Model EventType Ledger Design

Model EventType Ledger Design

The ledger will be composed of two hash maps (key-value stores) that allows us to traverser the unique IDs (UID) of each event in each generation sequence.

The photons collected are received in packets composed of:

```
struct Photon {
    tof:f64,
    power: f64,
    wavelength:f64,
    pos: Pos3,
    dir: Dir3,
    uid: UID
}
struct UID {
    seq_no: u32,
    type: EventType
}
```

The following specification focuses mostly on how to encode the `EventType`, but also gives a usage example below.

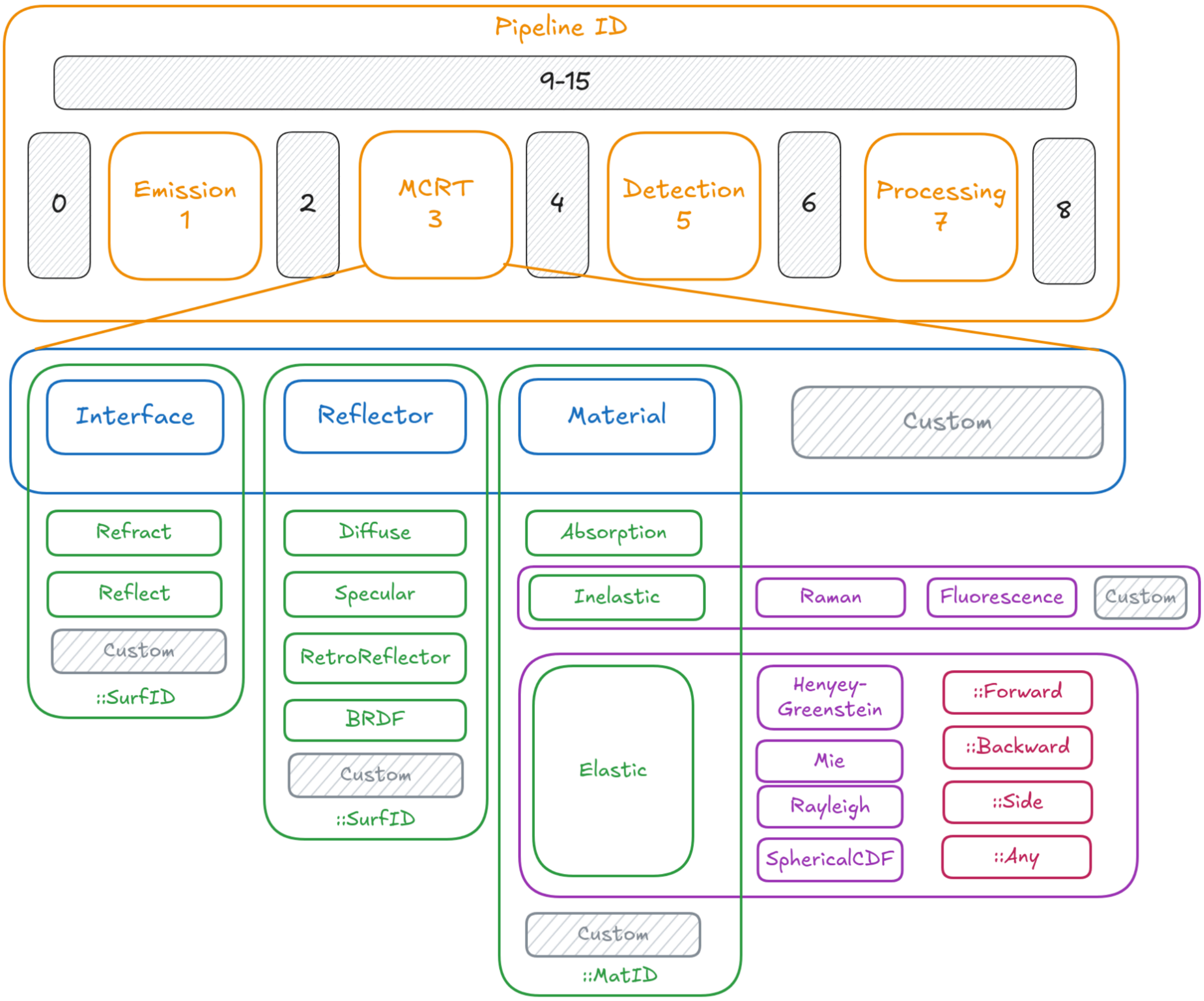

Fig. 1: Simulation Event Types Encoding Space Segmentation

Encoding Scheme

Encoding Scheme

31 28	27 24	23 22	21 16	15 0	
Reserved	Pipeline	EventType		SrcId	Generic
Reserved	0001	EventType		LightId	Emission Event
Reserved	0011	SuperType	SubType	MatSurfId	MCRT Event

MCRT Events

_	0011 (MCRT)	00	SubType	MatSurfId	Interface
_	MCRT	01	SubType	MatSurfId	Reflector
_	MCRT	10	SubType	MatSurfId	Material
_	MCRT	11	SubType	MatSurfId	Custom

Interface Events

_	MCRT	Interface	000000	SurfId	Reflection
_	MCRT	Interface	000001	SurfId	Refraction
_	MCRT	Interface	1xxxxx	SurfId	Custom

Reflector Events

_	MCRT	Reflector	00001x	SurfId	Diffuse
_	MCRT	Reflector	00010x	SurfId	Specular
_	MCRT	Reflector	00011x	SurfId	Composite
_	MCRT	Reflector	001000	SurfId	RetroReflective
_	MCRT	Reflector	001001	SurfId	CompRetroRef
_	MCRT	Reflector	001001	SurfId	CompRetroRef

Material Events

31 28	27 24	23 22	21 20	19 18	17 16	15 0
Reserved	Pipeline	Material	SubType			MatId
Reserved	Pipeline	Material	Interaction	Extra		MatId
R ```eserved	Pipeline	Material	Interaction	ScatterType	Direction	MatId

_	MCRT	Material	00	xxxxxx		MatId	Absorption
_	MCRT	Material	01	ScatterType	Direction	MatId	Inelastic
_	MCRT	Material	10	ScatterType	Direction	MatId	Elastic
_	MCRT	Material	11	xxxxxx		MatId	Custom

Inelastic Scattering Events

_	MCRT	Material	Inelastic	00	Direction	MatId	Raman
_	MCRT	Material	Inelastic	01	Direction	MatId	Fluorescence
_	MCRT	Material	Inelastic	1x	Direction	MatId	Custom

Elastic Scattering Events

_	MCRT	Material	Elastic	00	Direction	MatId	Henyey-Greenstein
_	MCRT	Material	Elastic	01	Direction	MatId	Mie
_	MCRT	Material	Elastic	10	Direction	MatId	Rayleigh
_	MCRT	Material	Elastic	11	Direction	MatId	SphericalCDF

Direction Description

_	MCRT	Material	Elastic	ScatterType	00	MatId	Any
_	MCRT	Material	Elastic	ScatterType	01	MatId	Forward
_	MCRT	Material	Elastic	ScatterType	10	MatId	Side
_	MCRT	Material	Elastic	ScatterType	11	MatId	Backward

Ledger Show-case

UID { seq_no, type}	next(seq_no)	Description/Ptr to struct definition
{1, Laser}	2	Laser emission
{1, Background }	3	Background emission
{2, FS{Mat{Air}}}	4	Forward scatter with air
{2, BS{Mat{Air}}}	5	Background scatter with air
{2, Reflection{Mat{TransPLA}}}	6	Reflection from Fresnel refraction with target
{2, Refraction{Mat{TransPLA}}}	7	Refraction from Fresnel refraction with target
{7, FS{Mat{TransPLA}}}	8	Forward scatter target
{8, FS{Mat{TransPLA}}}	9	Forward scatter target
{9, BS{Mat{TransPLA}}}	10	Forward scatter target
{10, Refraction{Mat{Air}}}	11	Refraction from Fresnel refraction back to air
{11, Detection{PhotonCollector}}	12	Detected at apperture of SPAD sensor from SSS
{6, Detection{PhotonCollector}}	13	SPAD Detection of balistic

UID { seq_no, type}	next(seq_no)	Description/Ptr to struct definition
seq_no	UID	
---	---	
1	{0, Root}	
2	{1,Laser}	
3	{2, Background}	
4	{2, FS{Mat{Air}}}	
6	{2, Reflection{Mat{TransPLA}}}	
7	{2, Refraction{Mat{TransPLA}}}	
8	{7, FS{Mat{TransPLA}}}	
9	{8, FS{Mat{TransPLA}}}	
10	{9, BS{Mat{TransPLA}}}	
11	{10, Refraction{Mat{Air}}}	
12	{11, Detection{PhotonCollector}}	

Filter example

`photon.filter_deny(type:Background).filter_deny(type:SSS{TranslucentPLA}).filter_allow(type:Reflection/Refraction{Mat{TranslucentPLA})`

`photon.filter_deny(type:Background).filter_allow(type:SSS{TranslucentPLA})`

UID & EventType values encoding

```
abstract type AbstractEvent end
abstract type SubTypeAbstract <: UInt6;
abstract type MatSurfId <: UInt16;
struct MatId = MatSurfId;
struct SurfId = MatSurfId;

type SurfId <: MatSurfId

@enum SuperType <: UInt2
    Interface = 0,
    Reflector = 1,
    Material = 2,
    Custom = 3,
end

@enum Pipeline <: UInt4
    # Custom = 0
    Emisssion = 1,
    # Custom = 2
    MCRT = 3,
    # Custom = 4
    Detection = 5,
    # Custom = 6
    Processing = 7,
    # Custom = 8-15
end

@enum InterfaceEvent <: SubTypeAbstract
    Reflect = 0,
    Refract = 1,
end

@enum ReflectEvent <: SubTypeAbstract
    Diffuse = 0
    Specular = 1,
```

```
    RetroReflector = 2,
    BRDF = 3,
end

@enum MaterialEvent <: SubTypeAbstract
    interaction_type::UInt2,
    scatter_type::UInt2,
    direction::UInt2,
end

struct EventType {
    super_type::SuperType,
    sub_type::SubTypeAbstract,
}

struct MCEvent <: AbstractEvent {
    msb_inter_id::UInt4, # u4
    pipeline::Pipeline,  # u4
    event_type::EventType, # u8
    inter_id::MatSurfID, # u16
} # u32

struct UID {
    seq_no::UInt32,    # u32
    event_type::Event # u32
} # u64
```

Pipeline encoding: 4-bits

The `pipeline` enum is defined as $r_3 b_2 b_1 r_0$, where $r_3 = 0$ and $r_0 = 1$ are reserved bits that can be changed on a follow up specification an allowing left and right padding of the stages enumerate, such that further functionality can be interleaved in the modelling pipeline. From the PoV of each model the `PipelinedSuperType` is each SuperType enumeration and shouldn't care about the details apart from setting these bits correctly.

C	Emission	C	MCRT	C	Detection	C	Processing	C	Unordered C
0	1	2	3	4	5	6	7	8	9-15

C = Custom

SuperType events: 4-bits

> [NOTE] From here on we are only talking about types referring to the MCRT/Aetherus events

Aetherus Event SuperTypes:

- Interface
- Reflector `Mirror <: Reflector`
- Material

These are just given values in the order described as an enum:

```
enum SuperType : uint4 {
    Interface,
    Reflector,
    Material,
    // CustomCodes
}
```

SubTypes events: 8-bits

Now looking under the hierarchy of each events described by the SuperType above, we can build the hierarchy as follows:

- Interface
    - Reflect
    - Refract

- Reflector
  - Diffuse
  - Specular `<: Mirror`
  - RetroReflector
  - BRDF (Bi-directional Reflection Distribution Function)
- Material
  - Raman
  - Fluorescence
  - Scatter
    - Heyney-Greenstein | Mie | Rayleigh | SphericalCDF
      - ForwardScatter
      - SideScatter
      - BackwardScatter
      - Any

Material SubType

Material events have a particular labelling for scattered photons, that describe the scattering model used, but also the direction the photons are scattered.
All volume/material interaction are scattering or absorption, however only scattering keeps propagating as events, but maybe we should keep track of absorption as well

SubType enum	Material Interaction	Scatter Type	Direction
7 - 6	5 - 4	3 - 2	1 - 0
Material	Absorption	0	0
Material	InelasticScatter	Raman	NA
Material	InelasticScatter	Fluorescence	NA
Material	ElasticScatter	HG	Any
Material	ElasticScatter	HG	Forward
Material	ElasticScatter	HG	Side
Material	ElasticScatter	HG	Backward
Material	ElasticScatter	Mie	...
Material	ElasticScatter	Rayleigh	...
Material	ElasticScatter	SphericalCDF	...

Material/Surface ID: 16-bits

Each surface and material are described by an ID. The ID don't have to be unique, i.e. multiple surfaces or objects can map to the same ID.

Then these scene is described by 2 HashMaps `Surface → MatSurfID` and `Material → MatSurfID`.

These IDs are decided on at runtime based on the scene that is composed, but we can restrict the values such that are useful for downstream processing.
The rules described below are loose, but are desirable to be implement in order to be easier to discern objects in the scene.

I) Use the same ID for Material and Surface of the same object, hence, there will be a single ID to query for in the `MatSurfID` if photons interacted with certain object at all.
II) Group together multiple objects to map to the same ID if it's not necessary to separate them, as it will make filtering easier. Then the Surface and Material HashMaps will have multiple entries mapping to the same `MatSurfID`. This could be of interest for multiple objects that compose the far-field objects that are not interest in the scene.