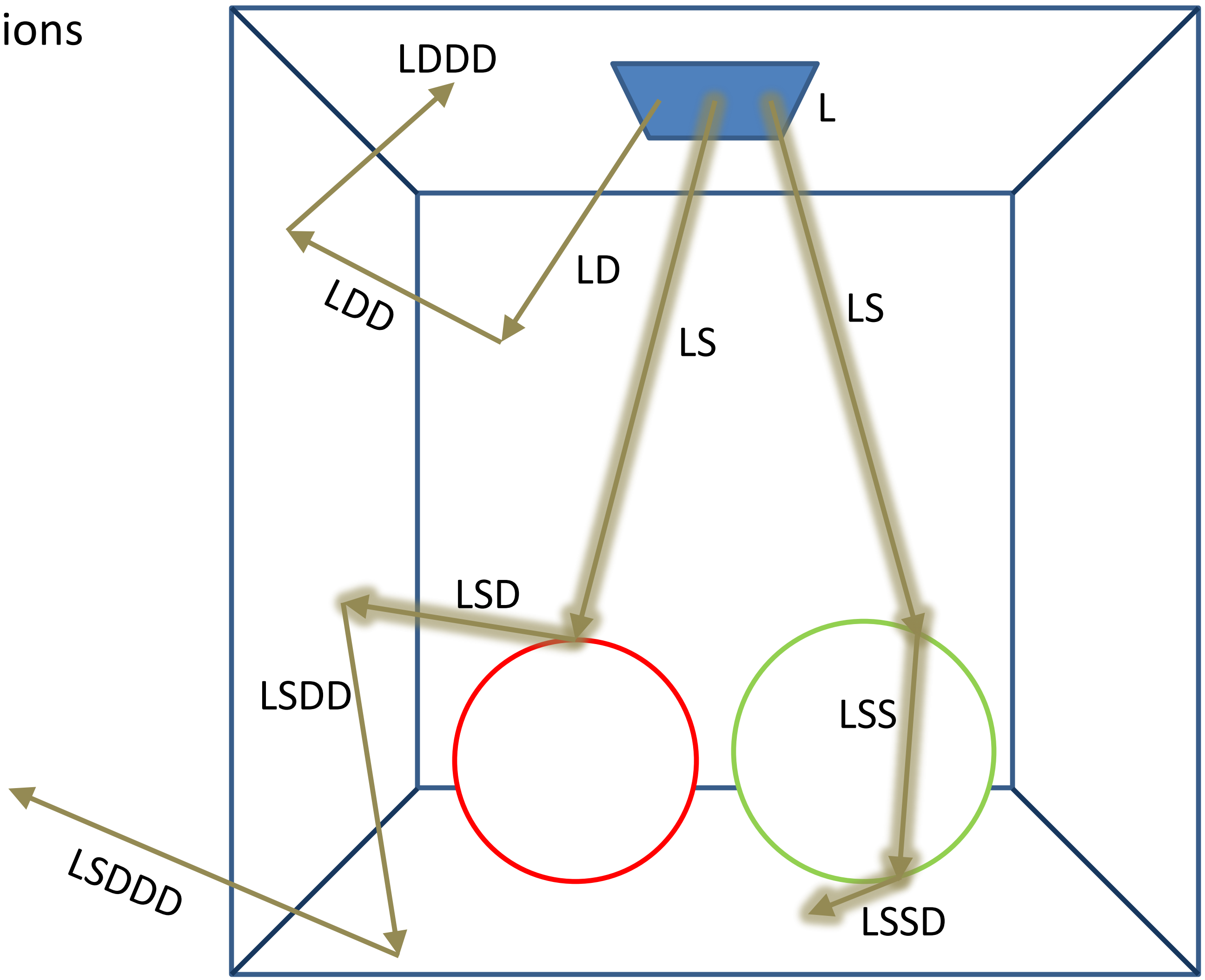
Paths of Photons

- Diffuse Interactions
- LD* path
- Solution: FMM



Paths of Photons

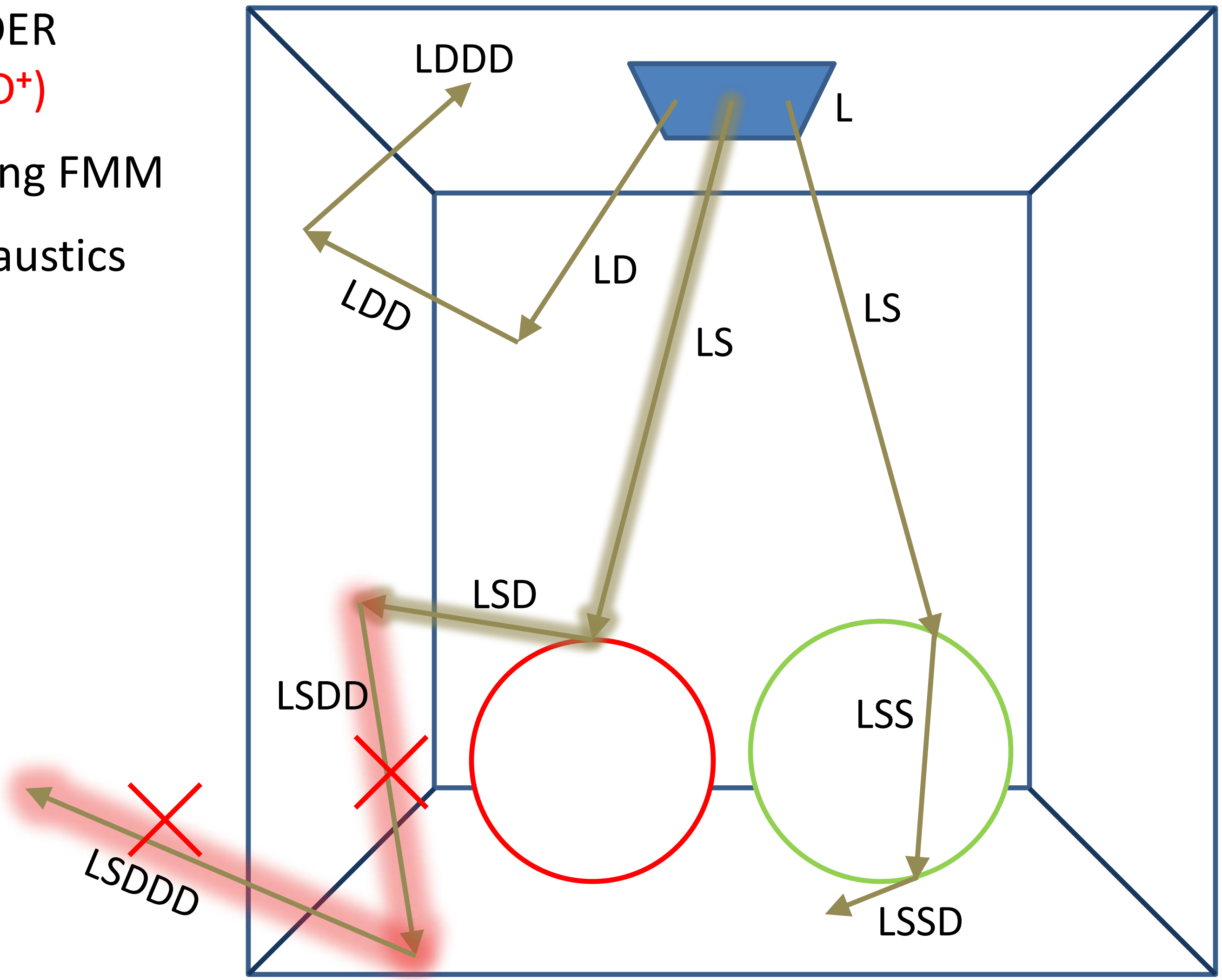
- Specular Interactions
- LS^+D path
- Result: Caustics



Rhushabh Goradia, Sharat Chandran

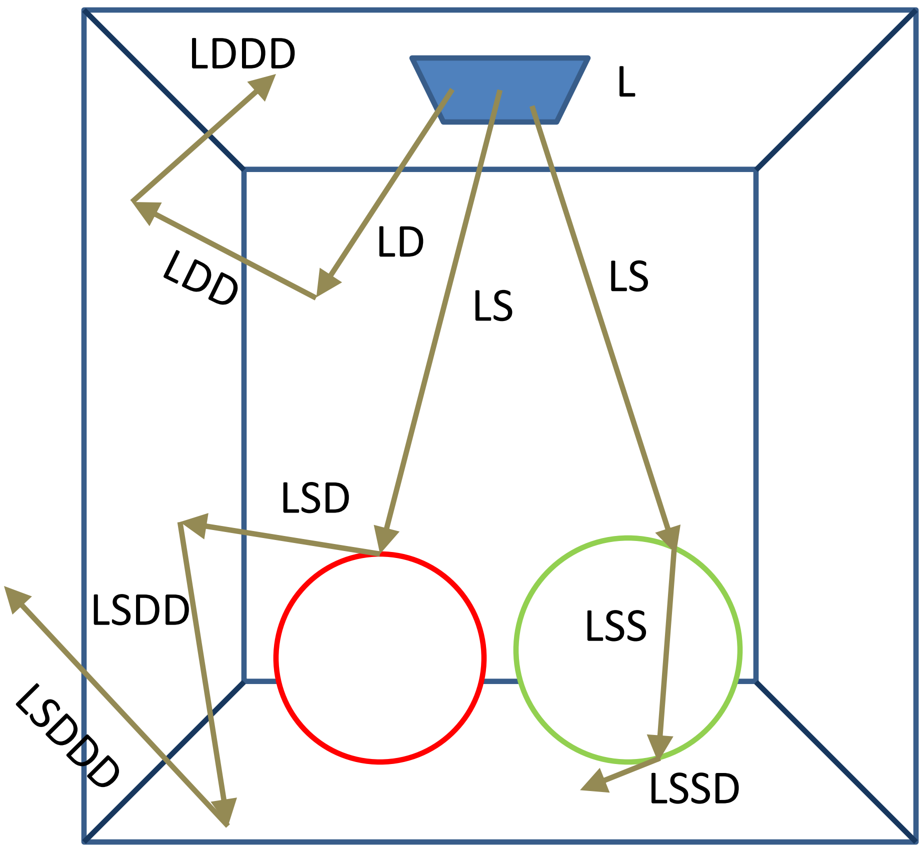
Paths of Photons: Our Approach

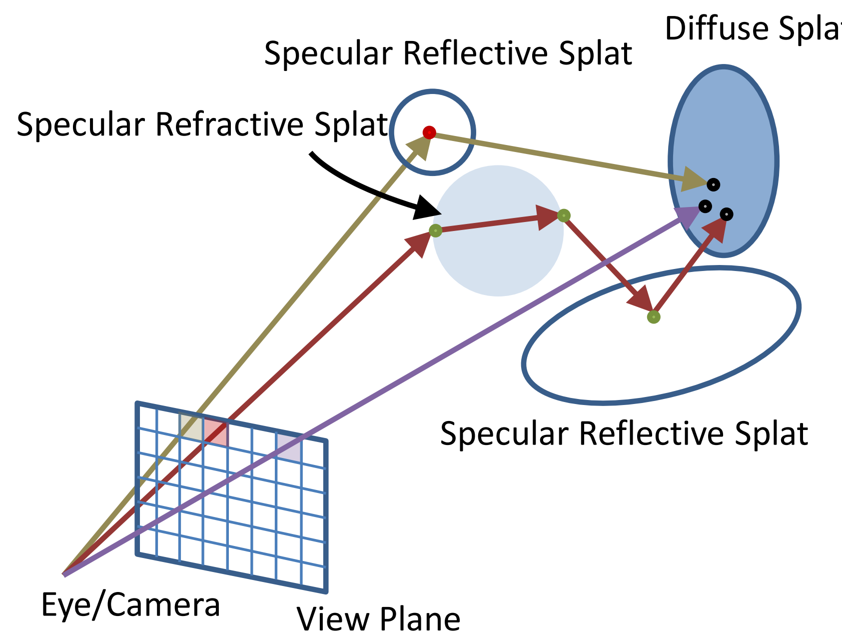
- DO NOT CONSIDER
  RED PATHS (LS^+D^+)
- Considered during FMM
- ONLY LS^+D for Caustics



Rhushabh Goradia, Sharat Chandran

View-Independence v/s View-Dependence





- EDD/ED^{+}
- ESD/ES^{+}D
- EDS/ED^{+}S^{+}D
- ESS

Tracing Caustic Photons