

Sviluppo di plugin per l'import e preview di fixture virtuali basate su GDTF in Unreal Engine

@ ClayPaky S.R.L.



SAPIENZA
UNIVERSITÀ DI ROMA

Responsabile interno:
Prof. Angelo Monti

Responsabile esterno:
Massimo Callegari

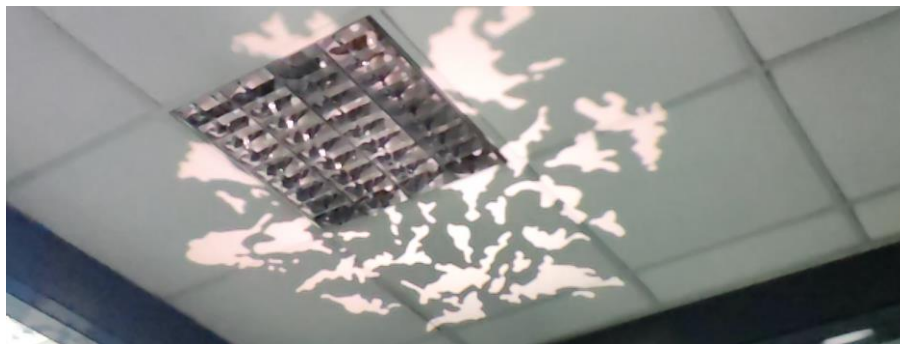
Candidato:
Luca Sorace 1910722

Introduzione

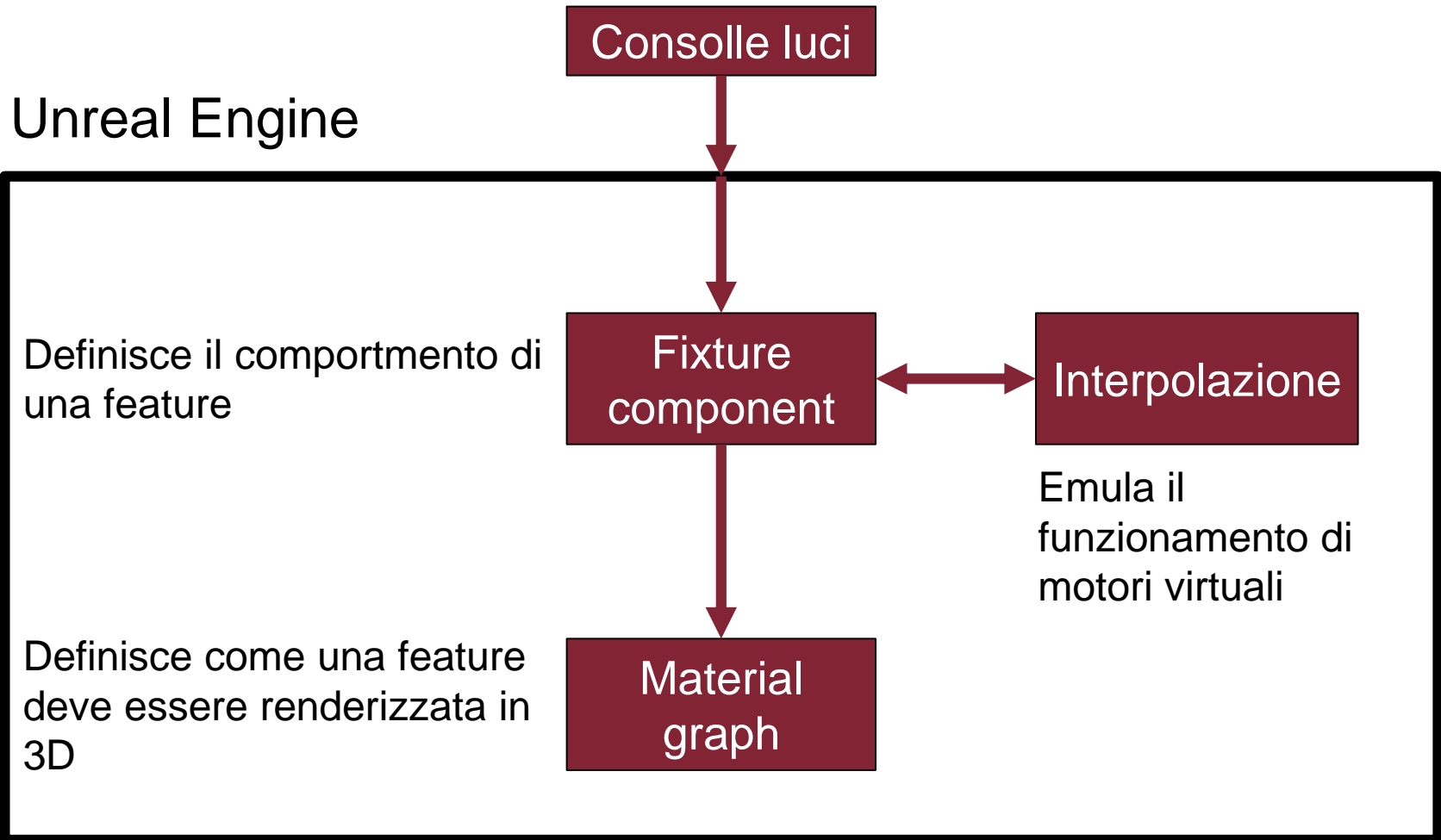
ClayPaky GDTF Importer

- Visualizer 3D per la progettazione e preprogrammazione di palcoscenici per concerti
- Realizzato su UnrealEngine, motore grafico sviluppato dalla Epic Games
- Fork del plugin ufficiale DMX/VirtualProduction realizzato dalla Epic Games stessa
- Import di fari in formato GDTF

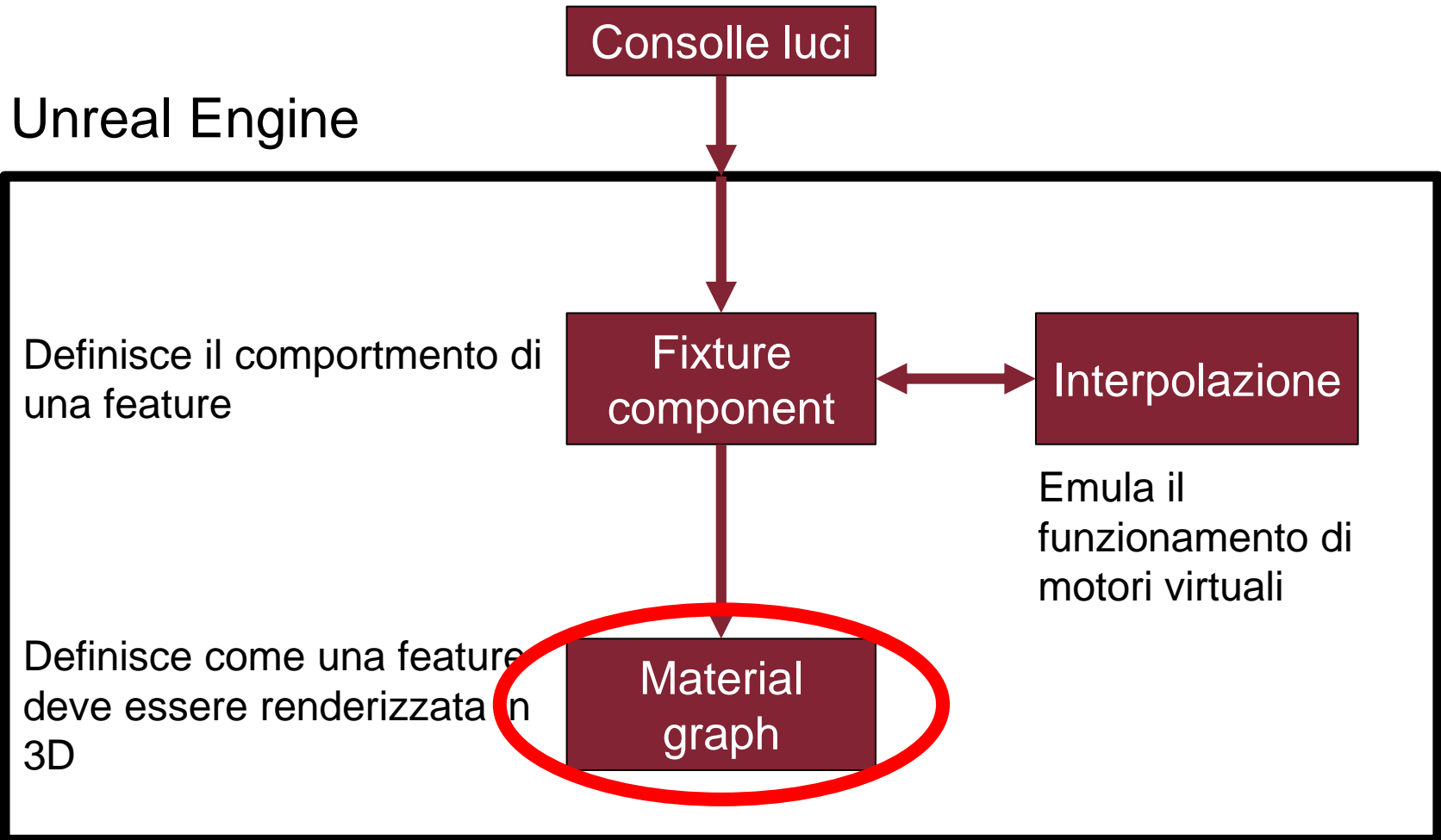
Introduzione – Com'è fatta una vera fixture



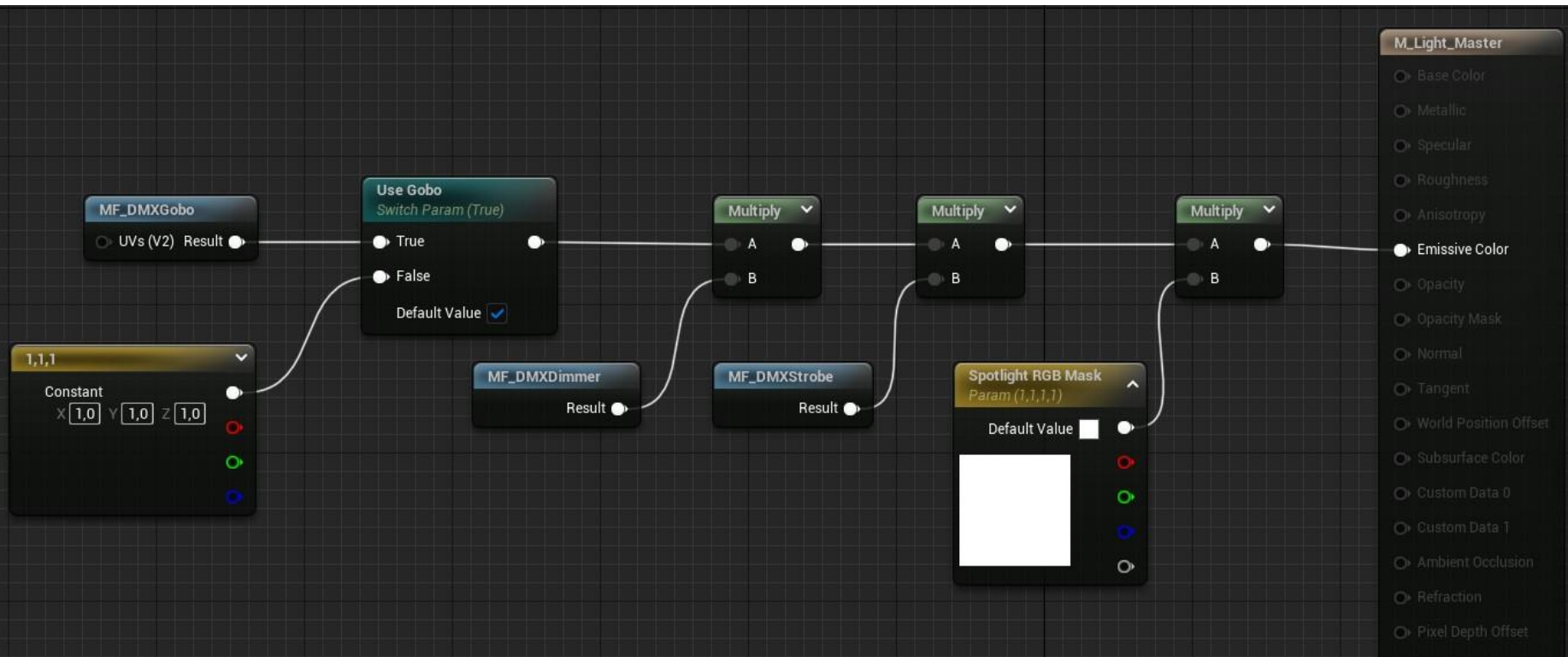
Introduzione – Flow dei dati



Introduzione – Flow dei dati

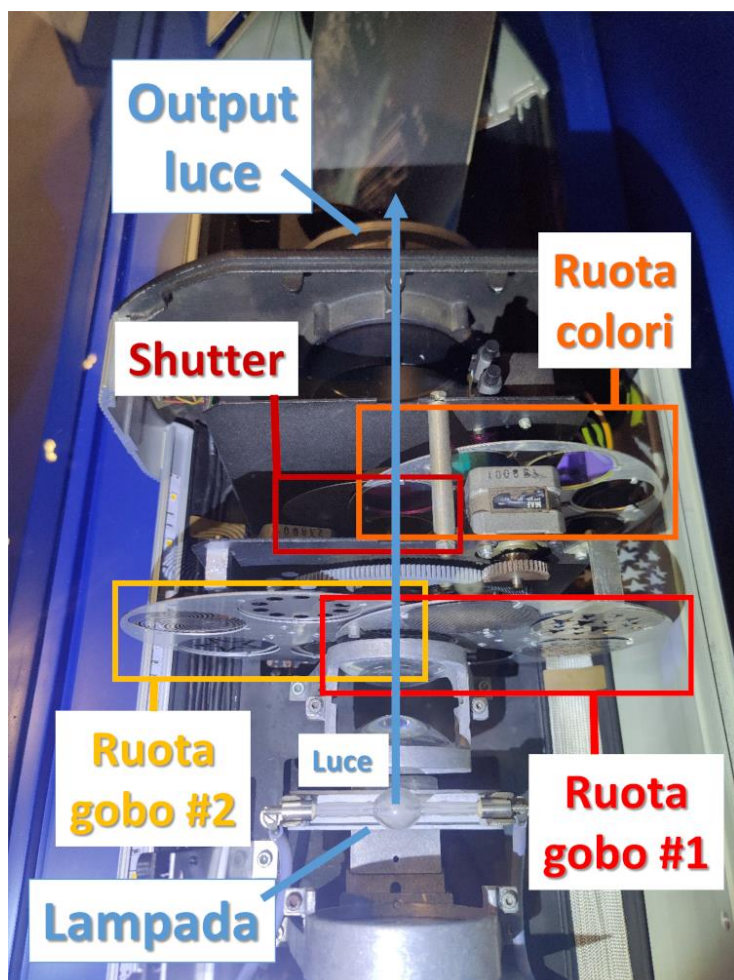


Pipeline di rendering – Stato preliminare



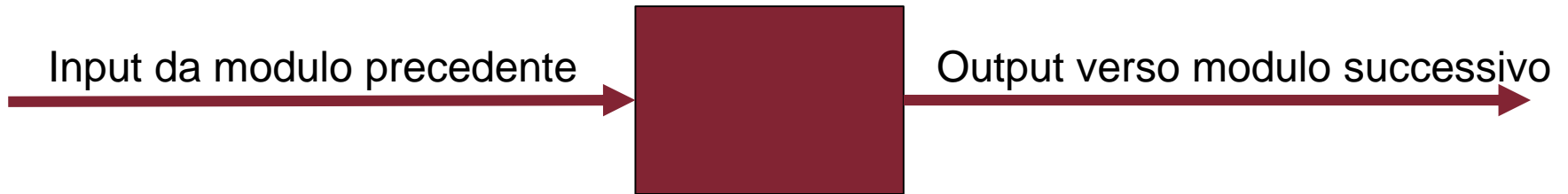
- Impossibilità di implementare nuove features
- Impossibilità di implementare più istanze della stessa feature

Pipeline di rendering – Moduli in una vera fixture



Pipeline di rendering – Idea

Istanza di una feature



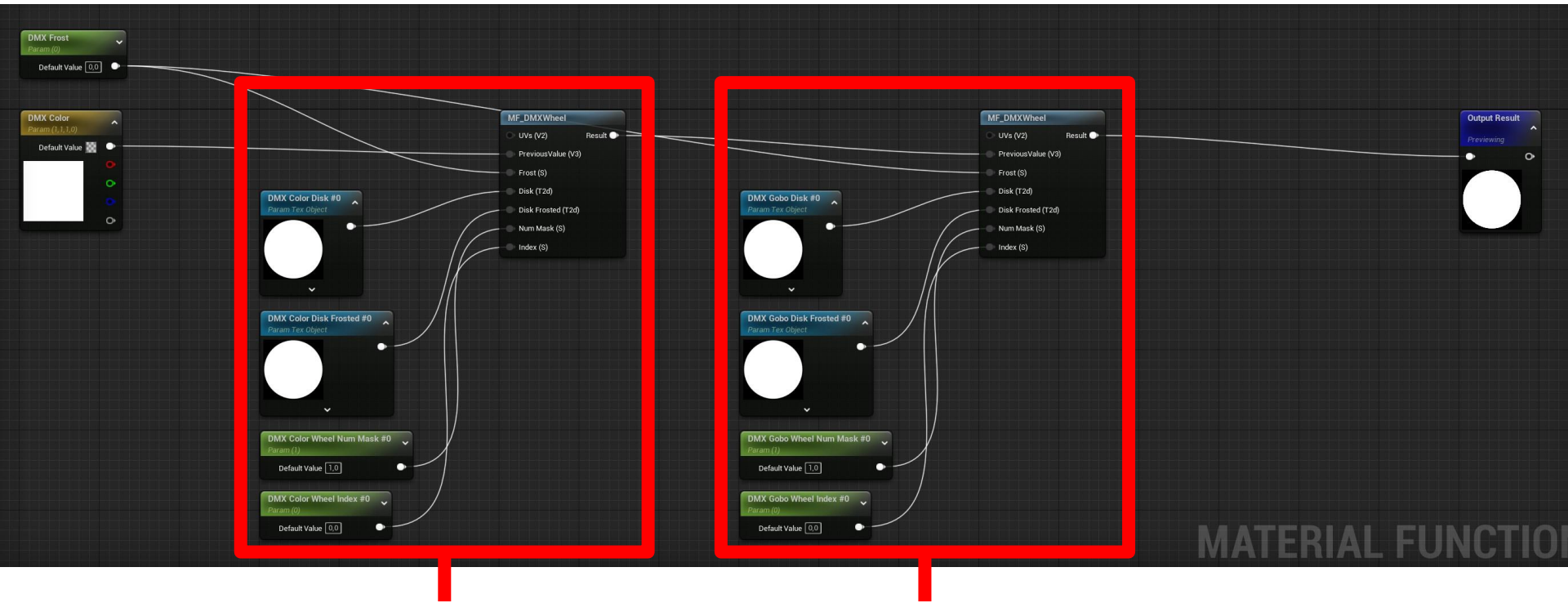
$$m_i = m_{i-1} - (1 - f_i)$$

Le features preesistenti dovranno essere modificate
per rispecchiare questo nuovo paradigma

Pipeline di rendering – Idea

- Analisi delle features presenti in un faro che stiamo importando attraverso il formato GDTF
- Attraverso le API messe a disposizione da Unreal Engine, andiamo a generare un nuovo materiale contenente le features collegate a catena
- Il primo nodo di questa catena prende in input il colore che la luce dovrà assumere
- L'output dell'ultimo nodo è collegato all'output del materiale

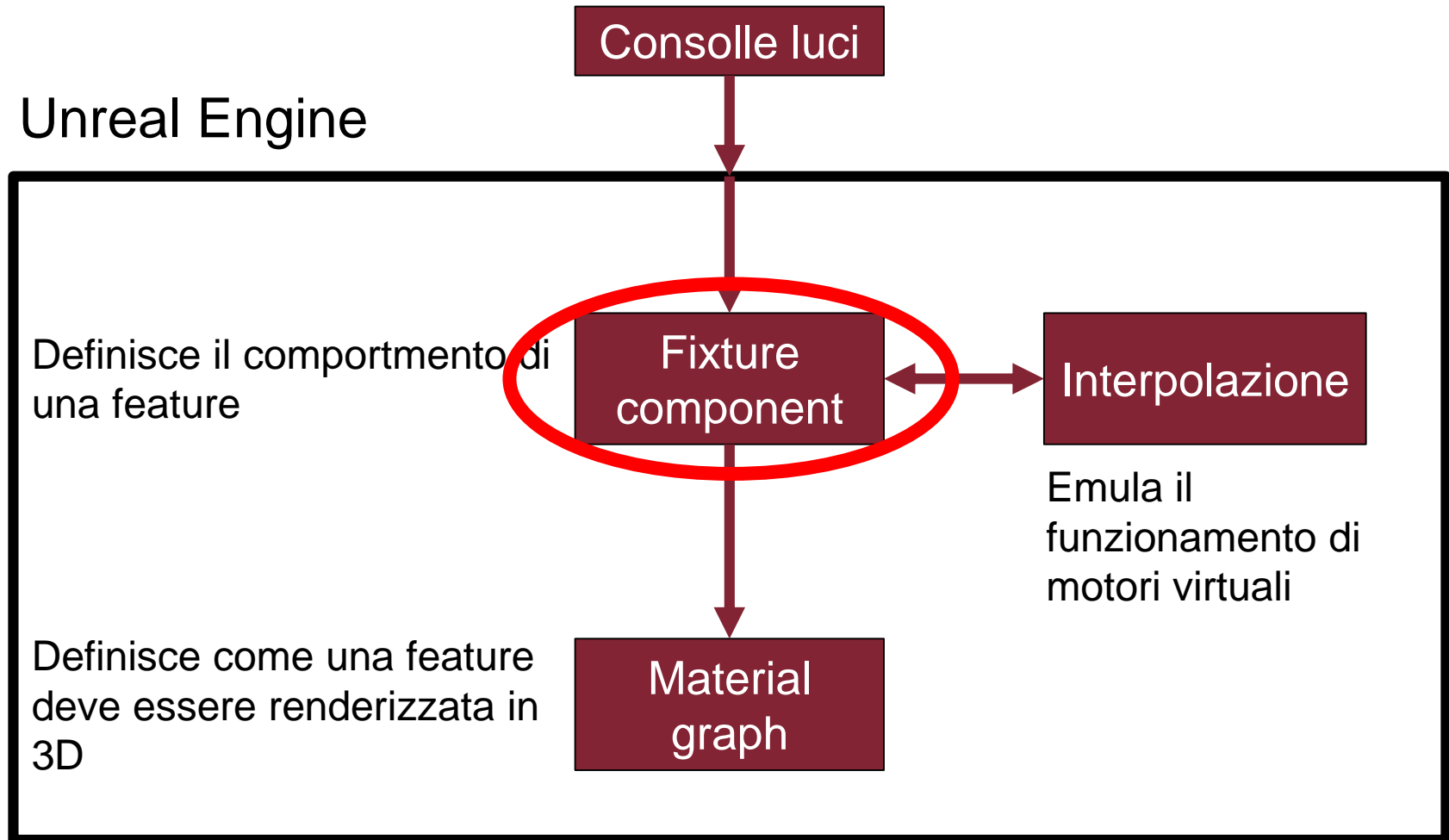
Pipeline di rendering – Risultato



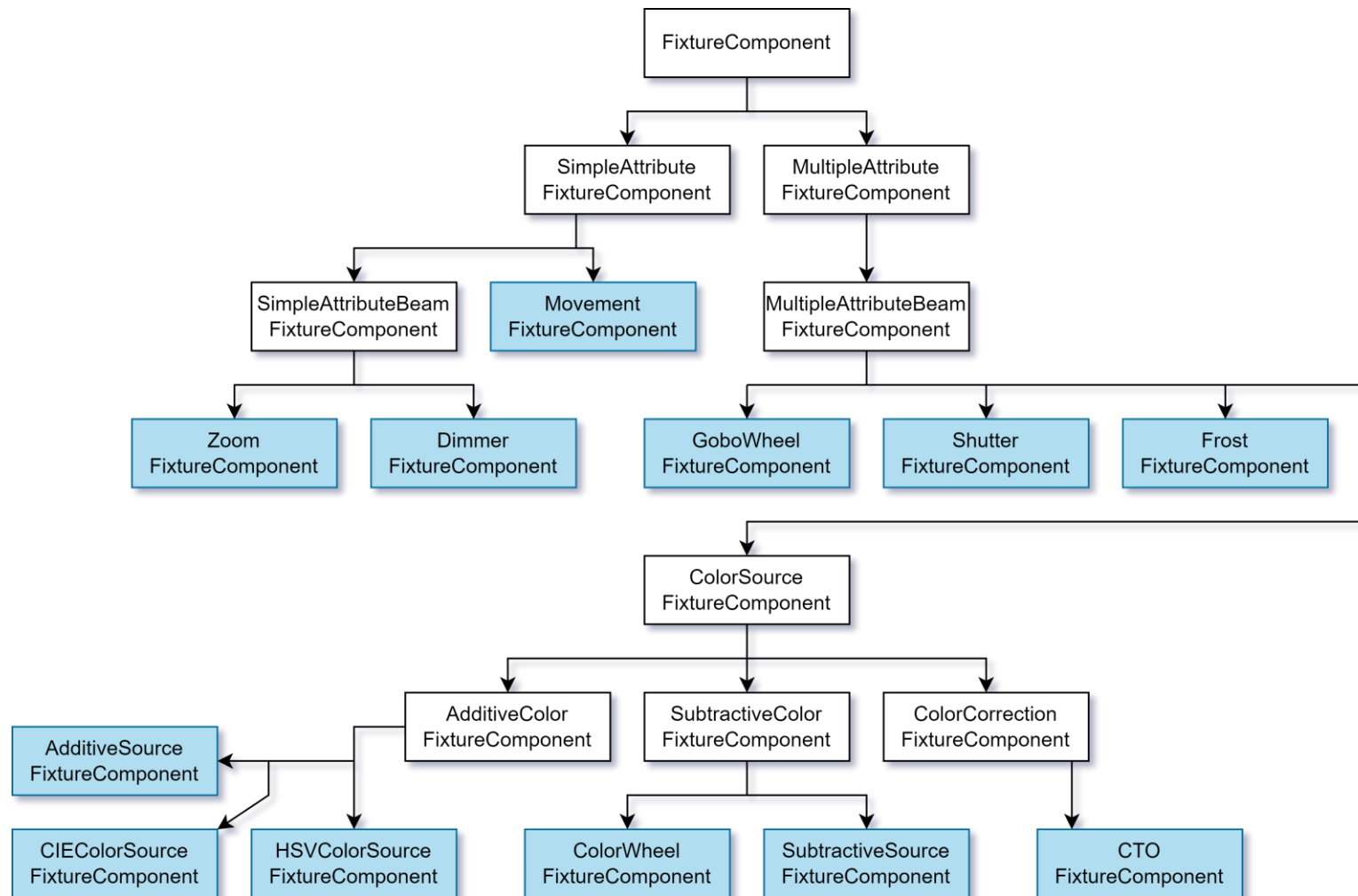
Modulo
ColorWheel#1

Modulo
GoboWheel#1

Unreal Engine



Redesign gerarchia di FixtureComponent



Redesign gerarchia di FixtureComponent – Problemi riscontrati

- Codice duplicato
- Scarsa virtualizzazione delle funzioni
- Elementi superflui
- Nessuna centralizzazione della gestione delle features
- Errori di calcolo nei valori
- Interpolazione mancante su molte features
- Gestione hard-coded di $N > 1$ canali

Redesign gerarchia di FixtureComponent

Cosa dovrebbe specificare un componente

- Funzione di init
- Come interpretare nuovi dati in input
- Come mandare in output i dati
- Come aggiornarsi ad ogni tick

Cosa dovrebbe essere gestito da FixtureComponentBase

Redesign gerarchia di FixtureComponent

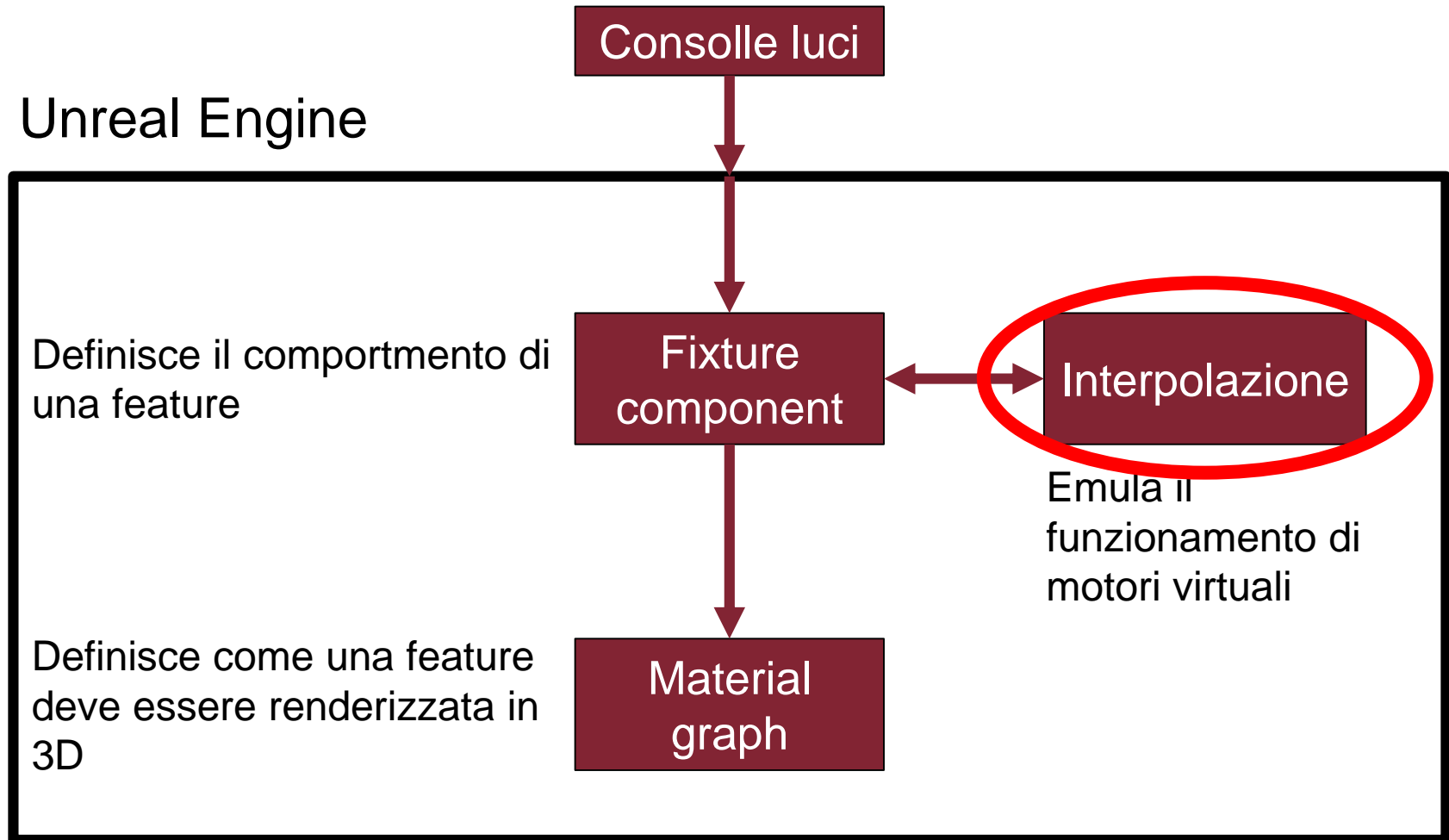
Cosa dovrebbe specificare un componente

- Funzione di init
- Come interpretare nuovi dati in input
- Come mandare in output i dati
- Come aggiornarsi ad ogni tick

Cosa dovrebbe essere gestito da FixtureComponentBase

- Caricamento dei valori di un canale in fase di init
- Inizializzazione delle interpolazioni in base ai canali del componente
- Estrazione dei valori in input e selezione automatica delle feature
- Impostazione ed aggiornamento dell'interpolazione ad ogni tick

Unreal Engine



Ristrutturazione algoritmo interpolazione – Bug riscontrati

- CurrentValue non convergeva a TargetValue
- Flickering
- Movimenti talvolta incredibilmente lenti
- Codice di difficile manutenzione
- Non fedele ai movimenti di una vera fixture

Ristrutturazione algoritmo interpolazione – Bug riscontrati

- CurrentValue non convergeva a TargetValue
- Flickering
- Movimenti talvolta incredibilmente lenti
- Codice di difficile manutenzione
- Non fedele ai movimenti di una vera fixture

Soluzione:

Creazione di un algoritmo il più possibile vicino a quello presente nelle fixture

Ristrutturazione algoritmo interpolazione – Bug riscontrati

- CurrentValue non convergeva a TargetValue
- Flickering
- Movimenti talvolta incredibilmente lenti
- Codice di difficile manutenzione
- Non fedele ai movimenti di una vera fixture

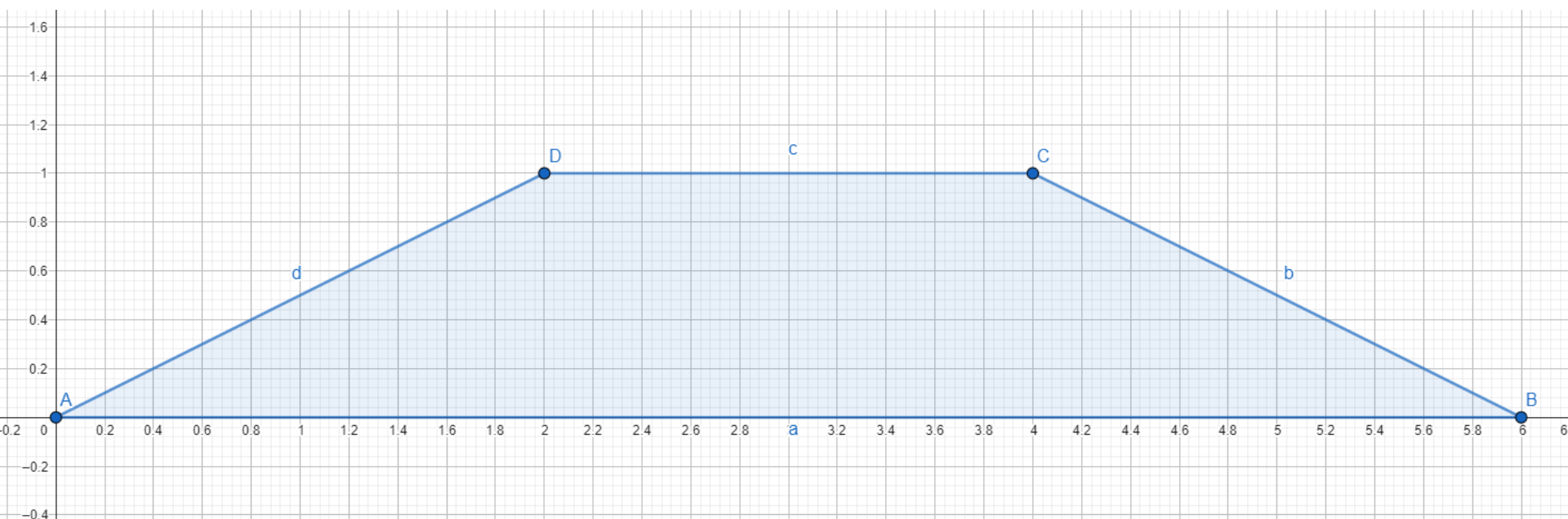
Soluzione:

Creazione di un algoritmo il più possibile vicino a quello presente nelle fixture

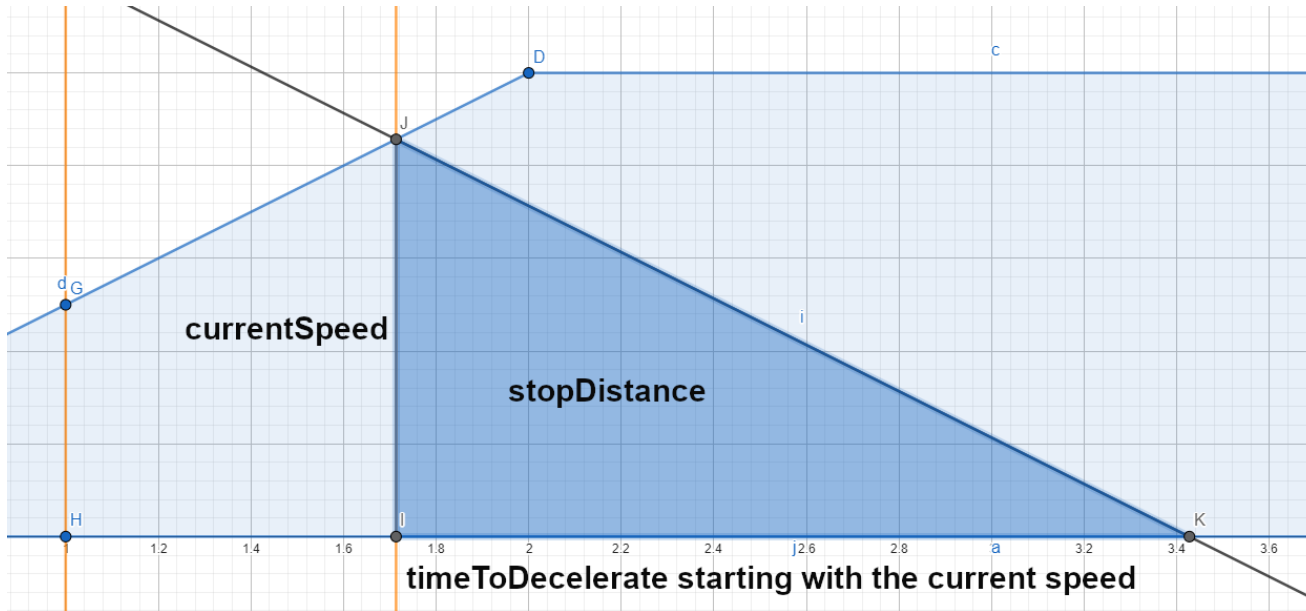
Problema:

R&D non ha voluto fornire il codice per un progetto open-source

Ristrutturazione algoritmo interpolazione

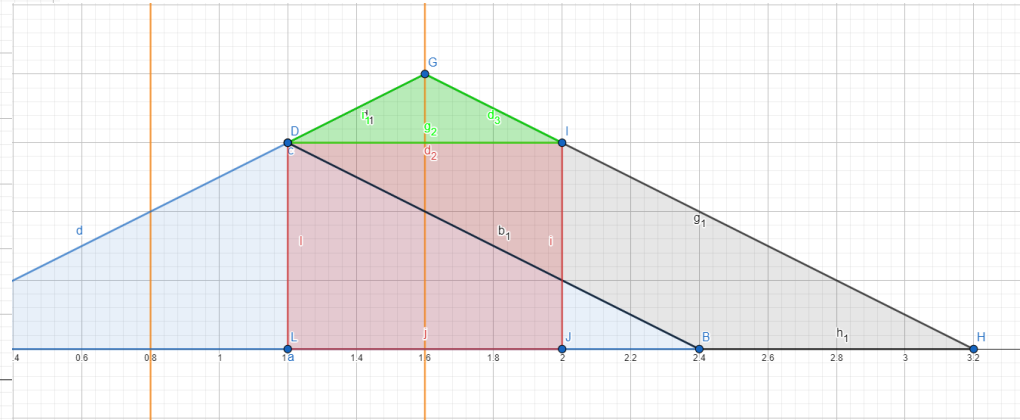
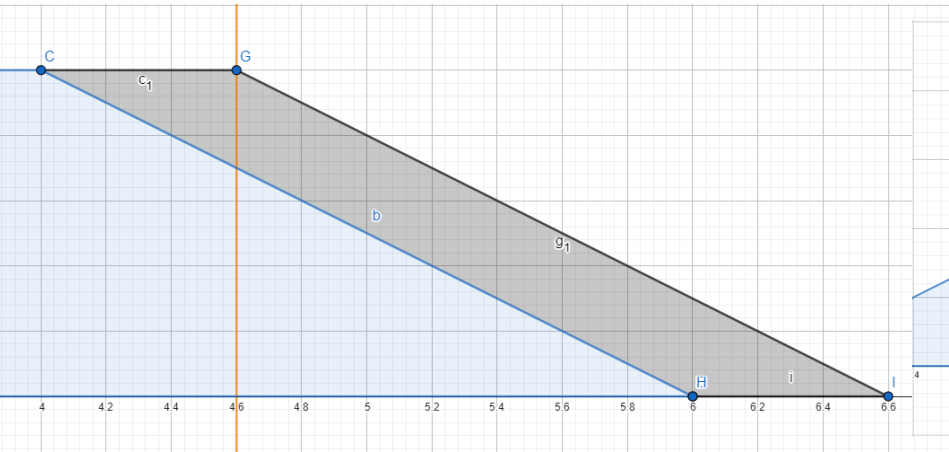


Ristrutturazione algoritmo interpolazione – getStopDistance()



$$stopDistance = \frac{TTF A * currentSpeed^2}{2}$$

Ristrutturazione algoritmo interpolazione – compensateLateCall()



$$t = \frac{D_i}{S_i}$$

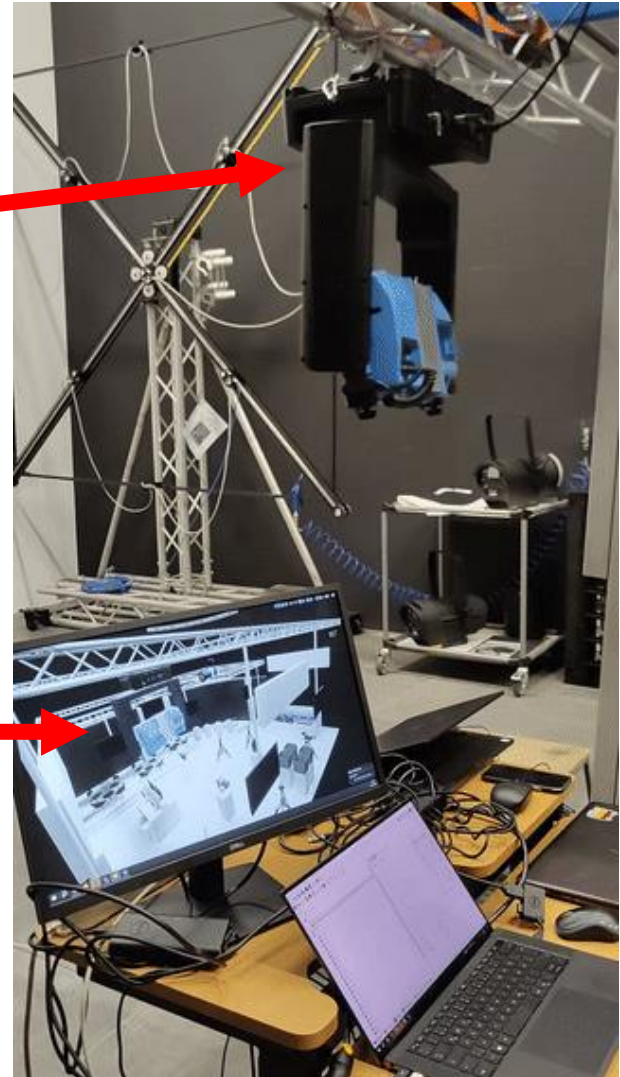
$$\text{se } S_i = S_{i-1}$$

$$\frac{a}{4}t^2 - S_it + D_i = 0 \quad \text{altrimenti}$$

Ristrutturazione algoritmo interpolazione – Demo

**Arri Skypanel X
+ orbiter**

**Unreal Engine con il
mio algoritmo di
interpolazione**

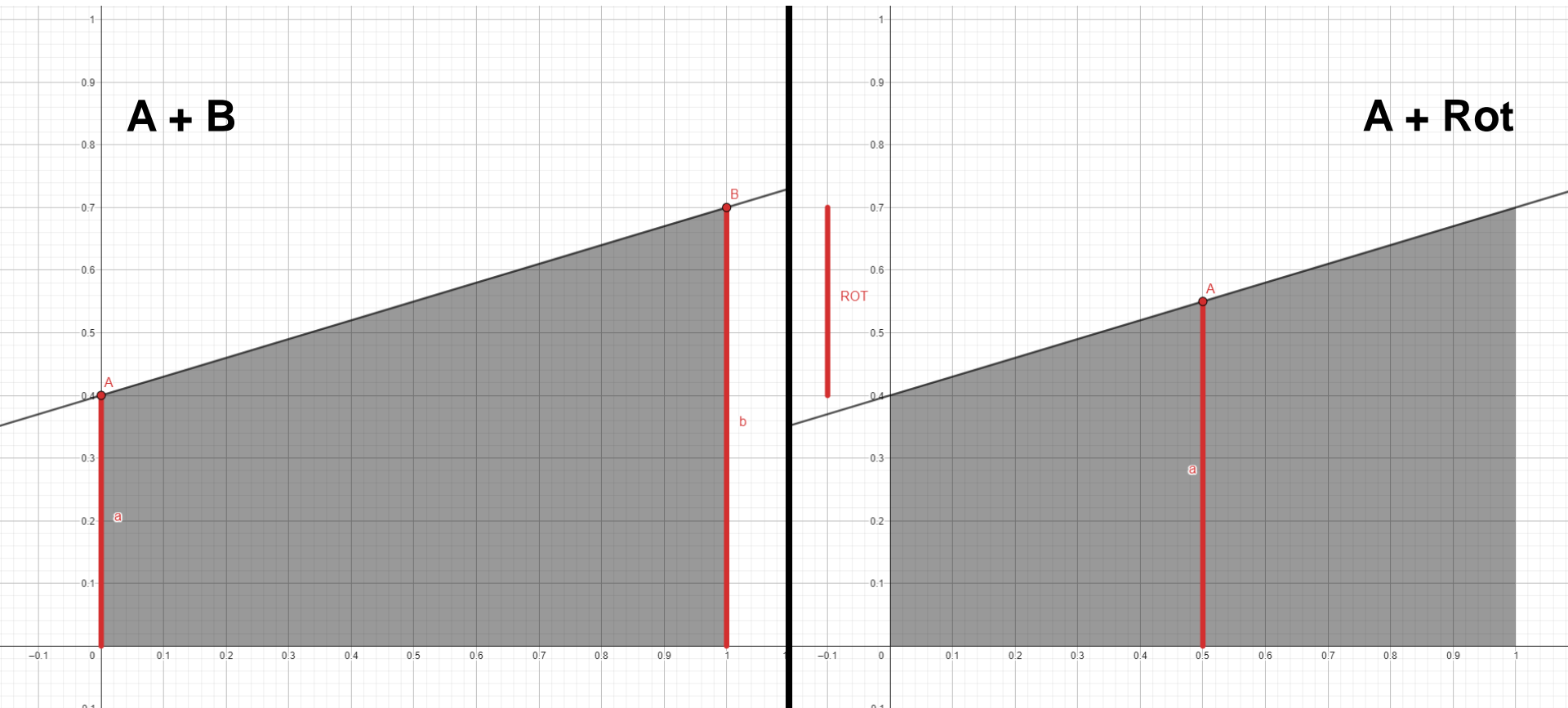


Finalmente si possono implementare nuove features!

Nuove funzionalità – Framing system



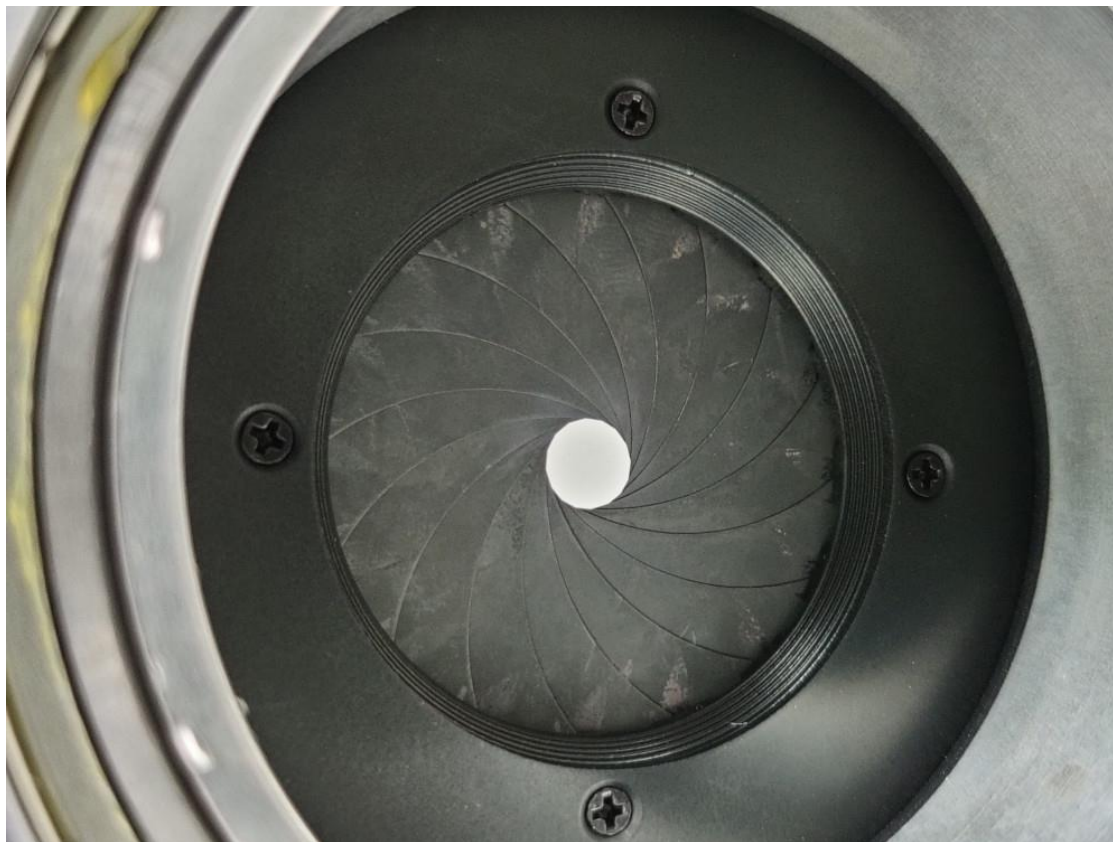
Nuove funzionalità – Framing system



$$y > x(B - A) + A$$

$$y > \tan(R)x + \left(A - \frac{\tan(R)}{2}\right)$$

Nuove funzionalità – Iris



$$i > \sqrt{(2x - 1)^2 + (2y - 1)^2}$$

Nuove funzionalità – Frost

$$d = |y - g(x)|$$

$$d = \left| \sqrt{(2x - 1)^2 + (2y - 1)^2} - i \right|$$

Nuove funzionalità – Frost

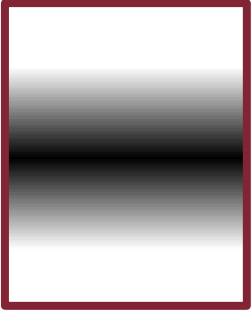
$$d = |y - g(x)| \qquad d = \left| \sqrt{(2x - 1)^2 + (2y - 1)^2} - i \right|$$

$$\text{remapFrost}(f) := f * 0.05 + 0.002$$

Nuove funzionalità – Frost

$$d = |y - g(x)| \qquad d = \left| \sqrt{(2x - 1)^2 + (2y - 1)^2} - i \right|$$

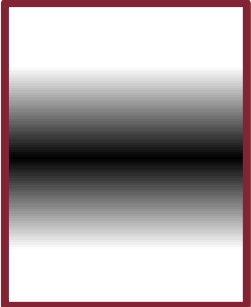
$$\text{remapFrost}(f) := f * 0.05 + 0.002$$

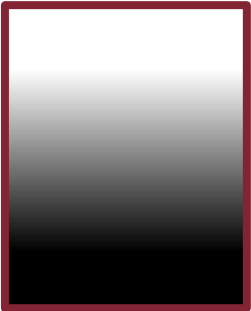

$$\text{remapDist}(d, f) := \begin{cases} 1 & \text{se } d > \text{remapFrost}(f) \\ \frac{d}{\text{remapFrost}(f)} & \text{altrimenti} \end{cases}$$

Nuove funzionalità – Frost

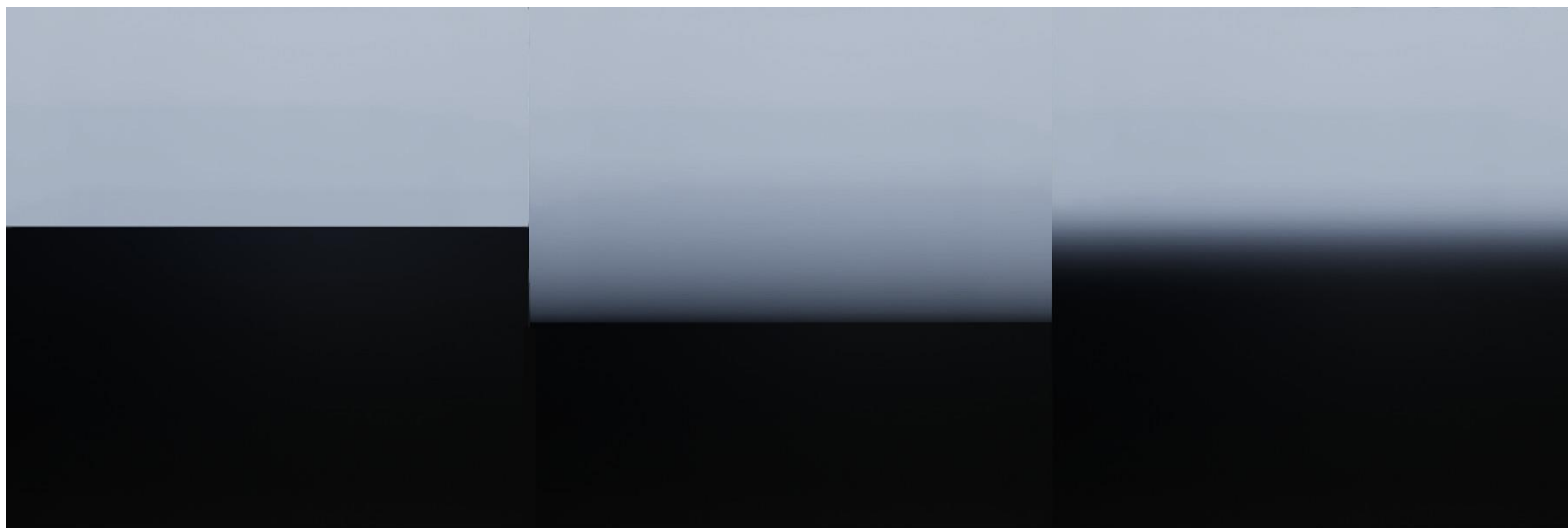
$$d = |y - g(x)| \qquad d = \left| \sqrt{(2x - 1)^2 + (2y - 1)^2} - i \right|$$

$$\text{remapFrost}(f) := f * 0.05 + 0.002$$


$$\text{remapDist}(d, f) := \begin{cases} 1 & \text{se } d > \text{remapFrost}(f) \\ \frac{d}{\text{remapFrost}(f)} & \text{altrimenti} \end{cases}$$

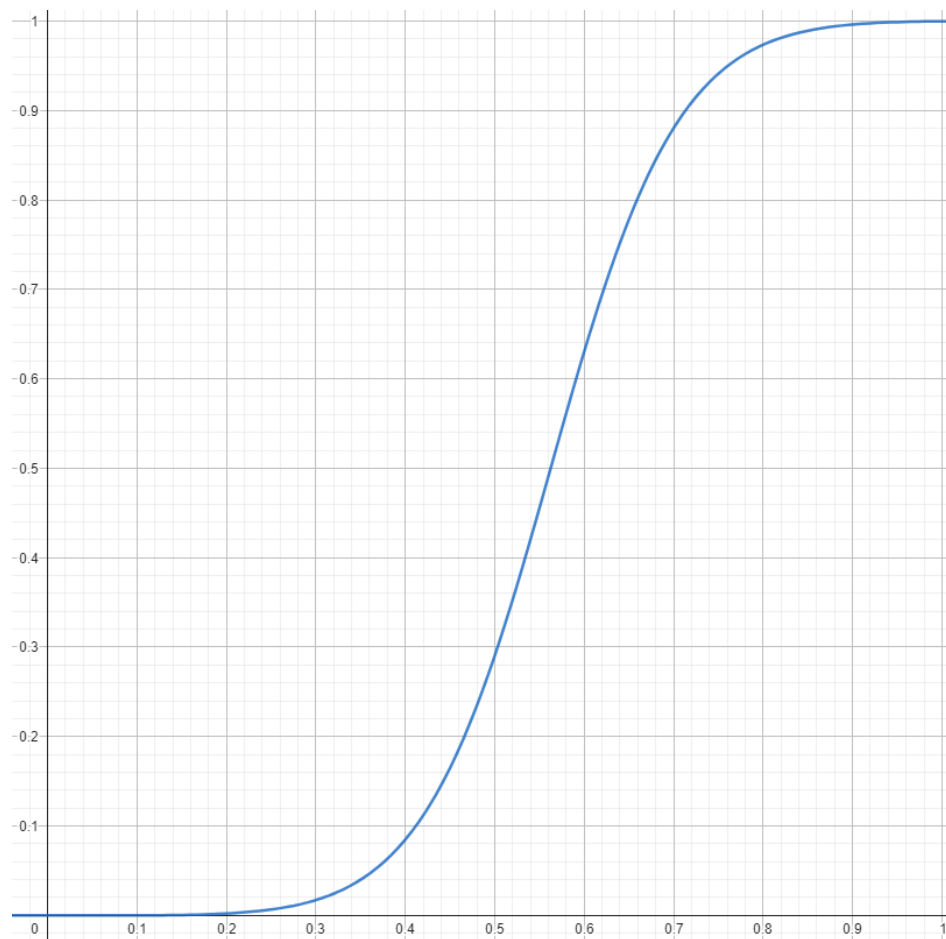

$$\text{fade}(d, f) := \begin{cases} \frac{\text{remapDist}(d, f)}{2} & \text{se } y > g(x) \\ -\frac{\text{remapDist}(d, f)}{2} & \text{altrimenti} \end{cases}$$

Nuove funzionalità – Frost



Nuove funzionalità – Frost

$$\textit{sigmoid}(v) := \frac{1}{1 + \left(\frac{v}{1.125-v}\right)^{-4}}$$

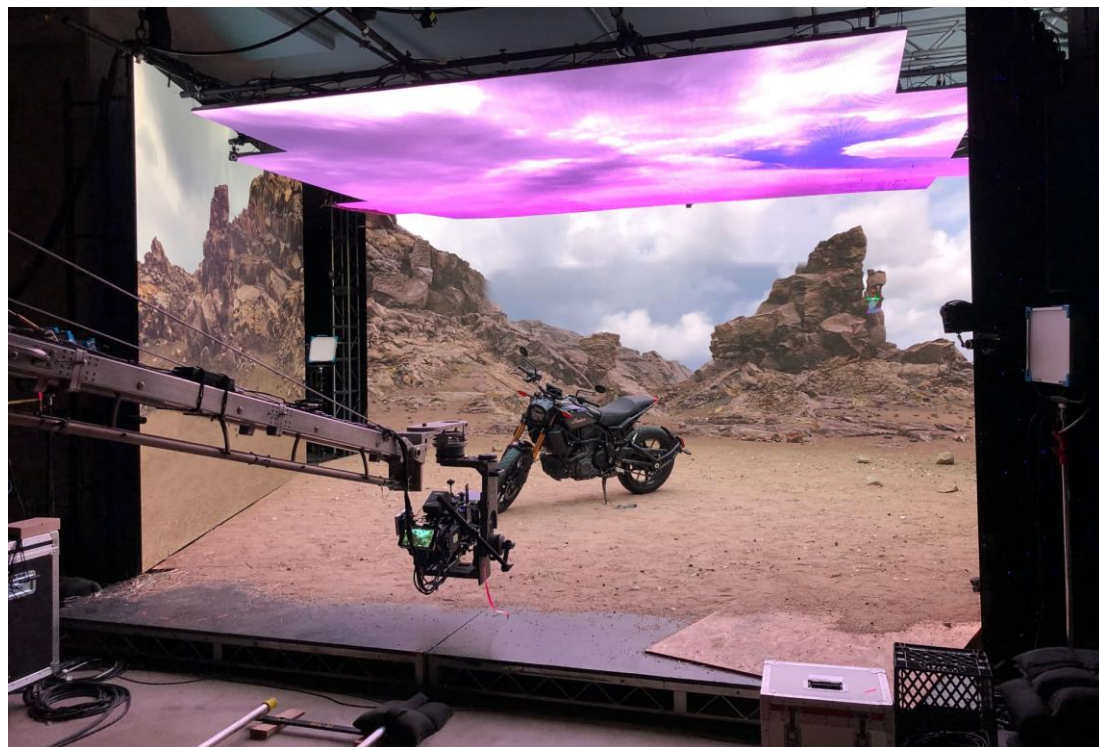


Problemi riscontrati



- Scarsa documentazione
- Mala gestione degli oggetti
- Mancanza di guide e tutorial
- Scarso ascolto delle richieste degli utenti

Sviluppi futuri – Acquisizione da parte ARRI & collaborazione con Epic Games



Grazie per la vostra attenzione.



SAPIENZA
UNIVERSITÀ DI ROMA

Responsabile interno:
Prof. Angelo Monti

Responsabile esterno:
Massimo Callegari

Candidato:
Luca Sorace 1910722

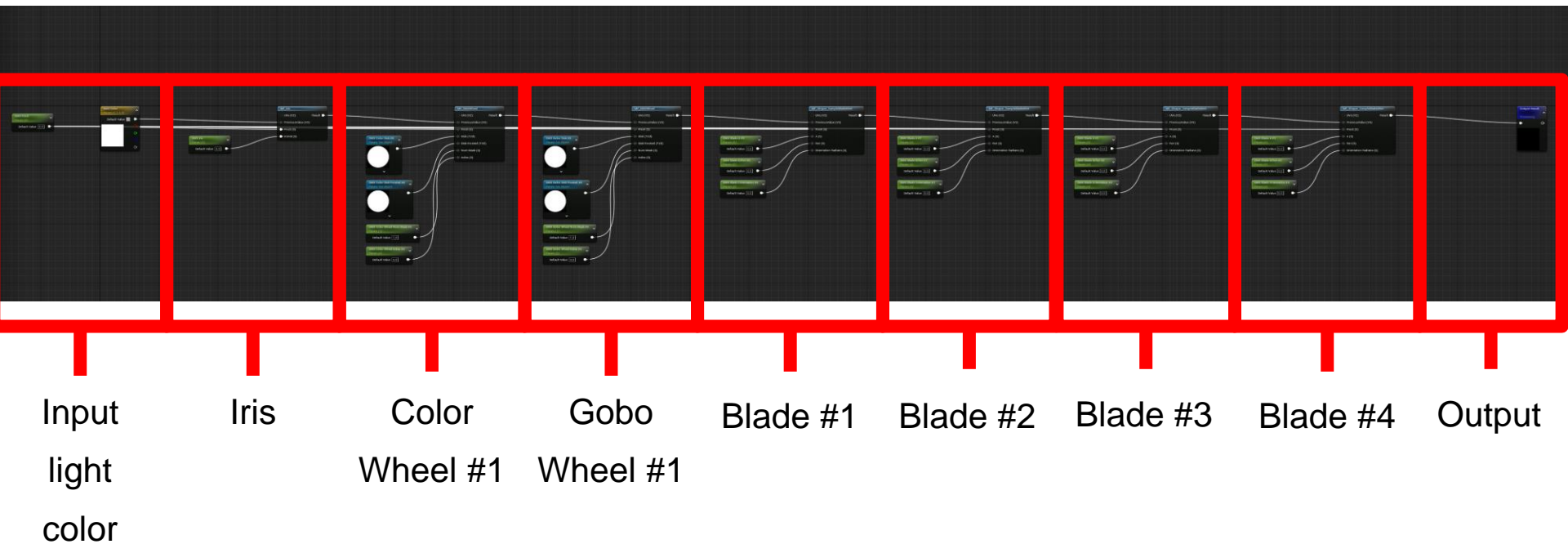
Generazione della pipeline di rendering

```
for (FDMXImportGDTFWheel wheel : this->mWheels) {
    FCPGDTFWheelImporter::WheelType wType = FCPGDTFWheelImporter::GetWheelType(this->mGdtfDescription, wheel.Name, selectedMode);
    this->mWheelsNo[wType]++;
}

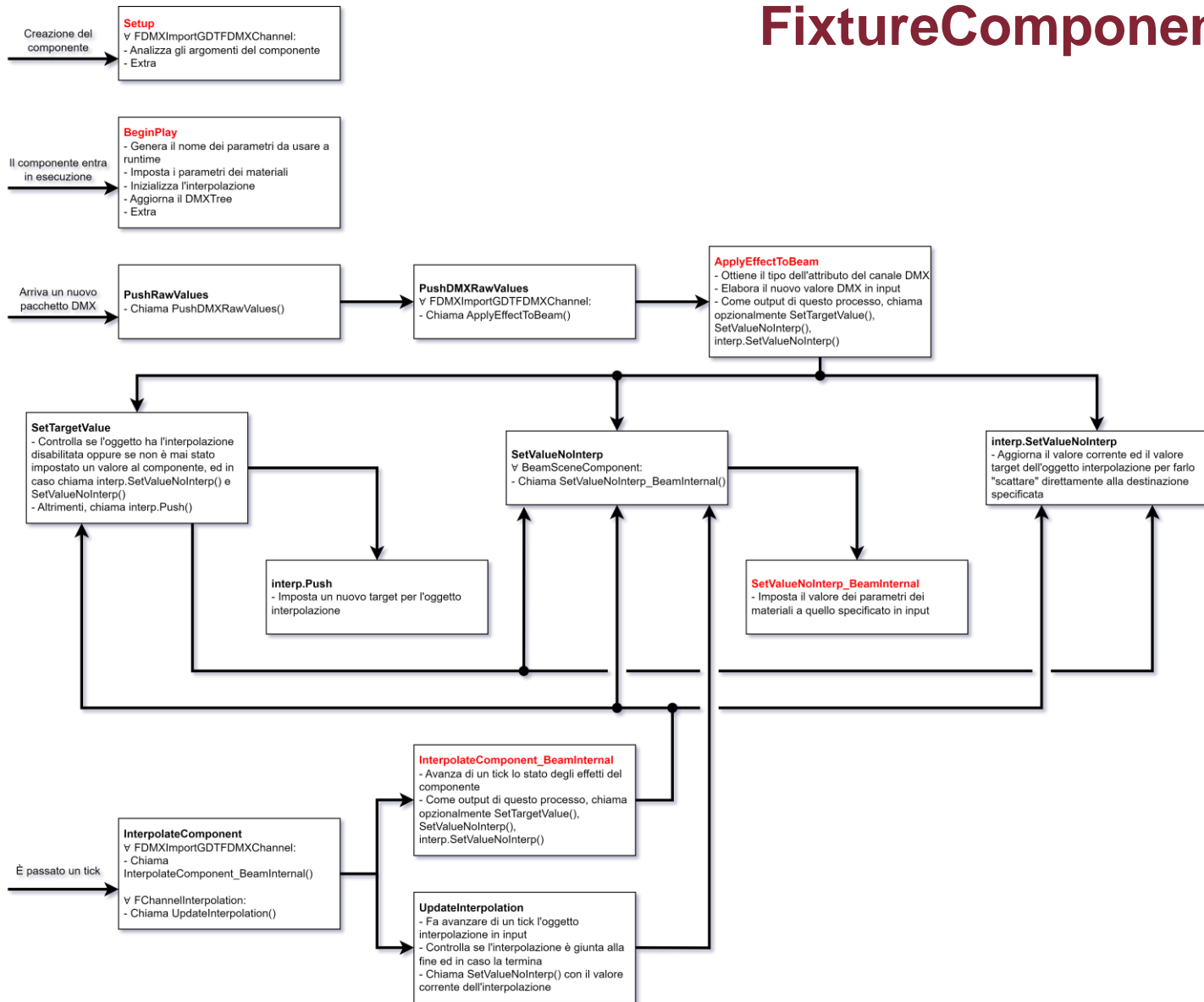
for (int i = 0; i < components.Num(); i++) {
    UCPGDTFShaperFixtureComponent *shaper = Cast<UCPGDTFShaperFixtureComponent>(components[i]);
    if (shaper != nullptr) { shapers.Add(shaper); continue; }
    UCPGDTFIrisFixtureComponent *iris = Cast<UCPGDTFIrisFixtureComponent>(components[i]);
    if (iris) { hasIris = true; continue; }
}
```

```
template <typename MType>
MType* CPGDTRenderPipelineBuilder::generateMaterialExpression(UMaterial* material) {
    MType* obj = NewObject<MType>(material);
    obj->Material = material;
    obj->UpdateMaterialExpressionGuid(false, true);
    return obj;
}
```

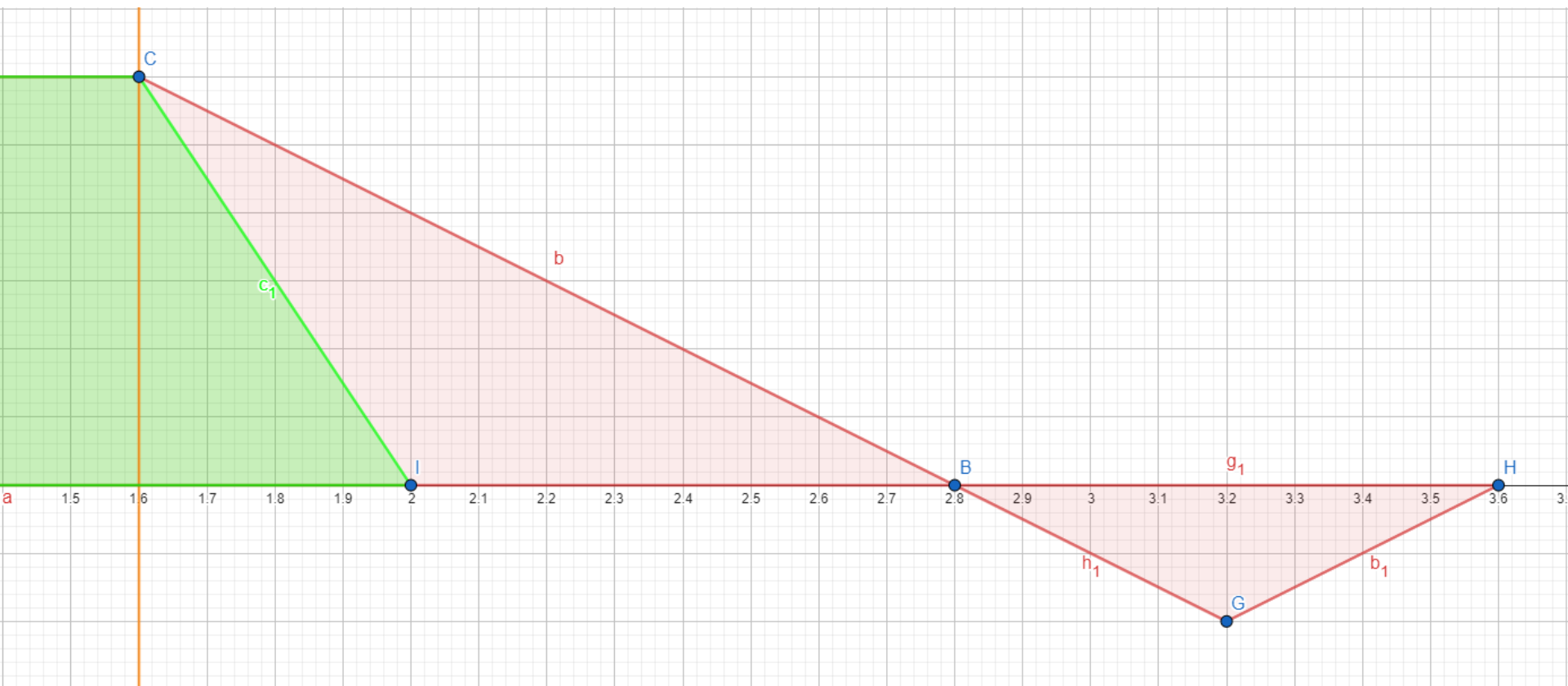
Pipeline completa



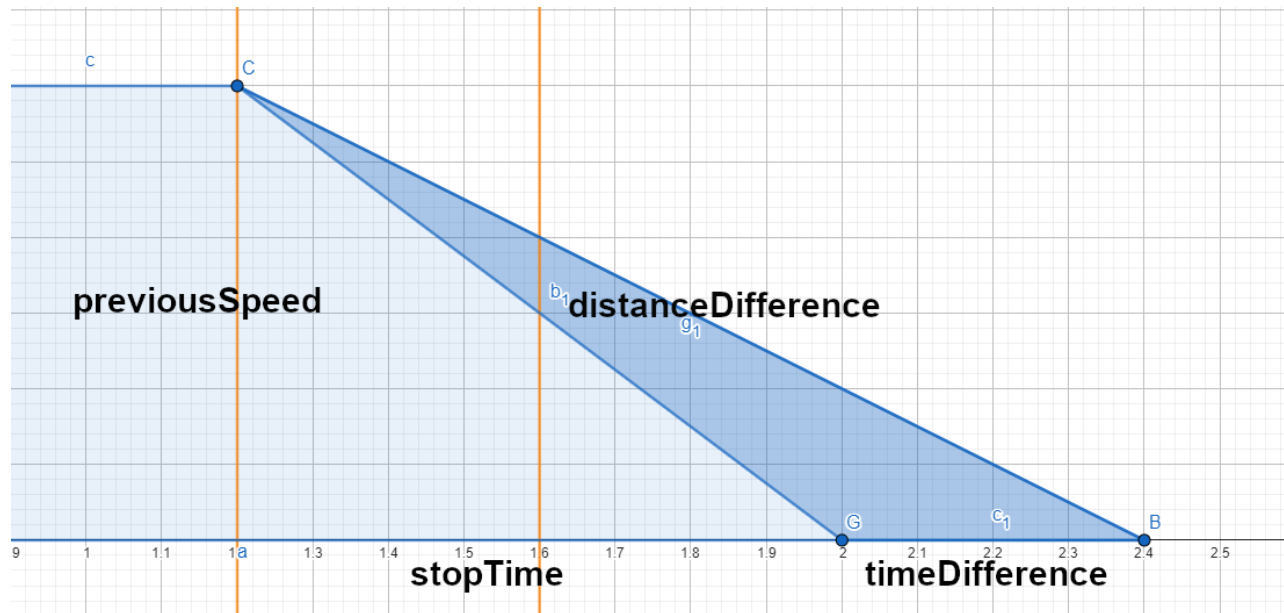
FixtureComponent – Call tree



Ristrutturazione algoritmo interpolazione – overrideDeceleration()



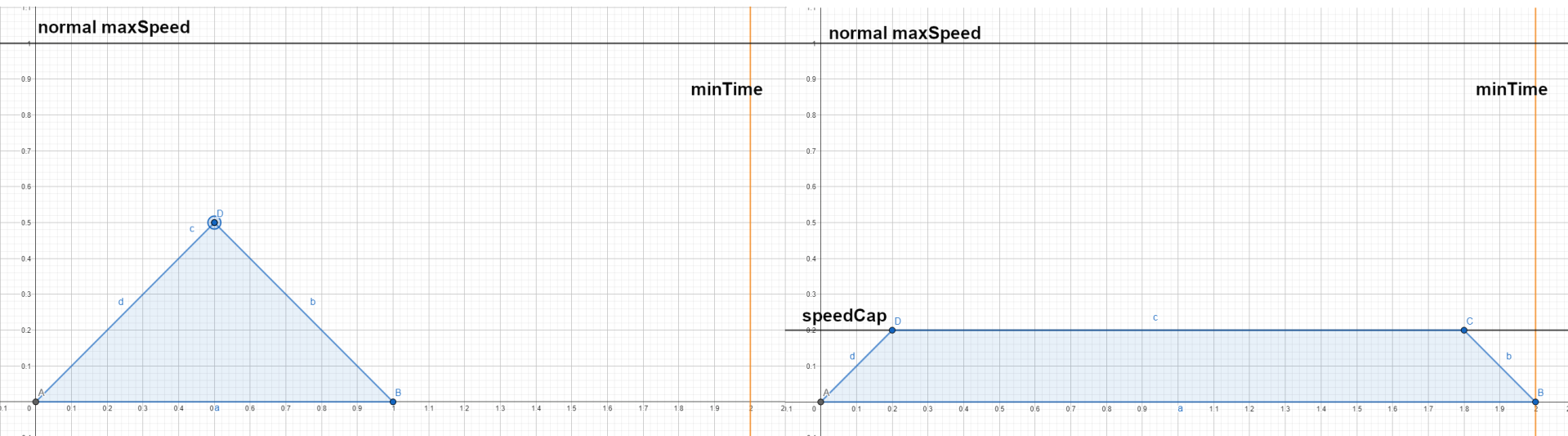
Ristrutturazione algoritmo interpolazione – overrideDeceleration()



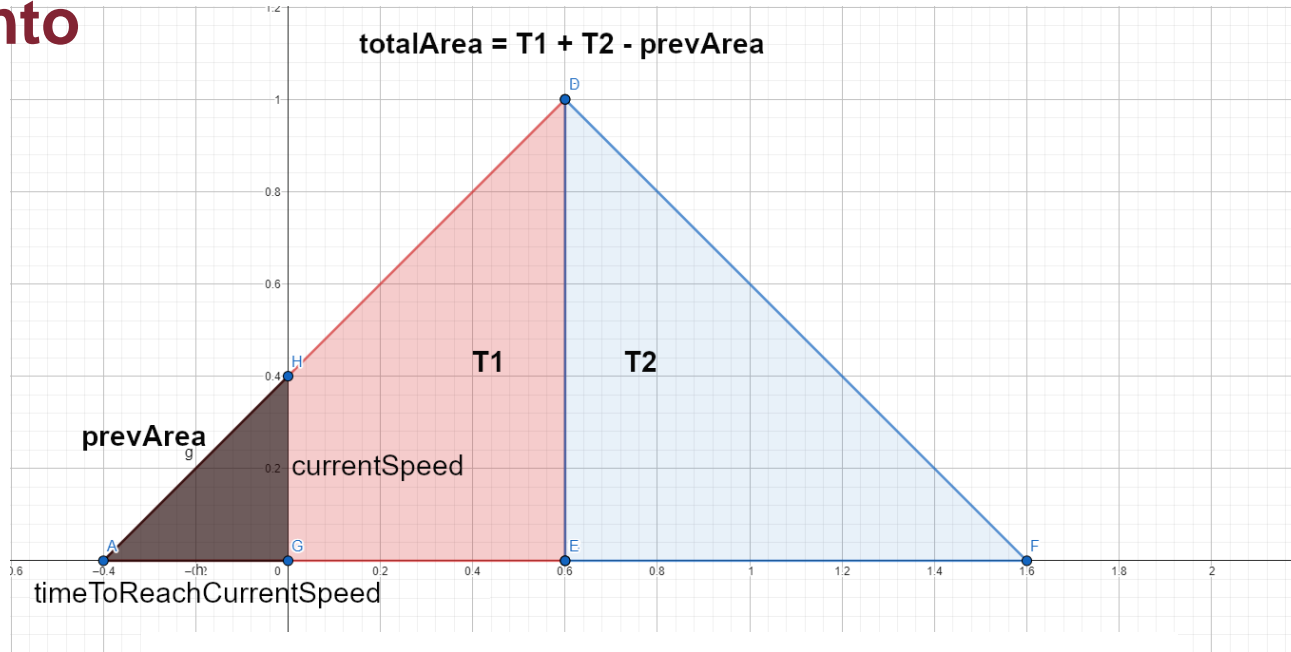
$$t = s - \frac{2D}{S_{i-1}}$$

$$a = \frac{0 - S_{i-1}}{t - 0} = -\frac{S_{i-1}}{t}$$

Ristrutturazione algoritmo interpolazione – setSpeedCap()



Interpolazione – setSpeedCap(): Calcolo del tempo di movimento



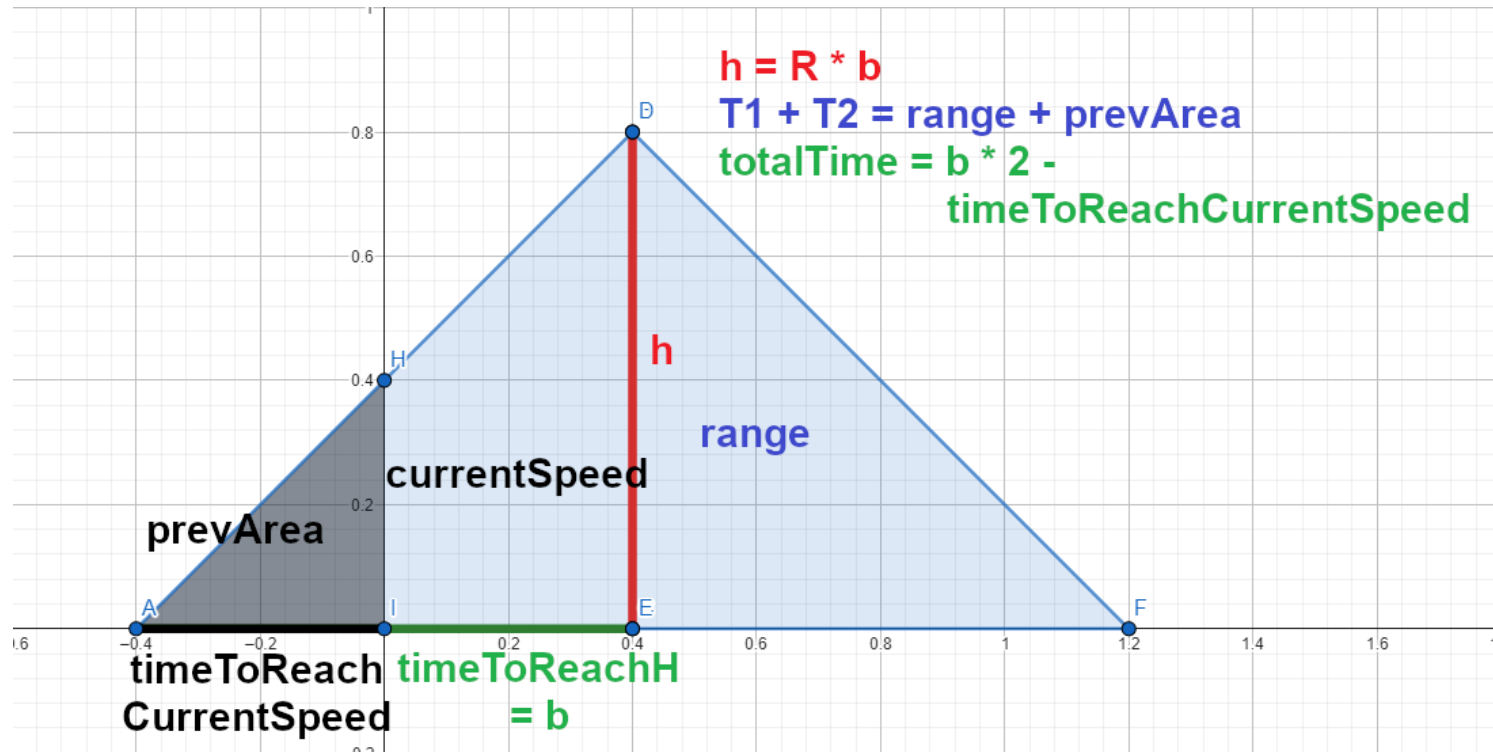
$$TTRCS = TTF A * currentSpeed$$

$$prevArea = \frac{TTRCS * currentSpeed}{2} = \frac{TTF A * currentSpeed^2}{2}$$

$$A_t = T_1 + T_2 = 2 \frac{TTF A * 1}{2} - prevArea = TTF A - prevArea$$

Interpolazione – setSpeedCap(): Calcolo del tempo di movimento

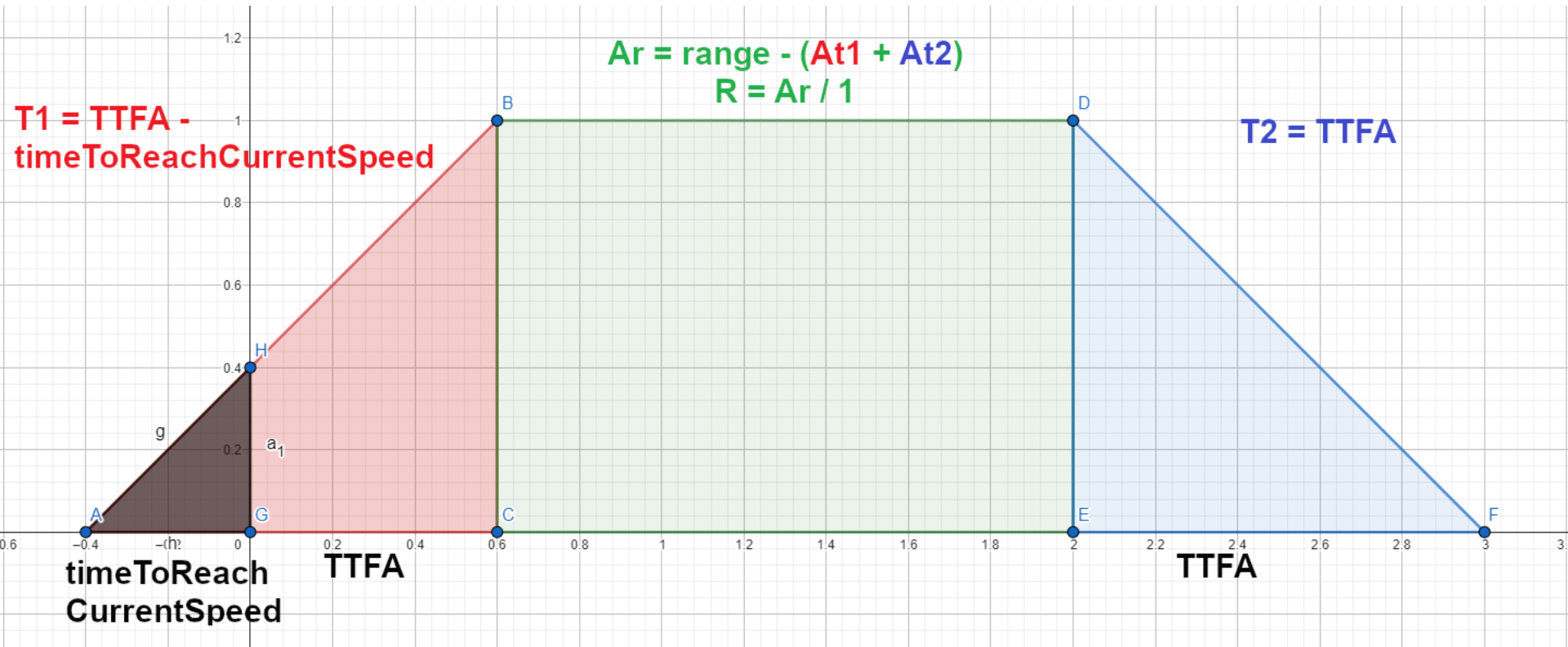
$$\text{Se } A_t \geq D$$



$$B = 2\sqrt{\frac{A + \text{prevArea}}{a}} - TTRCS$$

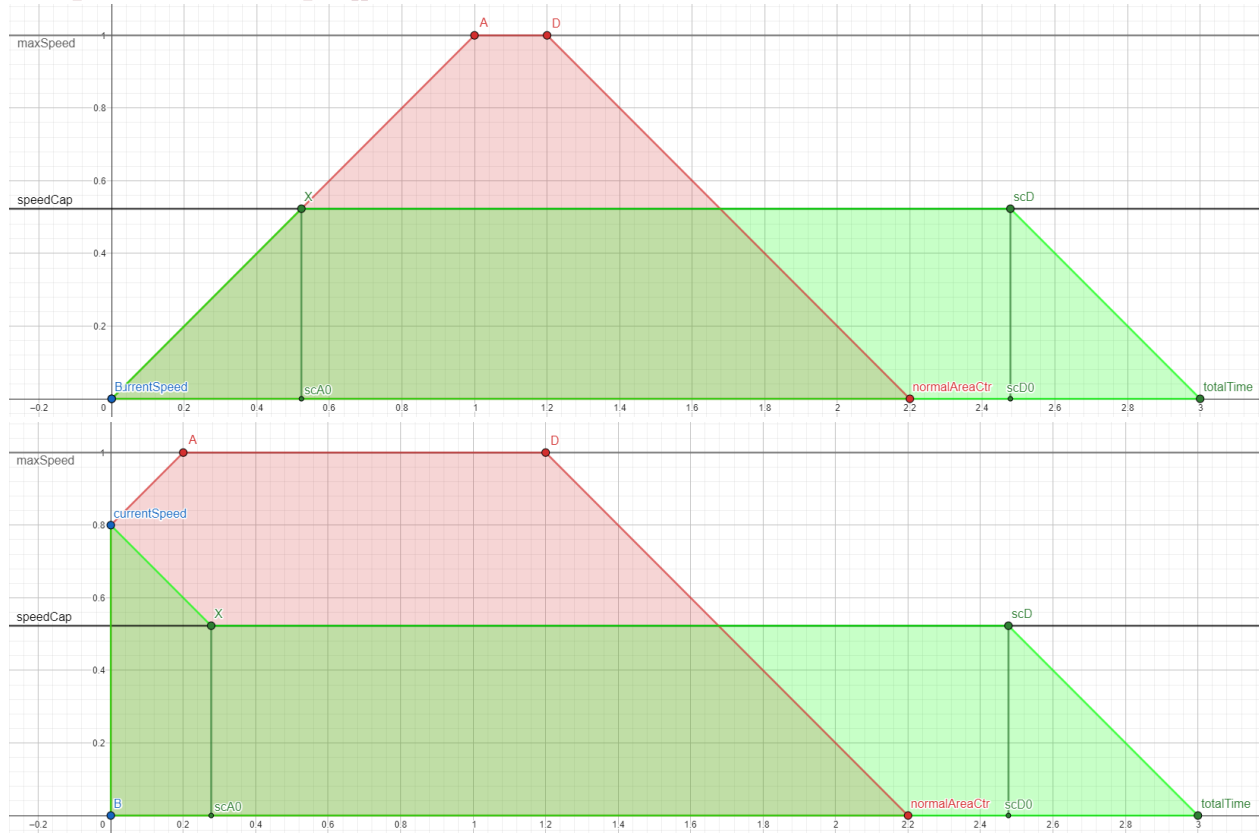
Interpolazione – setSpeedCap(): Calcolo del tempo di movimento

$$\text{Se } A_t < D$$



$$time = (TTFA * 2 - TTRCS) + (distanceDifference - A_t)$$

Ristrutturazione algoritmo interpolazione – setSpeedCap()



$$\frac{S_i |Tx - TS_i|}{2} + \frac{x |Tx - TS_i|}{2} + \frac{Tx^2}{2} + tx - D = 0$$

Nuove funzionalità – Framing system

$$\textit{rotateUV}(a) := \begin{bmatrix} \cos(a)(x - \frac{1}{2}) + \sin(a)(y - \frac{1}{2}) + x \\ \cos(a)(y - \frac{1}{2}) + \sin(a)(x - \frac{1}{2}) + y \end{bmatrix}$$

Nuove funzionalità – Frost

```
float sigmoid(float f){  
    const float exp = -4;  
    f = saturate(f);  
    return 1 / (1 + pow((f / (1.125 - f)), exp));  
}  
  
float applyFrost(bool passed, float d, float frost){  
    float max = 0.05 * frost + 0.002; // Hardcoded blur value  
    float clamp = d > max;  
    d = clamp + (1 - clamp) * (d / max);  
  
    return sigmoid((passed * 2 - 1) * (d / 2) + 0.5);  
}
```


Problemi riscontrati



- Dettagli minori omessi dalla specifica
- Valori ridondanti e troppo specifici, che spingono le case produttrici a inserire dati approssimati

Stato dell'arte

Features implementate	Features da implementare
Pan / Tilt + Rotazione continua	Prisma
Dimmer / Shutter / Strobe	Focus
Frost	Ruota animazione
Zoom / Iris	Canali multifunzione
CMYRGB+ (inclusi WW e CW)	
CTO/CTB/CTC/Tint	
CIE/HSB	
Ruota colori	
Macro colori	
Framing system	
Gobo fisse e rotanti	