GPU raytracer

3

Generirao Doxygen 1.8.17

Poglavlje 1

Hijerarhijsko kazalo

1.1 Stablo klasa

Stablo naslijeđivanja je složeno približno po abecedi:

Poglavlje 2

Skupno kazalo

2.1 Sve klase

Popis svih klasa, unija i struktura s kratkim opisom :

Poglavlje 3

Kazalo datoteka

3.1 Popis datoteka

Popis svih dokumentiranih datoteka, s kratkim opisom:

Poglavlje 4

Dokumentacija klasa

4.1 Opis klase GPUCopyable< T >

Public članovi

- **GPUCopyable** (const GPUCopyable &g2)
- **GPUCopyable** (GPUCopyable &&g2)=default
- GPUCopyable & **operator=** (const GPUCopyable &g2)
- GPUCopyable & **operator=** (GPUCopyable &&g2)=default
- T **PrekopirajNaGPU** ()

Dokumentacija klase je napravljena iz datoteka:

- GPUCopyable.h

4.2 Opis klase GPUResident< P >

Public članovi

- P **DajPodatke** ()

Protected članovi

- void **UpisiPodatke** (const P &)
- void **Unisti** ()

Dokumentacija klase je napravljena iz datoteka:

- GPUResident.h

4.3 Opis strukture ModelOBJ::Grupa

Public atributi

- int **smoothGroup**
- size_t **brojPoligona**
- std::vector< Poligon > **poligoni**
- std::shared_ptr< Materijal > **materijal**

Dokumentacija strukture je napravljena iz datoteka:

- OBJ.h

4.4 Opis klase Kamera

`#include <Kamera.h>`

Public tipovi

- enum TipProjekcije { TipProjekcije::Perspektivna, TipProjekcije::Ortografska }

Public članovi

- __device__ __host__ Kamera (TipProjekcije proj, const Vektor4 &o, const Vektor3 &d, const Vektor3 &g, const Vektor3 &i, Realan fov, int sirina, int visina)
- __device__ __host__ **Kamera** (const Kamera &)=default
- __device__ __host__ Kamera & **operator=** (const Kamera &)=default
- __device__ __host__ **Kamera** (Kamera &&)=default
- __device__ __host__ Kamera & **operator=** (Kamera &&)=default
- __device__ __host__ void PostaviDuljinu (Realan d)
- __device__ __host__ void PostaviFOV (Realan fov)
- __device__ __host__ void PostaviDimenzije (int sirina, int visina)
- __device__ __host__ void PostaviTransformaciju (const Matrica4x4 &tran)
- __device__ __host__ void KomponirajTransformaciju (const Matrica4x4 &tran)
- __device__ __host__ void PostaviTip (TipProjekcije tip)
- __device__ __host__ void GledajU (const Vektor4 &cilj, const Vektor3 &gore)
- __device__ __host__ Vektor3 Pozicija () const
- __device__ const __host__ Vektor3 & U () const
- __device__ const __host__ Vektor3 & V () const
- __device__ const __host__ Vektor3 & W () const
- __device__ __host__ Realan Duljina () const
- __device__ __host__ int Sirina () const
- __device__ __host__ int Visina () const
- __device__ __host__ Realan HFOV () const
- __device__ __host__ TipProjekcije Tip () const

4.4.1 Detaljno objašnjenje

Osnovna virtualna kamera za renderiranje.

Ova klasa podržava promjene FOV-a, žarišne duljine i općenitih transformacija pogleda virtualne kamere u 3D koordinatnom sustavu. U ovom konkretnom programu, koristi se samo radi raytracinga, ali mogla bi biti iskorištena za rasterizaciju.

Svaka kamera može koristiti ili perpsketivnu ili ortografsku projekciju, što se može odabrati prilikom konstrukcije ili kasnije slobodno izmijeniti, što utječe ili na konačne transformacije u screenspace (rasterizacija) ili jednostavno na način računanja izlaznih zraka (raytracing).

4.4.2 Dokumentacija enumeracijskih članova

4.4.2.1 TipProjekcije

```
enum Kamera::TipProjekcije  [strong]
```

Tip projekcije virtualne kamere.

Vrijednosti enumeracija

Perspektivna	Perspektivni tip projekcije (za realistične prikaze)
Ortografska	Ortografska projekcija (sve zrake su paralelne)

4.4.3 Dokumentacija konstruktora i destruktora

4.4.3.1 Kamera()

```
__device__ __host__ Kamera::Kamera (
            TipProjekcije proj,
            const Vektor4 & o,
            const Vektor3 & d,
            const Vektor3 & g,
            const Vektor3 & i,
            Realan fov,
            int sirina,
            int visina )
```

Glavni konstruktor za virtualnu kameru. Uzima sve potrebne inicijalne projektivne parametre.

Parametri

in	*proj*	Tip projekcije.

Parametri

in	*o*	Početna pozicija (ishodište viewspace-a) kamere.
in	*d,g,i*	Ortonormirani vektori desno orijentiranog virtualnog okvira kamere (desno, gore, iza).
in	*fov*	Horizontalni kut pogleda (FOV) kamere u radijanima.
in	*w,h*	Dimenzije slike (trenutno je nužno da budu identične piksel rezoluciji).

4.4.4 Dokumentacija funkcija

4.4.4.1 Duljina()

```
__device__ __host__ Realan Kamera::Duljina ( ) const
```

Vraća trenutnu žarišnu duljinu kamere.

Povratne vrijednosti

Žarišna duljina kamere u viewspace jedinicama.

4.4.4.2 GledajU()

```
__device__ __host__ void Kamera::GledajU (
            const Vektor4 & cilj,
            const Vektor3 & gore )
```

Usmjerava kameru u fiksnu točku u svjetskim koordinatama.

Parametri

in	*cilj*	Točka u svjetskom prostoru u koju kamera mora gledati.
in	*gore*	Normiran vektor usmjeren ravno iznad kamere, dakle ekvivalentan Y osi u viewspace-u.

4.4.4.3 HFOV()

```
__device__ __host__ Realan Kamera::HFOV ( ) const  [inline]
```

Vraća trenutni horizontalni kut pogleda (HFOV) kamere.

Povratne vrijednosti

HFOV kamere u radijanima.

4.4.4.4 KomponirajTransformaciju()

```
__device__ __host__ void Kamera::KomponirajTransformaciju (
            const Matrica4x4 & tran )
```

Komponira 4x4 transformaciju na kameru.

Ova metoda komponira zadanu transformaciju s trenutnom modelview matricom, što rezultira u kumulativnom efektu transformacije, korisnom za animacije kamere i sl.

Parametri

in	*tran*	4x4 transformacijska matrica.

4.4.4.5 PostaviDimenzije()

```
__device__ __host__ void Kamera::PostaviDimenzije (
            int sirina,
            int visina )
```

Dinamički mijenja dimenzije slike na zadani par cijelih brojeva.

Parametri

in	*sirina,visina*	Nove dimenzije u ekranskim pikselima.

4.4.4.6 PostaviDuljinu()

```
__device__ __host__ void Kamera::PostaviDuljinu (
            Realan d )
```

Dinamički mijenja žarišnu duljinu kamere na zadani realan broj.

Parametri

in	*d*	Nova žarišna duljina u viewspace jedinicama.

4.4.4.7 PostaviFOV()

```
__device__ __host__ void Kamera::PostaviFOV (
            Realan fov )
```

Dinamički mijenja horizontalni kut pogleda kamere na zadani realan broj.

Parametri

in	*fov*	Novi horizontalni kut pogleda u radijanima.

4.4.4.8 PostaviTip()

```
__device__ __host__ void Kamera::PostaviTip (
            TipProjekcije tip )  [inline]
```

Dinamički mijenja tip projekcije kamere.

Parametri

in	*tip*	Novi tip projekcije kamere.

4.4.4.9 PostaviTransformaciju()

```
__device__ __host__ void Kamera::PostaviTransformaciju (
            const Matrica4x4 & tran )
```

Postavlja apsolutnu 4x4 transformaciju na kameru.

Ova metoda *neće* komponirati transformacije; dakle, sve se provodi nad inicijalnim okvirom i pozicijom kamere!

Parametri

in	*tran*	4x4 transformacijska matrica.

4.4.4.10 Pozicija()

```
__device__ __host__ Vektor3 Kamera::Pozicija ( ) const
```

Vraća trenutnu poziciju kamere u svjetskom prostoru.

Povratne vrijednosti

3-vektor trenutne pozicije kamere u svjetskim koordinatama.

4.4.4.11 Sirina()

```
__device__ __host__ int Kamera::Sirina ( ) const
```

Vraća trenutnu X dimenziju slike kamere.

Povratne vrijednosti

X dimenzija (širina) slike.

4.4.4.12 Tip()

```
__device__ __host__ TipProjekcije Kamera::Tip ( ) const  [inline]
```

Vraća trenutni tip projekcije kamere.

Povratne vrijednosti

Tip projekcije kamere.

4.4.4.13 U()

```
__device__ const __host__ Vektor3 & Kamera::U ( ) const
```

Vraća trenutni vektor "desno" okvira kamere, u svjetskim koordinatama.

Povratne vrijednosti

Normiran vektor u svjetskim koordinatama koji u viewspace-u predstavlja X os.

4.4.4.14 V()

```
__device__ const __host__ Vektor3 & Kamera::V ( ) const
```

Vraća trenutni vektor "gore" okvira kamere, u svjetskim koordinatama.

Povratne vrijednosti

Normiran vektor u svjetskim koordinatama koji u viewspace-u predstavlja Y os.

4.4.4.15 Visina()

```
__device__ __host__ int Kamera::Visina ( ) const
```

Vraća trenutnu Y dimenziju slike kamere.

Povratne vrijednosti

Y dimenzija (visina) slike.

4.4.4.16 W()

```
__device__ const __host__ Vektor3 & Kamera::W ( ) const
```

Vraća trenutni vektor "iza" okvira kamere, u svjetskim koordinatama.

Povratne vrijednosti

Normiran vektor u svjetskim koordinatama koji u viewspace-u predstavlja Z os.

Dokumentacija klase je napravljena iz datoteke:

- Kamera.h
- Kamera.cpp

4.5 Opis klase Kanal8

Public članovi

- __device__ __host__ **Kanal8** (Realan broj)
- __device__ __host__ **Kanal8** (unsigned int broj=0)
- __device__ __host__ **Kanal8** (const Kanal8 &)=default
- __device__ __host__ Kanal8 & **operator=** (const Kanal8 &)=default
- __device__ __host__ **Kanal8** (Kanal8 &&)=default
- __device__ __host__ Kanal8 & **operator=** (Kanal8 &&)=default
- __device__ __host__ Kanal8 **operator+** (const Kanal8 &)
- __device__ __host__ Kanal8 **operator-** (const Kanal8 &)
- __device__ __host__ Kanal8 **operator*** (const Kanal8 &)
- __device__ __host__ Kanal8 **operator/** (const Kanal8 &)
- __device__ __host__ void **operator+=** (const Kanal8 &)
- __device__ __host__ void **operator*=** (const Kanal8 &)
- __device__ __host__ **operator Realan** ()
- __device__ __host__ **operator uint8_t** ()

Public atributi

- Realan **broj**

Friend-ovi

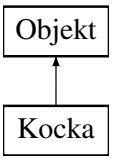
- __device__ __host__ Kanal8 **operator**∗ (Realan, Kanal8)

Dokumentacija klase je napravljena iz datoteke:

- Kanal.h
- Kanal.cpp

4.6 Opis klase Kocka

Dijagram klasa za Kocka



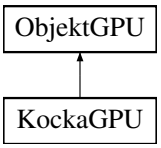
Dodatni naslijeđeni članovi

Dokumentacija klase je napravljena iz datoteke:

- Kocka.h
- Kocka.cpp
- KockaGPU.cpp

4.7 Opis klase KockaGPU

Dijagram klasa za KockaGPU



Dodatni naslijeđeni članovi

Dokumentacija klase je napravljena iz datoteka:

- KockaGPU.h

4.8 Opis klase KoordinatnaMreza

Public članovi

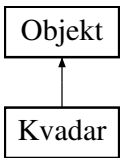
- __device__ __host__ **KoordinatnaMreza** (const Ravnina &, int, int, int, int, const Vektor3 &, const Vektor3 &, RGB boja={128, 128, 128})
- __device__ __host__ **KoordinatnaMreza** (const KoordinatnaMreza &)=default
- __device__ __host__ KoordinatnaMreza & **operator=** (const KoordinatnaMreza &)=default
- __device__ __host__ **KoordinatnaMreza** (KoordinatnaMreza &&)=default
- __device__ __host__ KoordinatnaMreza & **operator=** (const KoordinatnaMreza &&)=default
- __device__ __host__ void **Renderiraj** (const Kamera &, const Scena &)

Dokumentacija klase je napravljena iz datoteke:

- KoordinatnaMreza.h
- KoordinatnaMreza.cpp

4.9 Opis klase Kvadar

Dijagram klasa za Kvadar



Public članovi

- **Kvadar** (Realan, Realan, Realan, RGB boja=RGB(255u, 128u, 0u))
- void **Transformiraj** () override
- bool **PresijeciZraku** (const Zraka &zraka, Realan &t) override
- RGB **Sjencaj** (const std::forward_list< Svjetlo * > &) override

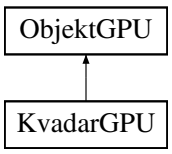
Dodatni naslijeđeni članovi

Dokumentacija klase je napravljena iz datoteke:

- Kvadar.h
- Kvadar.cpp

4.10 Opis klase KvadarGPU

Dijagram klasa za KvadarGPU



Public članovi

- __device__ __host__ **KvadarGPU** (Realan, Realan, Realan, RGB boja={255u, 128u, 0u})
- __device__ __host__ **KvadarGPU** (const KvadarGPU &)=default
- __device__ __host__ KvadarGPU & **operator=** (const KvadarGPU &)=default
- __device__ __host__ **KvadarGPU** (KvadarGPU &&)=default
- __device__ __host__ KvadarGPU & **operator=** (KvadarGPU &&)=default
- __device__ __host__ void **Transformiraj** () override
- __device__ __host__ bool **PresijeciZraku** (const Zraka &zraka, Realan &t, int &presjecenTrokut, Vektor3 &sjeciste)
- __device__ __host__ RGB **Sjencaj** (int presjecenTrokut, const Vektor3 &sjeciste, const Svjetlo[ ], size_t)

Dodatni naslijeđeni članovi

Dokumentacija klase je napravljena iz datoteke:

- KvadarGPU.h
- KvadarGPU.cpp

4.11 Opis klase MatBaza

`#include <MatBaza.h>`

Static public članovi

- static Realan Radijana (Realan stupnjeva)
- static Realan sign (Realan broj)

4.11.1 Detaljno objašnjenje

Klasa koja nudi neke osnovne matematičke funkcije koje nisu dostupne u standardnim/HIP matematičkim bibliotekama.

4.11.2 Dokumentacija funkcija

4.11.2.1 Radijana()

```
static Realan MatBaza::Radijana (
            Realan stupnjeva )  [inline], [static]
```

Pretvara dani broj stupnjeva (°) u radijane (rad).

Parametri

in	*stupnjeva*	Broj stupnjeva.

Povratne vrijednosti

Ekvivalentan broj radijana.

4.11.2.2 sign()

```
static Realan MatBaza::sign (
            Realan broj )  [inline], [static]
```

Signum (sign) funkcija za realne brojeve.

Parametri

in	*broj*	Realan broj.

Povratne vrijednosti

sign(broj)

Dokumentacija klase je napravljena iz datoteka:

- MatBaza.h

4.12 Opis strukture Materijal

Public atributi

- std::string **naziv**
- RGB **ka**
- RGB **kd** = {1.0f, 0.5f, 0.5f}
- RGB **ks**
- RGB **ns**
- Realan **prozirnost**
- bool **fresnel**
- bool **sjene**
- bool **refleksije**

Dokumentacija strukture je napravljena iz datoteka:

- Materijal.h

4.13 Opis klase Matrica3x3

`#include <Matrica3x3.h>`

Public članovi

- __device__ __host__ Matrica3x3 ()
- __device__ __host__ Matrica3x3 (Realan, Realan, Realan, Realan, Realan, Realan, Realan, Realan, Realan)
- __device__ __host__ Matrica3x3 (const Vektor3 stupci[3])
- __device__ __host__ Matrica3x3 (const Realan polje[3][3])
- __device__ __host__ **Matrica3x3** (const Matrica3x3 &)=default
- __device__ __host__ Matrica3x3 & **operator=** (const Matrica3x3 &)=default
- __device__ __host__ **Matrica3x3** (Matrica3x3 &&)=default
- __device__ __host__ Matrica3x3 & **operator=** (Matrica3x3 &&)=default
- __device__ __host__ Matrica3x3 operator+ (const Matrica3x3 &druga) const
- __device__ __host__ Matrica3x3 operator- (const Matrica3x3 &druga) const
- __device__ __host__ void operator- ()
- __device__ __host__ Realan Element (int i, int j) const
- __device__ __host__ bool operator== (const Matrica3x3 &druga) const
- __device__ __host__ bool operator!= (const Matrica3x3 &druga) const
- __device__ __host__ Matrica3x3 Transponiraj () const
- __device__ __host__ Matrica3x3 Invertiraj ()
- __device__ __host__ Realan Determinanta ()

Static public članovi

- static __device__ __host__ Matrica3x3 Jedinicna ()
- static __device__ __host__ Matrica3x3 RotirajX (Realan radijana)
- static __device__ __host__ Matrica3x3 RotirajY (Realan radijana)
- static __device__ __host__ Matrica3x3 RotirajZ (Realan radijana)

Friend-ovi

- __device__ __host__ Matrica3x3 operator∗ (const Matrica3x3 &matA, const Matrica3x3 &matB)
- __device__ __host__ Matrica3x3 operator∗ (Realan skalar, const Matrica3x3 &mat)
- __device__ __host__ Matrica3x3 operator∗ (const Matrica3x3 &mat, Realan skalar)
- __device__ __host__ Vektor3 operator∗ (const Matrica3x3 &mat, const Vektor3 &vek)

4.13.1 Detaljno objašnjenje

Tip općenite 3x3 matrice s odgovarajućim operacijama.

Osnovni tip matrice za predstavljanje bilo kojeg linearnog operatora u trodimenzionalnom vektorskom prostoru.

4.13.2 Dokumentacija konstruktora i destruktora

4.13.2.1 Matrica3x3() [1/4]

```
__device__ __host__ Matrica3x3::Matrica3x3 ( )
```

Default konstruktor stvara nulmatricu.

4.13.2.2 Matrica3x3() [2/4]

```
__device__ __host__ Matrica3x3::Matrica3x3 (
            Realan m11,
            Realan m12,
            Realan m13,
            Realan m21,
            Realan m22,
            Realan m23,
            Realan m31,
            Realan m32,
            Realan m33 )
```

Konstruktor matrice element-po-element.

Elementi se navode slijeva na desno, od najvišeg do najnižeg retka.

4.13.2.3 Matrica3x3() [3/4]

```
__device__ __host__ Matrica3x3::Matrica3x3 (
            const Vektor3 stupci[3] )
```

Stupčani konstruktor matrice.

Parametri

in	*stupci*	Polje od triju stupaca, redom slijeva u desno.

4.13.2.4 Matrica3x3() [4/4]

```
__device__ __host__ Matrica3x3::Matrica3x3 (
            const Realan polje[3][3] )  [explicit]
```

Konstruktor matrice od 2D polja.

Parametri

in	*polje*	2D polje koje prirodno odgovara matrici (row-major).

4.13.3 Dokumentacija funkcija

4.13.3.1 Determinanta()

```
__device__ __host__ Realan Matrica3x3::Determinanta ( )
```

Vraća determinantu matrice reda 3.

Povratne vrijednosti

Determinanta ove matrice.

4.13.3.2 Element()

```
__device__ __host__ Realan Matrica3x3::Element (
            int i,
            int j ) const
```

Vraća traženi element (i,j) matrice.

Indeksi počinju od 0.

Parametri

in	*i*	Indeks retka.
in	*j*	Indeks stupca.

Povratne vrijednosti

Element (i,j)

4.13.3.3 Invertiraj()

```
__device__ __host__ Matrica3x3 Matrica3x3::Invertiraj ( )
```

Invertiranje matrice reda 3.

Povratne vrijednosti

Inverz ove matrice.

4.13.3.4 Jedinicna()

```
__device__ __host__ Matrica3x3 Matrica3x3::Jedinicna ( )  [static]
```

Vraća jediničnu matricu reda 3.

4.13.3.5 operator"!=()

```
__device__ __host__ bool Matrica3x3::operator!= (
            const Matrica3x3 & druga ) const
```

Provjera nejednakosti matrica reda 3.

Parametri

in	*druga*	Druga matrica za usporedbu.

Povratne vrijednosti

`true` ako matrice nisu jednake; inače `false`.

4.13.3.6 operator+()

```
__device__ __host__ Matrica3x3 Matrica3x3::operator+ (
            const Matrica3x3 & druga ) const
```

Matrično zbrajanje reda 3.

Parametri

in	*druga*	Desni matrični operand zbrajanja.

Povratne vrijednosti

Suma trenutne (lijeve) matrice i dane desne matrice reda 3.

4.13.3.7 operator-() [1/2]

```
__device__ __host__ void Matrica3x3::operator- ( )
```

Negiranje matrice reda 3 (unarni operator).

4.13.3.8 operator-() [2/2]

```
__device__ __host__ Matrica3x3 Matrica3x3::operator- (
            const Matrica3x3 & druga ) const
```

Matrično oduzimanje reda 3.

Parametri

in	*druga*	Desni matrični operand oduzimanja.

Povratne vrijednosti

Razlika trenutne (lijeve) matrice i dane desne matrice reda 3.

4.13.3.9 operator==()

```
__device__ __host__ bool Matrica3x3::operator== (
            const Matrica3x3 & druga ) const
```

Provjera jednakosti matrica reda 3.

Parametri

in	*druga*	Druga matrica za usporedbu.

Povratne vrijednosti

`true` ako su matrice jednake; inače `false`.

4.13.3.10 RotirajX()

```
__device__ __host__ Matrica3x3 Matrica3x3::RotirajX (
            Realan radijana ) [static]
```

Vraća rotacijsku matricu po osi X.

Parametri

in	*radijana*	Kut u radijanama za rotaciju oko osi X.

4.13.3.11 RotirajY()

```
__device__ __host__ Matrica3x3 Matrica3x3::RotirajY (
            Realan radijana )  [static]
```

Vraća rotacijsku matricu po osi Y.

Parametri

in	*radijana*	Kut u radijanama za rotaciju oko osi Y.

4.13.3.12 RotirajZ()

```
__device__ __host__ Matrica3x3 Matrica3x3::RotirajZ (
            Realan radijana )  [static]
```

Vraća rotacijsku matricu po osi Z.

Parametri

in	*radijana*	Kut u radijanama za rotaciju oko osi Z.

4.13.3.13 Transponiraj()

```
__device__ __host__ Matrica3x3 Matrica3x3::Transponiraj ( ) const
```

Transponiranje matrice reda 3.

Povratne vrijednosti

Transponirana matrica.

4.13.4 Dokumentacija povezanih funkcija

4.13.4.1 operator* [1/4]

```
__device__ __host__ Vektor3 operator* (
            const Matrica3x3 & mat,
            const Vektor3 & vek )  [friend]
```

Matrično-vektorsko množenje reda 3.

Parametri

in	*mat*	Lijevi matrični operand.
in	*vek*	Desni vektorski operand.

Povratne vrijednosti

Rezultantni 3-vektor.

4.13.4.2 operator* [2/4]

```
__device__ __host__ Matrica3x3 operator* (
            const Matrica3x3 & mat,
            Realan skalar )  [friend]
```

Skalar-matrično množenje s matricama reda 3.

Parametri

in	*skalar*	Desni skalarni operand.
in	*mat*	Lijevi matrični operand.

Povratne vrijednosti

3x3 rezultantna matrica.

4.13.4.3 operator* [3/4]

```
__device__ __host__ Matrica3x3 operator* (
            const Matrica3x3 & matA,
            const Matrica3x3 & matB )  [friend]
```

Množenje matrica reda 3.

Parametri

in	*matA*	Lijevi matrični operand.
in	*matB*	Desni matrični operand.

Povratne vrijednosti

3x3 matrica produkta.

4.13.4.4 operator* [4/4]

```
__device__ __host__ Matrica3x3 operator* (
            Realan skalar,
            const Matrica3x3 & mat )  [friend]
```

Skalar-matrično množenje s matricama reda 3.

Parametri

in	*skalar*	Lijevi skalarni operand.
in	*mat*	Desni matrični operand.

Povratne vrijednosti

3x3 rezultantna matrica.

Dokumentacija klase je napravljena iz datoteke:

- Matrica3x3.h
- Matrica3x3.cpp

4.14 Opis klase Matrica4x4

```
#include <Matrica4x4.h>
```

Public članovi

- __device__ __host__ Matrica4x4 ()
- __device__ __host__ Matrica4x4 (Realan, Realan, Realan, Realan, Realan, Realan, Realan, Realan, Realan, Realan, Realan, Realan, Realan, Realan, Realan, Realan)
- __device__ __host__ Matrica4x4 (const Realan[4][4])
- __device__ __host__ **Matrica4x4** (const Matrica4x4 &)=default
- __device__ __host__ Matrica4x4 & **operator=** (const Matrica4x4 &)=default
- __device__ __host__ **Matrica4x4** (Matrica4x4 &&)=default
- __device__ __host__ Matrica4x4 & **operator=** (Matrica4x4 &&)=default
- __device__ __host__ Matrica4x4 operator+ (const Matrica4x4 &) const
- __device__ __host__ Matrica4x4 operator- (const Matrica4x4 &) const
- __device__ __host__ void operator- ()
- __device__ __host__ Realan Element (int, int) const
- __device__ __host__ bool operator== (const Matrica4x4 &) const
- __device__ __host__ bool operator!= (const Matrica4x4 &) const
- __device__ __host__ Matrica4x4 Transponiraj () const
- __device__ __host__ Matrica4x4 Invertiraj ()
- __device__ __host__ Realan Determinanta ()
- __device__ __host__ Matrica3x3 Linearna () const

Static public članovi

- static __device__ __host__ Matrica4x4 Jedinicna ()
- static __device__ __host__ Matrica4x4 Skaliraj (Realan uniformSkala)
- static __device__ __host__ Matrica4x4 **RotirajX** (Realan radijana)
- static __device__ __host__ Matrica4x4 **RotirajY** (Realan radijana)
- static __device__ __host__ Matrica4x4 **RotirajZ** (Realan radijana)
- static __device__ __host__ Matrica4x4 Translatiraj (const Vektor3 &vektor)

Friend-ovi

- __device__ __host__ Matrica4x4 operator∗ (const Matrica4x4 &matA, const Matrica4x4 &matB)
- __device__ __host__ Matrica4x4 operator∗ (Realan skalar, const Matrica4x4 &mat)
- __device__ __host__ Matrica4x4 operator∗ (const Matrica4x4 &mat, Realan skalar)
- __device__ __host__ Vektor4 operator∗ (const Matrica4x4 &mat, const Vektor4 &vek)

4.14.1 Detaljno objašnjenje

Tip općenite 4x4 matrice s odgovarajućim operacijama.

Osnovni tip matrice za predstavljanje bilo kojeg afinog operatora (moglo bi se proširiti i na druge projektivne transformacije) u trodimenzionalnom afinom prostoru.

4.14.2 Dokumentacija konstruktora i destruktora

4.14.2.1 Matrica4x4() [1/3]

```
__device__ __host__ Matrica4x4::Matrica4x4 ( )
```

Default konstruktor stvara nulmatricu.

4.14.2.2 Matrica4x4() [2/3]

```
__device__ __host__ Matrica4x4::Matrica4x4 (
            Realan m11,
            Realan m12,
            Realan m13,
            Realan m14,
            Realan m21,
            Realan m22,
            Realan m23,
            Realan m24,
            Realan m31,
            Realan m32,
            Realan m33,
            Realan m34,
            Realan m41,
            Realan m42,
            Realan m43,
            Realan m44 )
```

Konstruktor matrice element-po-element.

Elementi se navode slijeva na desno, od najvišeg do najnižeg retka.

4.14.2.3 Matrica4x4() [3/3]

```
__device__ __host__ Matrica4x4::Matrica4x4 (
            const Realan elementi[4][4] )  [explicit]
```

Konstruktor matrice od 2D polja.

Parametri

in	*polje*	2D polje koje prirodno odgovara matrici (row-major).

4.14.3 Dokumentacija funkcija

4.14.3.1 Determinanta()

```
__device__ __host__ Realan Matrica4x4::Determinanta ( )
```

Vraća determinantu matrice reda 4.

Povratne vrijednosti

Determinanta ove matrice.

4.14.3.2 Element()

```
__device__ __host__ Realan Matrica4x4::Element (
            int i,
            int j ) const
```

Vraća traženi element (i,j) matrice.

Indeksi počinju od 0.

Parametri

in	*i*	Indeks retka.
in	*j*	Indeks stupca.

Povratne vrijednosti

Element (i,j)

4.14.3.3 Invertiraj()

```
__device__ __host__ Matrica4x4 Matrica4x4::Invertiraj ( )
```

Invertiranje matrice reda 4.

Povratne vrijednosti

Inverz ove matrice.

4.14.3.4 Jedinicna()

```
__device__ __host__ Matrica4x4 Matrica4x4::Jedinicna ( )  [static]
```

Vraća jediničnu matricu reda 4.

4.14.3.5 Linearna()

```
__device__ __host__ Matrica3x3 Matrica4x4::Linearna ( ) const
```

Vraća odgovarajući linearnu transformaciju.

Vraća podmatricu reda 3 koja predstavlja linearni operator jedinstveno dobiven zanemarivanjem translacije trenutne matrice.

Povratne vrijednosti

Matrica reda 3 odgovarajućeg linearnog operatora.

4.14.3.6 operator"!=()

```
__device__ __host__ bool Matrica4x4::operator!= (
            const Matrica4x4 & matB ) const
```

Provjera nejednakosti matrica reda 4.

Parametri

in	*druga*	Druga matrica za usporedbu.

Povratne vrijednosti

`true` ako matrice nisu jednake; inače `false`.

4.14.3.7 operator+()

```
__device__ __host__ Matrica4x4 Matrica4x4::operator+ (
            const Matrica4x4 & mat ) const
```

Matrično zbrajanje reda 4.

Parametri

in	*druga*	Desni matrični operand zbrajanja.

Povratne vrijednosti

Suma trenutne (lijeve) matrice i dane desne matrice reda 4.

4.14.3.8 operator-() [1/2]

```
__device__ __host__ void Matrica4x4::operator- ( )
```

Negiranje matrice reda 4 (unarni operator).

4.14.3.9 operator-() [2/2]

```
__device__ __host__ Matrica4x4 Matrica4x4::operator- (
            const Matrica4x4 & mat ) const
```

Matrično oduzimanje reda 4.

Parametri

in	*druga*	Desni matrični operand oduzimanja.

Povratne vrijednosti

Razlika trenutne (lijeve) matrice i dane desne matrice reda 4.

4.14.3.10 operator==()

```
__device__ __host__ bool Matrica4x4::operator== (
            const Matrica4x4 & matB ) const
```

Provjera jednakosti matrica reda 4.

Parametri

in	*druga*	Druga matrica za usporedbu.

Povratne vrijednosti

`true` ako su matrice jednake; inače `false`.

4.14.3.11 Skaliraj()

```
__device__ __host__ Matrica4x4 Matrica4x4::Skaliraj (
            Realan uniformSkala )  [static]
```

Vraća transformaciju koja uniformno skalira po svim osima.

Parametri

in	*uniformSkala*	Skalar za uniformno skaliranje.

4.14.3.12 Translatiraj()

```
__device__ __host__ Matrica4x4 Matrica4x4::Translatiraj (
            const Vektor3 & vektor )  [static]
```

Vraća translacijsku matricu.

Parametri

in	*vektor*	Vektor pomaka za translaciju.

4.14.3.13 Transponiraj()

```
__device__ __host__ Matrica4x4 Matrica4x4::Transponiraj ( ) const
```

Transponiranje matrice reda 4.

Povratne vrijednosti

Transponirana matrica.

4.14.4 Dokumentacija povezanih funkcija

4.14.4.1 operator* [1/4]

```
__device__ __host__ Vektor4 operator* (
            const Matrica4x4 & mat,
            const Vektor4 & vek )  [friend]
```

Matrično-vektorsko množenje reda 4.

Parametri

in	*mat*	Lijevi matrični operand.
in	*vek*	Desni vektorski operand.

Povratne vrijednosti

Rezultantni 4-vektor.

4.14.4.2 operator* [2/4]

```
__device__ __host__ Matrica4x4 operator* (
            const Matrica4x4 & mat,
            Realan skalar )  [friend]
```

Skalar-matrično množenje s matricama reda 4.

Parametri

in	*skalar*	Desni skalarni operand.
in	*mat*	Lijevi matrični operand.

Povratne vrijednosti

4x4 rezultantna matrica.

4.14.4.3 operator* [3/4]

```
__device__ __host__ Matrica4x4 operator* (
            const Matrica4x4 & matA,
            const Matrica4x4 & matB )  [friend]
```

Množenje matrica reda 4.

Parametri

in	*matA*	Lijevi matrični operand.
in	*matB*	Desni matrični operand.

Povratne vrijednosti

4x4 matrica produkta.

4.14.4.4 operator* [4/4]

```
__device__ __host__ Matrica4x4 operator* (
            Realan skalar,
            const Matrica4x4 & mat )  [friend]
```

Skalar-matrično množenje s matricama reda 4.

Parametri

in	*skalar*	Lijevi skalarni operand.
in	*mat*	Desni matrični operand.

Povratne vrijednosti

4x4 rezultantna matrica.

Dokumentacija klase je napravljena iz datoteke:

- Matrica4x4.h
- Matrica4x4.cpp

4.15 Opis klase ModelOBJ

Strukture

- struct Grupa
- struct Objekt
- struct Poligon

Public članovi

- **ModelOBJ** (const std::string &datoteka)
- const std::vector< Objekt > & **DajObjekte** () const

Dokumentacija klase je napravljena iz datoteke:

- OBJ.h
- OBJ.cpp

4.16 Opis strukture ModelOBJ::Objekt

Public atributi

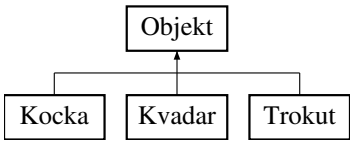
- std::string **naziv**
- size_t **brojGrupa**
- size_t **brojVrhova**
- size_t **brojUV**
- size_t **brojNormala**
- std::vector< Vektor4 > **vrhovi**
- std::vector< Vektor3 > **normale**
- std::vector< Vektor3 > **UV**
- std::unordered_map< std::string, Grupa > **grupe**

Dokumentacija strukture je napravljena iz datoteka:

- OBJ.h

4.17 Opis klase Objekt

Dijagram klasa za Objekt



Public članovi

- virtual void **Transformiraj** ()=0
- virtual RGB **Sjencaj** (const std::forward_list< Svjetlo * > &svjetla)=0
- virtual bool **PresijeciZraku** (const Zraka &zraka, Realan &t)=0

Public atributi

- Matrica4x4 **transformacija** = Matrica4x4::Jedinicna()

Protected atributi

- Vektor3 **sjeciste**

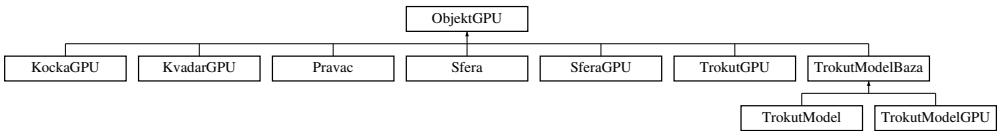
Dokumentacija klase je napravljena iz datoteka:

- Objekt.h

4.18 Opis klase ObjektGPU

`#include <ObjektGPU.h>`

Dijagram klasa za ObjektGPU



Public članovi

- __device__ __host__ void Transformiraj (int)
- __device__ __host__ int DajBrojNitiZaTransform () const
- __device__ __host__ **ObjektGPU** (const ObjektGPU &)=default
- __device__ __host__ ObjektGPU & **operator=** (const ObjektGPU &)=default
- __device__ __host__ **ObjektGPU** (ObjektGPU &&)=default
- __device__ __host__ ObjektGPU & **operator=** (ObjektGPU &&)=default
- __device__ __host__ RGB Sjencaj (const Vektor3 &sjeciste, const Svjetlo svjetla[ ], size_t n)
- __device__ __host__ bool PresijeciZraku (const Zraka &zraka, Realan &t, Vektor3 &sjeciste)

Public atributi

- Matrica4x4 transformacija = Matrica4x4::Jedinicna()

4.18.1 Detaljno objašnjenje

Osnovna klasa za sve primitivne grafičke objekte koji se mogu direktno renderirati.

Ova klasa bi trebala biti čisto apstraktna, ali budući da ovaj objekt mora doći do GPU-a i tamo izvoditi metode nad raznim objektima, ne možemo imati virtualne metode jer ih HIP trenutno ne podržava.

4.18.2 Dokumentacija funkcija

4.18.2.1 DajBrojNitiZaTransform()

```
__device__ __host__ int ObjektGPU::DajBrojNitiZaTransform ( ) const  [inline]
```

Vraća broj niti potreban za 100% paralelnu transformaciju objekta (tj. svih njegovih elemenata) na GPU-u.

4.18.2.2 PresijeciZraku()

```
__device__ __host__ bool ObjektGPU::PresijeciZraku (
            const Zraka & zraka,
            Realan & t,
            Vektor3 & sjeciste )  [inline]
```

Apstraktna metoda za presjecanje zrake i objekta u raytracingu.

Podklase nadjačavaju ovu metodu s točnim (što optimalnijim) algoritmom za presjek zrake i konkretnog geometrijskog objekta (ili jednostavnijeg proxy-a ako je to dovoljno dobro).

Parametri

in	*zraka*	Ulazna zraka.
out	*t*	Parametar t u jednadžbi zrake koji odgovara pronađenom sjecištu.
out	*sjeciste*	Pronađeno sjecište; podklase ovo mogu koristiti na različite načine.

Povratne vrijednosti

Vraća `true` ako zraka siječe objekte, a `false` inače.

4.18.2.3 Sjencaj()

```
__device__ __host__ RGB ObjektGPU::Sjencaj (
            const Vektor3 & sjeciste,
            const Svjetlo svjetla[],
            size_t n )  [inline]
```

Apstraktna metoda za sjenčanje (shading) objekta u raytracingu.

Podklase će nadjačati (specijalizirati) ovu metodu sa željenim shaderom. U trenutnoj implementaciji, neke primitive mogu odstupiti od ovog prototipa, pa bi trebalo refaktorirati s proširenjem koje dopušta da se bilo koji `int` indeks može vratiti preko reference u ovoj metodi. Na taj način bi se moglo direktno podržati složenije primitive poput indeksiranih mesheva koje zahtijevaju informaciju o točnom trokutu koji je presječen od strane neke zrake.

Parametri

in	*sjeciste*	Izračunato sjecište objekta i zrake.
in	*svjetla*	Polje svjetala na sceni koja utječu na prikaz objekata.
in	*n*	Broj svjetala u polju.

Povratne vrijednosti

RGB vrijednost koji predstavlja konačnu boju piksela koji odgovara trenutnoj zraci.

4.18.2.4 Transformiraj()

```
__device__ __host__ void ObjektGPU::Transformiraj (
            int  )  [inline]
```

Primjenjuje transformaciju zadanu matricom na objekt.

4.18.3 Documentacija varijabli

4.18.3.1 transformacija

```
Matrica4x4 ObjektGPU::transformacija = Matrica4x4::Jedinicna()
```

Transformacija iz model prostora u svjetski prostor.

Dokumentacija klase je napravljena iz datoteka:

- ObjektGPU.h

4.19 Opis strukture TrokutModelBaza::Podaci

```
#include <TrokutModelBaza.h>
```

Public atributi

- size_t **brojVrhova** = 0
- size_t **brojNormala** = 0
- size_t **brojUV** = 0
- size_t **brojPoligona** = 0
- Vektor3 * **vrhovi** = nullptr
- Vektor3 * **normale** = nullptr
- Vektor3 * **UV** = nullptr
- Poligon * **poligoni** = nullptr
- Vektor3 * **transformiraniVrhovi** = nullptr
- Vektor3 * **transformiraneNormale** = nullptr
- Matrica4x4 **transformacija**

4.19.1 Detaljno objašnjenje

Struktura Osnovnih podataka o meshu.

Dokumentacija strukture je napravljena iz datoteka:

- TrokutModelBaza.h

4.20 Opis strukture ModelOBJ::Poligon

Public atributi

- int **a**
- int **ta**
- int **na**
- int **b**
- int **tb**
- int **nb**
- int **c**
- int **tc**
- int **nc**

Dokumentacija strukture je napravljena iz datoteka:

- OBJ.h

4.21 Opis strukture TrokutModelBaza::Poligon

`#include <TrokutModelBaza.h>`

Public atributi

- int **a**
- int **b**
- int **c**
- int **na**
- int **nb**
- int **nc**
- int **ta**
- int **tb**
- int **tc**

4.21.1 Detaljno objašnjenje

Osnovna struktura za indeksirane poligone.

Postoji opcija uključivanja i teksturnih koordinata i vršnih normala.

Dokumentacija strukture je napravljena iz datoteka:

- TrokutModelBaza.h

4.22 Opis strukture Postavke

Public atributi

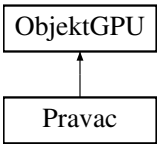
- int **w**
- int **h**
- Realan **fov**
- std::vector< std::string > **modelFajlovi**

Dokumentacija strukture je napravljena iz datoteka:

- main_hip.cpp

4.23 Opis klase Pravac

Dijagram klasa za Pravac



Public članovi

- __device__ __host__ **Pravac** (const Vektor4 &, const Vektor3 &, RGB boja={255u, 255u, 255u})
- __device__ __host__ **Pravac** (const Vektor4 &, const Vektor4 &, RGB boja={255u, 255u, 255u})
- __device__ __host__ **Pravac** (const Pravac &)=default
- __device__ __host__ Pravac & **operator=** (const Pravac &)=default
- __device__ __host__ **Pravac** (Pravac &&)=default
- __device__ __host__ Pravac & **operator=** (Pravac &&)=default
- __device__ __host__ void **Transformiraj** () override
- __device__ bool **PresijeciZraku** (const Zraka &zraka, Realan &t) override
- __device__ RGB **Sjencaj** (const Svjetlo svjetla[ ], size_t) override

Dodatni naslijeđeni članovi

Dokumentacija klase je napravljena iz datoteke:

- Pravac.h
- Pravac.cpp

4.24 Opis klase Prozor

Public članovi

- **Prozor** (const SDLIntegracija &, std::string naslov, int sirina, int visina)
- void **PromjeniVelicinu** (int sirina, int visina)
- void **Unisti** ()
- void **Vsync** (Uint32 *pikseli)
- void **PoveziIzlaz** (std::function< bool()> izlazHandler)
- void **PoveziPromjenu** (std::function< void(std::pair< int, int >)> promjenaHandler)
- void **PoveziTipke** (std::function< void(SDL_Keycode)> tipkaHandler)
- bool **operator==** (const Prozor &prozor2)
- int **DajSirinu** () const
- int **DajVisinu** () const
- std::string **DajNaslov** () const
- SDL_Window * **DajProzor** () const

Dokumentacija klase je napravljena iz datoteke:

- Prozor.h
- Prozor.cpp

4.25 Opis klase Ravnina

Public članovi

- __device__ __host__ **Ravnina** (const Vektor3 &, Realan)
- __device__ __host__ **Ravnina** (const Ravnina &)=default
- __device__ __host__ Ravnina & **operator=** (const Ravnina &)=default
- __device__ __host__ **Ravnina** (Ravnina &&)=default
- __device__ __host__ Ravnina & **operator=** (const Ravnina &&)=default

Public atributi

- Vektor3 **normala**
- Realan **udaljenost**

Dokumentacija klase je napravljena iz datoteke:

- Ravnina.h
- Ravnina.cpp

4.26 Opis klase RGB

`#include <RGB.h>`

Public članovi

- __device__ __host__ **RGB** (Kanal8 r, Kanal8 g, Kanal8 b)
- __device__ __host__ **RGB** (const RGB &)=default
- __device__ __host__ RGB & **operator=** (const RGB &)=default
- __device__ __host__ **RGB** (RGB &&)=default
- __device__ __host__ RGB & **operator=** (RGB &&)=default
- __device__ __host__ RGB operator∗ (const RGB &rgb2)
- __device__ __host__ Kanal8 operator[ ] (int i) const
- __device__ __host__ Kanal8 & **operator[ ]** (int idx)
- __device__ __host__ operator Uint32 ()
- __device__ __host__ Kanal8 **dajR** () const
- __device__ __host__ Kanal8 **dajG** () const
- __device__ __host__ Kanal8 **dajB** () const
- __device__ __host__ void **postaviR** (Kanal8 r)
- __device__ __host__ void **postaviG** (Kanal8 g)
- __device__ __host__ void **postaviB** (Kanal8 b)

4.26.1 Detaljno objašnjenje

Tip koji predstavlja općenitu RGB boju.

Interno, boje se predstavljaju kao realan broj, ali na kraju se uvijek koriste u cjelobrojnom obliku unutar fiksnog raspona. Za standardan dinamički raspon (trenutno jedini podržan), taj raspon je $[0, 255]$ i on se koristi pri računanju konačne boje pojedinačnih piksela slike za framebuffer ili nešto sl.

4.26.2 Dokumentacija funkcija

4.26.2.1 operator Uint32()

```
__device__ __host__ RGB::operator Uint32 ( )  [inline]
```

Pretvara RGB boju u jedan 32-bitni broj.

Slijedom SDL-ovog standardnog formata RGB piksela, crveni kanal ide na najniži bajt.

4.26.2.2 operator∗()

```
__device__ __host__ RGB RGB::operator* (
            const RGB & rgb2 )  [inline]
```

Množi RGB boju sa drugom danom bojom.

Ovo se svodi na jednostavno komponentno množenje i ima određenu (približno) fizičku interpretaciju u različitim modelima sjenčanja (npr. Cook-Torrance, Phong itd.).

Parametri

in	*rgb2*	Druga boja.

Povratne vrijednosti

Produktna boja.

4.26.2.3 operator[]()

```
__device__ __host__ Kanal8 RGB::operator[] (
            int i ) const  [inline]
```

Daje i-tu komponentu boje.

$i \in \{0, 1, 2\}$

Parametri

in	*i*	Indeks.

Povratne vrijednosti

Odgovarajuća komponenta (kanal) boje.

Dokumentacija klase je napravljena iz datoteka:

- RGB.h

4.27 Opis klase Scena

`#include <Scena.h>`

Public članovi

- void **TransformirajObjekte** ()
- std::forward_list< Svjetlo > & **DajSvjetla** ()
- std::forward_list< SferaGPU > & **DajSfere** ()
- std::forward_list< TrokutModel > & **DajModele** ()
- std::forward_list< TrokutGPU > & **DajTrokute** ()
- int **DajBrojSvjetala** () const
- int **DajBrojSfera** () const
- int **DajBrojModela** () const
- int **DajBrojTrokuta** () const
- void **DodajSvjetlo** (const Svjetlo &)
- void **DodajSferu** (const SferaGPU &)
- int **DodajModel** (TrokutModel &&)
- void **DodajTrokut** (const TrokutGPU &)

4.27.1 Detaljno objašnjenje

Osnovna klasa za scenu tj. kolekciju svih objekata za renderirati.

Korisnik bi uvijek trebao raditi samo s objektima ovog tipa kada želi nešto renderirati i smjestiti objekte. Instance drže svoje podatke isključivo u host memoriji; za rad s GPU-om je namijenjena posebna klasa ScenaGPU koja može instancirati svoje objekte koristeći upravo instance ove klase.

Za CPU raytracing, potrebno je direktno koristiti objekte ove klase i prilagođene rutine za raytracing; to bi trebalo unificirati u jednu zajedničku strukturu, no budući da na device kodu ne možemo koristiti proizvoljne STL strukture, taj pristup je dosta ograničavajuć i spor za razvoj pa je zasad ovo riješeno duplikacijom klasa ovisno jesu li namijenjene specifično CPU ili specifično GPU renderiranju.

Dokumentacija klase je napravljena iz datoteke:

- Scena.h
- Scena.cpp

4.28 Opis klase ScenaGPU

`#include <ScenaGPU.h>`

Public članovi

- __host__ ScenaGPU (Scena &)
- __host__ **ScenaGPU** (const ScenaGPU &)=default
- __host__ ScenaGPU & **operator=** (const ScenaGPU &&)
- __device__ __host__ SferaGPU * **DajSfere** () const
- __device__ __host__ TrokutModelBaza::Podaci * **DajModele** () const
- __device__ __host__ TrokutGPU * **DajTrokute** () const
- __device__ __host__ Svjetlo * **DajSvjetla** () const
- __host__ ScenaPodaci & **DajPodatke** ()
- __device__ __host__ int **DajBrojSfera** () const
- __device__ __host__ int **DajBrojModela** () const
- __device__ __host__ int **DajBrojTrokuta** () const
- __device__ __host__ int **DajBrojSvjetala** () const
- __device__ __host__ size_t **DajBrojTransformNiti** () const
- __device__ __host__ size_t **DajBrojSferaTransformNiti** () const
- __device__ __host__ size_t **DajBrojTrokutTransformNiti** () const
- __device__ __host__ size_t **DajBrojModelTransformNiti** () const

Static public članovi

- static __device__ __host__ void **TransformirajObjekte** (ScenaPodaci &podaci)

4.28.1 Detaljno objašnjenje

Klasa za scenu na GPU-u.

Za sada se scena drukčije reprezentira na GPU-u od CPU reprezentacije. Osnovna ideja je ista kao u klasi Scena.

4.28.2 Dokumentacija konstruktora i destruktora

4.28.2.1 ScenaGPU()

```
__host__ ScenaGPU::ScenaGPU (
            Scena & scena )
```

TODO: prebaci sve ovo u jedan memorijski bazen!

Dokumentacija klase je napravljena iz datoteke:

- ScenaGPU.h
- ScenaGPU.cpp

4.29 Opis strukture ScenaPodaci

```
#include <ScenaGPU.h>
```

Public atributi

- int **brojSvjetala** = 0
- int **brojSfera** = 0
- int **brojTrokutModela** = 0
- int **brojTrokuta** = 0
- Svjetlo * **svjetla** = nullptr
- SferaGPU * **sfere** = nullptr
- TrokutModelBaza::Podaci * **modeli** = nullptr
- TrokutGPU * **trokuti** = nullptr

4.29.1 Detaljno objašnjenje

Struktura za podatke o GPU sceni.

U ovoj strukturi su sve informacije potrebne za predstaviti scenu GPU primitiva. Bilo kakva lista se predstavlja poljem koje host dinamički alocira u GPU globalnoj memoriji. Dakle, svi pokazivači ovdje pokazuju u globalnu GPU memoriju!

Dokumentacija strukture je napravljena iz datoteka:

- ScenaGPU.h

4.30 Opis klase SDLIntegracija

Public tipovi

- typedef SDL_Window * **ProzorID**

Public članovi

- **SDLIntegracija** (const SDLIntegracija &)=delete
- void **operator=** (const SDLIntegracija &)=delete

Static public članovi

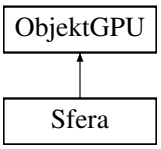
- static SDLIntegracija & **DajInstancu** (Uint32 opcije=0)

Dokumentacija klase je napravljena iz datoteke:

- SDLIntegracija.h
- SDLIntegracija.cpp

4.31 Opis klase Sfera

Dijagram klasa za Sfera



Public članovi

- **Sfera** (Realan, RGB)
- **Sfera** (Realan)
- void **Transformiraj** () override
- bool PresijeciZraku (const Zraka &zraka, Realan &t) override
- RGB **Sjencaj** (const std::forward_list< Svjetlo * > &) override

Dodatni naslijeđeni članovi

4.31.1 Dokumentacija funkcija

4.31.1.1 PresijeciZraku()

```
bool Sfera::PresijeciZraku (
            const Zraka & zraka,
            Realan & t )  [override]
```

double t1 = -(d * u + sign(d * u) * sqrt(D)) / (d * d);

double t2;

if (fabs(D) < 1.0e-6)

t2 = t1;

else t2 = (u * u - R * R) / ((d * d) * t1);

sjeciste = e + d * t2; t = t2;

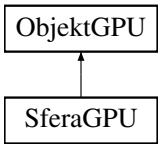
Dokumentacija klase je napravljena iz datoteke:

- Sfera.h
- Sfera.cpp

4.32 Opis klase SferaGPU

`#include <SferaGPU.h>`

Dijagram klasa za SferaGPU



Public članovi

- __device__ __host__ SferaGPU (Realan r, RGB rgb)
- __device__ __host__ SferaGPU (Realan r)
- __device__ __host__ **SferaGPU** (const SferaGPU &)=default
- __device__ __host__ SferaGPU & **operator=** (const SferaGPU &)=default
- __device__ __host__ **SferaGPU** (SferaGPU &&)=default
- __device__ __host__ SferaGPU & **operator=** (SferaGPU &&)=default
- __device__ __host__ void **Transformiraj** (int) override
- __device__ __host__ int DajBrojNitiZaTransform () const override
- __device__ __host__ bool **PresijeciZraku** (const Zraka &zraka, Realan &t, Vektor3 &sjeciste) override
- __device__ __host__ RGB **Sjencaj** (const Vektor3 &sjeciste, const Svjetlo[ ], size_t) override
- __host__ RGB Sjencaj (const Vektor3 &sjeciste, const std::forward_list< Svjetlo > &svjetla)

Dodatni naslijeđeni članovi

4.32.1 Detaljno objašnjenje

Tip grafičke primitive za raytracing koji predstavlja sferu.

Ova primitiva je vrlo jednostavna i ne zahtijeva nikakvu dodatnu dinamičku alokaciju memorije, pa se ista klasa koristi i za GPU i CPU raytracing.

4.32.2 Dokumentacija konstruktora i destruktora

4.32.2.1 SferaGPU() [1/2]

```
__device__ SferaGPU::SferaGPU (
            Realan r,
            RGB rgb )
```

Konstruktor sfere s radijusom i bojom.

Parametri

in	*r*	Radijus sfere.
in	*rgb*	Površinska boja sfere.

4.32.2.2 SferaGPU() [2/2]

```
__device__ SferaGPU::SferaGPU (
            Realan r )
```

Konstruktor sfere s radijusom.

Parametri

in	*r*	Radijus sfere.

4.32.3 Dokumentacija funkcija

4.32.3.1 DajBrojNitiZaTransform()

```
__device__ __host__ int SferaGPU::DajBrojNitiZaTransform ( ) const  [inline], [override]
```

Vraća broj niti za transformaciju sfere: 1.

Sfere se ne moraju posebno transformirati, zato je ovo nepotrebna funkcija. U računu sjecišta zrake i sfere, uzima se u obzir točno središte sfere direktno iz transformacijske matrice.

4.32.3.2 Sjencaj()

```
__host__ RGB SferaGPU::Sjencaj (
            const Vektor3 & sjeciste,
            const std::forward_list< Svjetlo > & svjetla )
```

Sjenčanje za CPU raytracing.

Parametri

in	*sjeciste*	Izračunato sjecište u svjetskom prostoru.
in	*svjetla*	Lista svjetala na sceni.

Povratne vrijednosti

RGB boja odgovarajućeg piksela.

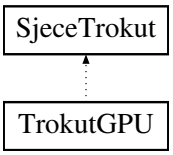
Dokumentacija klase je napravljena iz datoteke:

- SferaGPU.h
- SferaGPU.cpp

4.33 Opis klase SjeceTrokut

`#include <SjeceTrokut.h>`

Dijagram klasa za SjeceTrokut



Public članovi

- __host__ __device__ bool **PresijeciZrakuTrokut** (const Zraka &zraka, Realan &t, Vektor3 &bariSjeciste) const

Protected atributi

- Vektor3 **aTransformiran**
- Vektor3 **bTransformiran**
- Vektor3 **cTransformiran**

4.33.1 Detaljno objašnjenje

Mixin tip klase za generičnu funkcionalnost sječe trokuta.

Ideja je da pojedine primitive koje imaju potrebu sijeći zraku s trokutom naslijede ovu klasu. Ovdje je već implementiran osnovni algoritam koji koristi pojedinačan Trokut, ali moglo bi se koristiti i drugdje. Trenutno je beskorisno jer TrokutModel koristi svoju implementaciju; TODO: refaktoriraj to i napravi da ove metode rade čisto funkcionalno i neka budu statičke (stateless; ništa od međurezultata čuvati u klasi jer to dovodi do data raceova u GPU raytracingu).

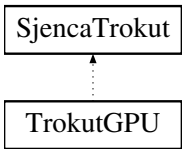
Dokumentacija klase je napravljena iz datoteka:

- SjeceTrokut.h

4.34 Opis klase SjencaTrokut

`#include <SjencaTrokut.h>`

Dijagram klasa za SjencaTrokut



Public članovi

- __host__ __device__ RGB **SjencajTrokut** (const Vektor3 &sjeciste, const Svjetlo svjetla[ ], size_t n) const

Protected atributi

- Vektor3 **transformiranaNormala**
- RGB **kd** = {1.0f, 0.5f, 0.5f}

4.34.1 Detaljno objašnjenje

Mixin klasa za osnovno sjenčanje trokuta.

Ova klasa nudi funkcionalnost sjenčanja trokuta bez interpolacije normala, dakle za najjednostavnije pojedinačne trokute koji imaju samo jednu definiranu normalu. Za složenije mesheve će najčešće biti potrebno raditi dodatne interpolacije i imati više ulaznih informacija, pa je i ovo na kraju u principu beskorisno jer ga koristi samo TrokutGPU.

Dokumentacija klase je napravljena iz datoteka:

- SjencaTrokut.h

4.35 Opis klase Slika

`#include <Slika.h>`

Public članovi

- Slika (int sirina, int visina)
- **Slika** (const SlikaGPU &)
- **Slika** (Slika &)=delete
- Slika & **operator=** (const SlikaGPU &)
- Slika & **operator=** (Slika &)=delete
- **Slika** (Slika &&)=delete
- Slika & **operator=** (Slika &&)=delete
- Uint32 * DajSliku () const
- __host__ void Raytrace (Kamera &kamera, Scena &scena)

Public atributi

- Uint32 **pozadinskaBoja** =0

4.35.1 Detaljno objašnjenje

Glavna klasa za host framebuffer i renderiranje.

Ova klasa je namijenjena za konačno prenošenje renderirane slike s GPU-a na host i konačno prikazvianje unutar prozora. Također, ako se uopće ne koristi GPU već samo CPU za renderiranje, može biti korištena samo ova klasa, bez SlikaGPU.

4.35.2 Dokumentacija konstruktora i destruktora

4.35.2.1 Slika()

```
Slika::Slika (
            int sirina,
            int visina )
```

Konstruktor slike s dimenzijama.

4.35.3 Dokumentacija funkcija

4.35.3.1 DajSliku()

```
Uint32 * Slika::DajSliku ( ) const
```

Vraća pokazivač na framebuffer.

4.35.3.2 Raytrace()

```
__host__ void Slika::Raytrace (
            Kamera & kamera,
            Scena & scena )
```

Raytrace na CPU-u.

Renderira scenu s mjesta kamere tehnikom raytracinga na CPU-u.

Parametri

in	*kamera*	Kamera koju treba koristiti u renderiranju.
in	*scena*	Scena za renderirati.

Dokumentacija klase je napravljena iz datoteke:

- Slika.h
- Slika.cpp

4.36 Opis klase SlikaGPU

`#include <SlikaGPU.h>`

Public članovi

- __host__ **SlikaGPU** (int sirina, int visina)
- __host__ **SlikaGPU** (const SlikaGPU &)=default
- SlikaGPU & **operator=** (const SlikaGPU &)
- SlikaGPU & **operator=** (SlikaGPU &&)
- __device__ __host__ Uint32 & operator[ ] (int idx)

  SlikaGPU(SlikaGPU &&); //TODO: implementiraj ovo!
- __device__ __host__ void **PostaviSliku** (Uint32 *nova)
- __device__ __host__ Uint32 * **DajSliku** () const
- __host__ void **DajSliku** (Uint32 *dest) const
- __device__ __host__ std::pair< int, int > DajDimenzije () const
- __device__ __host__ std::pair< int, int > DajBrojBlokova () const
- __host__ void Raytrace (ScenaGPU &scena, Kamera &kamera)
- __host__ void RaytraceCPU (ScenaGPU &scena, Kamera &kamera, Uint32 *slika)

4.36.1 Detaljno objašnjenje

Klasa koja predstavlja sliku na GPU-u.

U ovoj klasi se sve dinamički alocira na GPU-u, a kasnije po potrebi slika prekopira na host radi prikazivanja ili neke daljnje obrade. Ovdje se također događa GPU raytracing i transformacija preko zasebnih kernela.

4.36.2 Dokumentacija funkcija

4.36.2.1 DajBrojBlokova()

```
__device__ __host__ std::pair< int, int > SlikaGPU::DajBrojBlokova ( ) const
```

Povratne vrijednosti

Vraća par (bx,by) za broj blokova po pojedinoj dimenziji slike.

4.36.2.2 DajDimenzije()

```
__device__ __host__ std::pair< int, int > SlikaGPU::DajDimenzije ( ) const
```

Povratne vrijednosti

Vraća par (širina,visina) slike.

4.36.2.3 operator[]()

```
__device__ __host__ Uint32& SlikaGPU::operator[] (
            int idx )  [inline]
```

SlikaGPU(SlikaGPU &&); //TODO: implementiraj ovo!

Daje indeksirani piksel slike.

Parametri

in	*idx*	Indeks

Povratne vrijednosti

RGB piksel pakiran u 32-bitni broj.

4.36.2.4 Raytrace()

```
__host__ void SlikaGPU::Raytrace (
            ScenaGPU & scena,
            Kamera & kamera )
```

Glavna host funkcija za početak GPU raytracinga.

Ova funkcija se brine o transformaciji, idealnoj raspodjeli niti u blokove i pozive svih odgovarajućih kernela koliko god puta je to potrebno.

Parametri

in	*scena*	Scena za renderirati.
in	*kamera*	Kamera s čijeg stajališta treba renderirati scenu.

4.36.2.5 RaytraceCPU()

```
__host__ void SlikaGPU::RaytraceCPU (
            ScenaGPU & scena,
            Kamera & kamera,
            Uint32 * slika )
```

Ovo se ne koristi.

Dokumentacija klase je napravljena iz datoteke:

- SlikaGPU.h
- SlikaGPU.cpp

4.37 Opis klase Svjetlo

```
#include <Svjetlo.h>
```

Public članovi

- __device__ __host__ Svjetlo (const Vektor3 &poz, RGB boja)
- __device__ __host__ **Svjetlo** (const Svjetlo &)=default
- __device__ __host__ Svjetlo & **operator=** (const Svjetlo &)=default
- __device__ __host__ **Svjetlo** (Svjetlo &&)=default
- __device__ __host__ Svjetlo & **operator=** (Svjetlo &&)=default
- __device__ const __host__ Vektor4 & **DajPoziciju** () const
- __device__ __host__ void Transformiraj ()

Public atributi

- RGB boja
- Matrica4x4 transformacija = Matrica4x4::Jedinicna()

4.37.1 Detaljno objašnjenje

Tip jednostavnog, točkovnog izvora svjetlosti.

4.37.2 Dokumentacija konstruktora i destruktora

4.37.2.1 Svjetlo()

```
__device__ __host__ Svjetlo::Svjetlo (
            const Vektor3 & poz,
            RGB boja )
```

Konstruktor svjetla s pozicijom i bojom.

Parametri

in	*poz*	Pozicija svjetla u svjetskom prostoru.
in	*boja*	Boja svjetla.

4.37.3 Dokumentacija funkcija

4.37.3.1 Transformiraj()

```
__device__ __host__ void Svjetlo::Transformiraj ( )
```

Primijeni zadanu transformaciju na svjetlo.

4.37.4 Documentacija varijabli

4.37.4.1 boja

```
RGB Svjetlo::boja
```

RGB boja svjetla.

4.37.4.2 transformacija

```
Matrica4x4 Svjetlo::transformacija = Matrica4x4::Jedinicna()
```

Općenita transformacija svjetla.

Dokumentacija klase je napravljena iz datoteke:

- Svjetlo.h
- Svjetlo.cpp

4.38 Opis strukture Tekstura

Public atributi

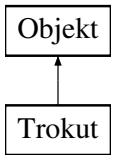
- uint16_t **sirina**
- uint16_t **visina**
- Uint32 * **tekseli**
- Vektor3 **ishodiste**
- Vektor3 **skala**
- Vektor3 **turbulencija**
- bool **clamp**

Dokumentacija strukture je napravljena iz datoteka:

- Materijal.h

4.39 Opis klase Trokut

Dijagram klasa za Trokut



Public članovi

- **Trokut** (const Vektor4 &, const Vektor4 &, const Vektor4 &, RGB)
- **Trokut** (const Vektor4 &, const Vektor4 &, const Vektor4 &)
- void **Transformiraj** () override
- bool **PresijeciZraku** (const Zraka &, Realan &) override
- RGB **Sjencaj** (const std::forward_list< Svjetlo * > &) override

Dodatni naslijeđeni članovi

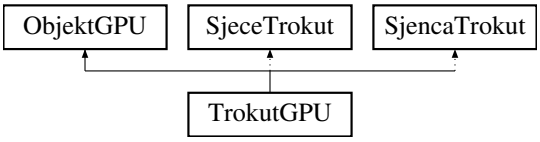
Dokumentacija klase je napravljena iz datoteke:

- Trokut.h
- Trokut.cpp

4.40 Opis klase TrokutGPU

`#include <TrokutGPU.h>`

Dijagram klasa za TrokutGPU



Public članovi

- __device__ __host__ TrokutGPU (const Vektor4 &a, const Vektor4 &b, const Vektor4 &c, RGB rgb)
- __device__ __host__ TrokutGPU (const Vektor4 &a, const Vektor4 &b, const Vektor4 &c)
- __device__ __host__ TrokutGPU (const Vektor4 &a, const Vektor4 &b, const Vektor4 &c, const Vektor3 &n)
- __device__ __host__ **TrokutGPU** (const TrokutGPU &)=default
- __device__ __host__ TrokutGPU & **operator=** (const TrokutGPU &)=default
- __device__ __host__ **TrokutGPU** (TrokutGPU &&)=default
- __device__ __host__ TrokutGPU & **operator=** (TrokutGPU &&)=default
- __device__ __host__ void **Transformiraj** (int) override
- __device__ __host__ int **DajBrojNitiZaTransform** () const override
- __host__ __device__ bool **PresijeciZraku** (const Zraka &, Realan &, Vektor3 &sjeciste) const override
- __host__ __device__ RGB **Sjencaj** (const Vektor3 &sjeciste, const Svjetlo[ ], size_t) const override
- __host__ RGB Sjencaj (const Vektor3 &sjeciste, const std::forward_list< Svjetlo > &svjetla)

Dodatni naslijeđeni članovi

4.40.1 Detaljno objašnjenje

Tip primitivnog GPU/CPU trokuta.

Ovaj tip je jednostavan i ne zahtijeva korištenje različitih klasa za GPU/CPU renderiranje. Ovaj tip se ne bi trebao koristiti za složene indeksirane mesheve; za to je TrokutModel!

4.40.2 Dokumentacija konstruktora i destruktora

4.40.2.1 TrokutGPU() [1/3]

```
__device__ __host__ TrokutGPU::TrokutGPU (
            const Vektor4 & a,
            const Vektor4 & b,
            const Vektor4 & c,
            RGB rgb )
```

Konstruktor trokuta s vrhovima i bojom.

Normala se automatski izračuna.

Parametri

in	*a,b,c*	Točke vrhova.
in	*rgb*	RGB boja cijelog trokuta.

4.40.2.2 TrokutGPU() [2/3]

```
__device__ __host__ TrokutGPU::TrokutGPU (
            const Vektor4 & a,
            const Vektor4 & b,
            const Vektor4 & c )
```

Konstruktor trokuta s vrhovima.

Normala se automatski izračuna.

Parametri

in	*a,b,c*	Točke vrhova.

4.40.2.3 TrokutGPU() [3/3]

```
__device__ __host__ TrokutGPU::TrokutGPU (
            const Vektor4 & a,
            const Vektor4 & b,
            const Vektor4 & c,
            const Vektor3 & n )
```

Konstruktor trokuta s vrhovima i normalom.

Parametri

in	*a,b,c*	Točke vrhova.
in	*n*	Vektor normale.

4.40.3 Dokumentacija funkcija

4.40.3.1 Sjencaj()

```
__host__ RGB TrokutGPU::Sjencaj (
            const Vektor3 & sjeciste,
            const std::forward_list< Svjetlo > & svjetla )
```

CPU-specifičan shader individualnog trokuta.

Parametri

in	*sjeciste*	Izračunato sjecište zrake i trokuta.
in	*svjetla*	Lista svih svjetala na sceni.

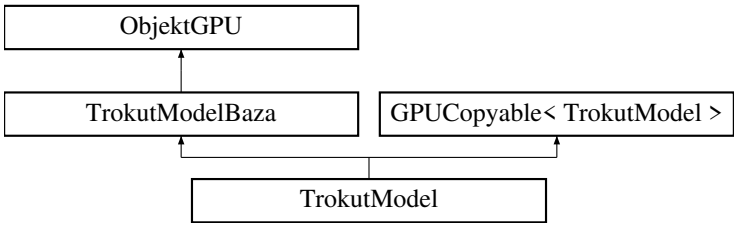
Dokumentacija klase je napravljena iz datoteke:

- TrokutGPU.h
- TrokutGPU.cpp

4.41 Opis klase TrokutModel

`#include <TrokutModel.h>`

Dijagram klasa za TrokutModel



Public članovi

- __host__ TrokutModel (const ModelOBJ::Objekt &obj)
- __host__ **TrokutModel** (const TrokutModel &)
- __host__ **TrokutModel** (TrokutModel &&)=default
- __host__ TrokutModel & **operator=** (const TrokutModel &)
- __host__ TrokutModel & **operator=** (TrokutModel &&)
- __host__ RGB Sjencaj (int, const Vektor3 &sjeciste, const std::forward_list< Svjetlo > &svjetla)
- __host__ bool PresijeciZraku (const Zraka &zraka, Realan &t, int &, Vektor3 &sjeciste)
- TrokutModelGPU PrekopirajNaGPU () override

  TrokutModelGPU PrekopirajNaGPU() const override;.
- __host__ void PrimijeniTransformaciju (const Matrica4x4 &mat)
- __host__ void **Transformiraj** (int) override
- __device__ __host__ Vektor3 * **DajVrhove** () const
- __device__ __host__ size_t **DajBrojVrhova** () const
- __device__ __host__ Vektor3 * **DajNormale** () const
- __device__ __host__ size_t **DajBrojNormala** () const
- __device__ __host__ Vektor3 * **DajUV** () const
- __device__ __host__ size_t **DajBrojUV** () const
- __device__ __host__ size_t **DajBrojPoligona** () const
- __device__ __host__ Poligon * **DajPoligone** () const
- __host__ void **PostaviVrhove** (Vektor3 *nv)
- __host__ void **PostaviNormale** (Vektor3 *nn)
- __host__ void **PostaviUV** (Vektor3 *nt)
- __host__ void **PostaviPoligone** (Poligon *p)

Public atributi

- friend **Scena**

Protected članovi

- __host__ void **Dealociraj** ()
- __host__ void **Realociraj** (const TrokutModel &)

Dodatni naslijeđeni članovi

4.41.1 Detaljno objašnjenje

Tip složenijeg indeksiranog trokutastog mesha na CPU-u.

Ovaj tip objekta zahtijeva dodatne alokacije i memorijski menadžment pa se za sada njegovo renderiranje radi odvojeno za GPU i za CPU. Ova klasa radi CPU stranu na kojoj svaki model počinje, a tada po potrebi korisnika može biti prekopiran na GPU i tamo renderiran preko ScenaGPU.

4.41.2 Dokumentacija konstruktora i destruktora

4.41.2.1 TrokutModel()

```
__host__ TrokutModel::TrokutModel (
            const ModelOBJ::Objekt & obj )
```

Konstruktor modela na temelju OBJ datoteke.

Parametri

in	*obj*	Jedan učitan objekt iz OBJ datoteke.

4.41.3 Dokumentacija funkcija

4.41.3.1 PrekopirajNaGPU()

```
TrokutModelGPU TrokutModel::PrekopirajNaGPU ( )  [override]
```

TrokutModelGPU PrekopirajNaGPU() const override;.

Prekopira model u GPU memoriju.

Povratne vrijednosti

Konstruirana kopija tipa TrokutModelGPU na GPU-u.

4.41.3.2 PresijeciZraku()

```
__host__ bool TrokutModel::PresijeciZraku (
            const Zraka & zraka,
            Realan & t,
            int & presjecenTrokut,
            Vektor3 & sjeciste )
```

Siječe zraku i cijeli mesh.

Ovo može biti vrlo skupo i zato treba ubrzavati s različitim hijerarhijskim akcelaracijskim strukturama. Za sada se jednostavno testiraju svi trokuti uz neke manje interne optimizacije. Za razliku od ostalih metoda za presjecanje, ova vraća i indeks presječenog trokuta koji je potreban radi sjenčanja kasnije.

Ovo je CPU verzija.

Parametri

in	*zraka*	Zraka koju provjeravamo.
out	*t*	Parametar t u jednadžbi zrake koji vodi do sjecišta.
out	*sjeciste*	Izračunato sjecište unutar trokuta u *baricentričnim koordinatama*!

Povratne vrijednosti

`true` ako zraka siječe bilo koji trokut mesha; inače `false`.

4.41.3.3 PrimijeniTransformaciju()

```
__host__ void TrokutModel::PrimijeniTransformaciju (
            const Matrica4x4 & mat )
```

Primijeni zadanu transformacijsku matricu na cijeli mesh.

Parametri

in	*mat*	Zadana matrica.

4.41.3.4 Sjencaj()

```
__host__ RGB TrokutModel::Sjencaj (
            int presjecenTrokut,
            const Vektor3 & sjeciste,
            const std::forward_list< Svjetlo > & svjetla )
```

CPU-specifičan shader za trokutasti mesh.

Parametri

in	*sjeciste*	Izračunato sjecište.
in	*svjetla*	Lista svih svjetala na sceni.

Povratne vrijednosti

RGB boja piksela.

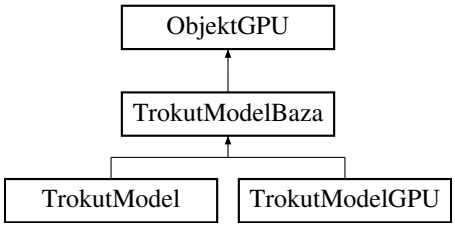
Dokumentacija klase je napravljena iz datoteke:

- TrokutModel.h
- TrokutModel.cpp

4.42 Opis klase TrokutModelBaza

```
#include <TrokutModelBaza.h>
```

Dijagram klasa za TrokutModelBaza



Strukture

- struct Podaci
- struct Poligon

Public članovi

- __device__ __host__ int **DajBrojNitiZaTransform** () const override

Public atributi

- Podaci **podaci**

Protected atributi

- RGB kd = {1.0f, 0.5f, 0.5f}

  Podaci podaci;.

4.42.1 Detaljno objašnjenje

Osnovna klasa za trokutaste mesheve.

Ovu klasu nasljeđuju svi koji žele koristiti osnovne podatke i funkcionalnosti vezane za primitivu trokutastih mesheva. U principu su to jedino TrokutModel i TrokutModelGPU.

4.42.2 Documentacija varijabli

4.42.2.1 kd

```
RGB TrokutModelBaza::kd = {1.0f, 0.5f, 0.5f}  [protected]
```

Podaci podaci;.

Difuzna boja površine mesha.

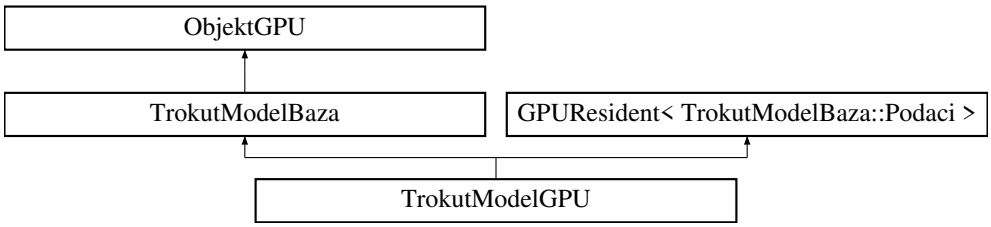
U nedostatku cijelog sustava za materijale, ovo služi za postavljanje različitih izgleda mesheva.

Dokumentacija klase je napravljena iz datoteka:

- TrokutModelBaza.h

4.43 Opis klase TrokutModelGPU

Dijagram klasa za TrokutModelGPU



Public članovi

- __host__ __device__ **TrokutModelGPU** (const TrokutModelBaza::Podaci &)
- __device__ __host__ **TrokutModelGPU** (const TrokutModel &mod)
- __host__ **TrokutModelGPU** (const TrokutModelGPU &)=default
- __host__ TrokutModelGPU & **operator=** (const TrokutModelGPU &)=default
- __host__ **TrokutModelGPU** (TrokutModelGPU &&)=default
- __device__ __host__ TrokutModelGPU & **operator=** (TrokutModelGPU &&)=default
- __host__ Podaci **DajPodatke** () override
- __device__ void **Transformiraj** (int offset) override
- __device__ bool PresijeciZraku (const Zraka &zraka, Realan &t, int &presjecenTrokut, Vektor3 &bariSjeciste)
- __device__ RGB **Sjencaj** (int presjecenTrokut, const Vektor3 &sjeciste, const Svjetlo[ ], size_t)

Public atributi

- friend **TrokutModel**
- friend **ScenaGPU**

Protected članovi

- __host__ void **UpisiPodatke** (const Podaci &novi) override
- __host__ void **Unisti** () override

Dodatni naslijeđeni članovi

4.43.1 Dokumentacija funkcija

4.43.1.1 PresijeciZraku()

```
__device__ bool TrokutModelGPU::PresijeciZraku (
             const Zraka & zraka,
             Realan & t,
             int & presjecenTrokut,
             Vektor3 & bariSjeciste )
```

Realan tMin = BESKONACNO;

Dokumentacija klase je napravljena iz datoteke:

- TrokutModelGPU.h
- TrokutModelGPU.cpp

4.44 Opis klase Vektor3

```
#include <Vektor3.h>
```

Public članovi

- __device__ __host__ Vektor3 ()
- __device__ __host__ Vektor3 (Realan a, Realan b, Realan c)
- __device__ __host__ **Vektor3** (const Vektor3 &)=default
- __device__ __host__ Vektor3 & **operator=** (const Vektor3 &)=default
- __device__ __host__ **Vektor3** (Vektor3 &&)=default
- __device__ __host__ Vektor3 & **operator=** (Vektor3 &&)=default
- __device__ __host__ Vektor3 operator+ (const Vektor3 &drugi) const
- __device__ __host__ Vektor3 operator- (const Vektor3 &drugi) const
- __device__ __host__ Vektor3 operator- () const
- __device__ __host__ Vektor3 & operator+= (const Vektor3 &drugi)
- __device__ __host__ Vektor3 operator^ (const Vektor3 &drugi) const
- __device__ __host__ Realan operator∗ (const Vektor3 &drugi) const
- __device__ __host__ Realan operator[ ] (int i) const
- __device__ __host__ Realan EuklidskaNorma () const
- __device__ __host__ Vektor3 Normiraj () const
- __device__ __host__ Realan **MaxNorma** () const
- __device__ __host__ Vektor4 DajTocku () const
- __device__ __host__ RGB **DajRGB** ()
- __device__ __host__ void DajKomponente (Realan &x, Realan &y, Realan &z) const

Friend-ovi

- __device__ __host__ friend Vektor3 operator∗ (Realan skalar, const Vektor3 &vektor)
- __device__ __host__ friend Vektor3 **operator**∗ (const Vektor3 &vektor, Realan skalar)

4.44.1 Detaljno objašnjenje

Tip trodimenzionalnog vektora.

4.44.2 Dokumentacija konstruktora i destruktora

4.44.2.1 Vektor3() [1/2]

```
__device__ __host__ Vektor3::Vektor3 ( )
```

Konstruira nulvektor.

4.44.2.2 Vektor3() [2/2]

```
__device__ __host__ Vektor3::Vektor3 (
            Realan a,
            Realan b,
            Realan c )
```

Konstruira 3-vektor iz 3 dana elementa.

Parametri

in	*a,b,c*	Dani elementi.

4.44.3 Dokumentacija funkcija

4.44.3.1 DajKomponente()

```
__device__ __host__ void Vektor3::DajKomponente (
            Realan & x,
            Realan & y,
            Realan & z ) const
```

Vraća sve komponente vektora po referenci.

Parametri

out	*x,y,z*	Komponente.

4.44.3.2 DajTocku()

```
__device__ __host__ Vektor4 Vektor3::DajTocku ( ) const
```

Vraća točku koja odgovara radijusvektoru. Vraća 4-vektor s 1.0 u posljednjoj komponenti, gdje su prethodne komponente identične onima iz trenutnog 3-vektora.

4.44.3.3 EuklidskaNorma()

```
__device__ __host__ Realan Vektor3::EuklidskaNorma ( ) const
```

Računa standardnu euklidsku normu vektora.

4.44.3.4 Normiraj()

```
__device__ __host__ Vektor3 Vektor3::Normiraj ( ) const
```

Vraća normiran vektor.

4.44.3.5 operator∗()

```
__device__ __host__ Realan Vektor3::operator* (
            const Vektor3 & drugi ) const
```

Računa skalarni produkt ovog i drugog danog 3-vektora.

Parametri

in	*drugi*	Desni operand skalarnog produkta.

Povratne vrijednosti

Skalar koji je rezultat skalarnog produkta.

4.44.3.6 operator+()

```
__device__ __host__ Vektor3 Vektor3::operator+ (
            const Vektor3 & drugi ) const
```

Zbraja dva 3-vektora.

Parametri

in	*drugi*	Desni vektorski operand.

Povratne vrijednosti

Novi 3-vektor sume.

4.44.3.7 operator+=()

```
__device__ __host__ Vektor3 & Vektor3::operator+= (
            const Vektor3 & drugi )
```

Kombinirani operator zbrajanja i pridruživanja 3-vektora.

Parametri

in	*drugi*	Desni vektorski operand.

Povratne vrijednosti

Referenca na ovaj 3-vektor.

4.44.3.8 operator-() [1/2]

```
__device__ __host__ Vektor3 Vektor3::operator- ( ) const
```

Negira 3-vektor.

Povratne vrijednosti

Novi negirani 3-vektor.

4.44.3.9 operator-() [2/2]

```
__device__ __host__ Vektor3 Vektor3::operator- (
            const Vektor3 & drugi ) const
```

Oduzima dva 3-vektora.

Parametri

in	*drugi*	Desni vektorski operand.

Povratne vrijednosti

Novi 3-vektor razlike.

4.44.3.10 operator[]()

```
__device__ __host__ Realan Vektor3::operator[] (
            int i ) const
```

Vraća i-ti element vektora.

Parametri

in	*i*	Indeks

4.44.3.11 operator^()

```
__device__ __host__ Vektor3 Vektor3::operator^ (
            const Vektor3 & drugi ) const
```

Računa vektorski produkt ovog i drugog danog 3-vektora.

Parametri

in	*drugi*	Desni operand vektorskog produkta.

Povratne vrijednosti

Vektorski produkt ovog i danog 3-vektora.

4.44.4 Dokumentacija povezanih funkcija

4.44.4.1 operator*

```
__device__ __host__ friend Vektor3 operator* (
            Realan skalar,
            const Vektor3 & vektor )  [friend]
```

Množenje vektora sa skalarom.

Parametri

in	*skalar*	Realan skalar.
in	*vektor*	Vektorski operand.

Povratne vrijednosti

Produktni vektor.

Dokumentacija klase je napravljena iz datoteke:

- Vektor3.h
- Vektor3.cpp

4.45 Opis klase Vektor4

`#include <Vektor4.h>`

Public članovi

- __device__ __host__ Vektor4 ()
- __device__ __host__ Vektor4 (Realan a, Realan b, Realan c, Realan d)
- __device__ __host__ **Vektor4** (const Vektor4 &)=default
- __device__ __host__ Vektor4 & **operator=** (const Vektor4 &)=default
- __device__ __host__ **Vektor4** (Vektor4 &&)=default
- __device__ __host__ Vektor4 & **operator=** (Vektor4 &&)=default
- __device__ __host__ Vektor4 operator+ (const Vektor4 &drugi) const
- __device__ __host__ Vektor4 operator- (const Vektor4 &drugi) const
- __device__ __host__ Vektor4 **operator-** () const
- __device__ __host__ Realan operator∗ (const Vektor4 &drugi)
- __device__ __host__ Realan operator[ ] (int i)
- __device__ __host__ Vektor3 PerspektivnoDijeljenje () const
- __device__ __host__ Realan **EuklidskaNorma** () const
- __device__ __host__ Realan **MaxNorma** () const
- __device__ __host__ Vektor3 Radijusvektor () const

Friend-ovi

- __device__ __host__ Vektor4 operator∗ (Realan, const Vektor4 &)
- __device__ __host__ Vektor4 **operator**∗ (const Vektor4 &, Realan)
- __device__ __host__ Vektor4 operator- (const Vektor4 &vek)

4.45.1 Detaljno objašnjenje

Tip četverodimenzionalnog vektora.

4.45.2 Dokumentacija konstruktora i destruktora

4.45.2.1 Vektor4() [1/2]

```
__device__ __host__ Vektor4::Vektor4 ( )
```

Konstruira nulvektor.

4.45.2.2 Vektor4() [2/2]

```
__device__ __host__ Vektor4::Vektor4 (
            Realan a,
            Realan b,
            Realan c,
            Realan d )
```

Konstruira 4-vektor iz 4 dana elementa.

Parametri

in	*a,b,c,d*	Dani elementi.

4.45.3 Dokumentacija funkcija

4.45.3.1 operator∗()

```
__device__ __host__ Realan Vektor4::operator* (
            const Vektor4 & drugi )
```

Računa skalarni produkt ovog i drugog danog 4-vektora.

Parametri

in	*drugi*	Desni operand skalarnog produkta.

Povratne vrijednosti

Skalar koji je rezultat skalarnog produkta.

4.45.3.2 operator+()

```
__device__ __host__ Vektor4 Vektor4::operator+ (
            const Vektor4 & drugi ) const
```

Zbraja dva 4-vektora.

Parametri

in	*drugi*	Desni vektorski operand.

Povratne vrijednosti

Novi 4-vektor sume.

4.45.3.3 operator-()

```
__device__ __host__ Vektor4 Vektor4::operator- (
            const Vektor4 & drugi ) const
```

Oduzima dva 4-vektora.

Parametri

in	*drugi*	Desni vektorski operand.

Povratne vrijednosti

Novi 4-vektor razlike.

4.45.3.4 operator[]()

```
__device__ __host__ Realan Vektor4::operator[] (
            int i )
```

Vraća i-ti element vektora.

Parametri

in	*i*	Indeks

4.45.3.5 PerspektivnoDijeljenje()

```
__device__ __host__ Vektor3 Vektor4::PerspektivnoDijeljenje ( ) const
```

Projicira točku u 3-vektor. Računa se vektor $(x/w, y/w, z/w, 1)$ i vraća samo prve tri navedene koordinate.

4.45.3.6 Radijusvektor()

```
__device__ __host__ Vektor3 Vektor4::Radijusvektor ( ) const
```

Vraća odgovarajući radijusvektor.

Povratne vrijednosti

3-vektor (x, y, z).

4.45.4 Dokumentacija povezanih funkcija

4.45.4.1 operator∗

```
__device__ __host__ Vektor4 operator* (
            Realan skalar,
            const Vektor4 & vek )  [friend]
```

Množenje vektora sa skalarom.

Parametri

in	*skalar*	Realan skalar.
in	*vektor*	Vektorski operand.

Povratne vrijednosti

Produktni vektor.

4.45.4.2 operator-

```
__device__ __host__ Vektor4 operator- (
            const Vektor4 & vek )  [friend]
```

Negira dani 4-vektor.

Parametri

in	*vek*	Dani vektor.

Povratne vrijednosti

Novi negirani 4-vektor.

Dokumentacija klase je napravljena iz datoteke:

- Vektor4.h
- Vektor4.cpp

4.46 Opis klase Zraka

`#include <Zraka.h>`

Public članovi

- __device__ __host__ Zraka (Vektor4 &o, Vektor4 &c)
- __device__ __host__ Zraka (Vektor4 &o, Vektor3 &smjer)
- __device__ __host__ Zraka (Vektor3 &o, Vektor3 &smjer)
- __device__ __host__ **Zraka** (const Zraka &)=default
- __device__ __host__ Zraka & **operator=** (const Zraka &)=default
- __device__ __host__ **Zraka** (Zraka &&)=default
- __device__ __host__ Zraka & **operator=** (Zraka &&)=default
- __device__ const __host__ Vektor3 & **Ishodiste** () const
- __device__ const __host__ Vektor3 & **Smjer** () const

4.46.1 Detaljno objašnjenje

Tip zrake u 3-prostoru.

4.46.2 Dokumentacija konstruktora i destruktora

4.46.2.1 Zraka() [1/3]

```
__device__ __host__ Zraka::Zraka (
            Vektor4 & o,
            Vektor4 & c )
```

Konstruktor zrake s ishodištem i ciljem.

Parametri

in	*o*	Točka ishodišta.
in	*c*	Točka cilja.

4.46.2.2 Zraka() [2/3]

```
__device__ __host__ Zraka::Zraka (
            Vektor4 & o,
            Vektor3 & smjer )
```

Konstruktor zrake s ishodištem i smjerom.

Parametri

in	*o*	Točka ishodišta.
in	*smjer*	Smjer zrake.

4.46.2.3 Zraka() [3/3]

```
__device__ __host__ Zraka::Zraka (
            Vektor3 & o,
            Vektor3 & smjer )
```

Konstruktor zrake s ishodištem i smjerom.

Parametri

in	*o*	Točka ishodišta.
in	*smjer*	Smjer zrake.

Dokumentacija klase je napravljena iz datoteke:

- Zraka.h
- Zraka.cpp

Poglavlje 5

Dokumentacija datoteka

5.1 Opis datoteke MatBaza.h

```
#include <cfloat>
#include <cmath>
```

Strukture

- class MatBaza

Typedef-ovi

- typedef float **Realan**

Varijable

- const float BESKONACNO = FLT_MAX

5.1.1 Detaljno objašnjenje

Zaglavlje s osnovnim konstantama i definicijama.

5.1.2 Dokumentacija varijable

5.1.2.1 BESKONACNO

```
const float BESKONACNO = FLT_MAX
```

Konstanta koja predstavlja maksimalan koristan float broj.

Korišteno primarno radi pronalaženja minimalnih sjecišta zraka.

Kazalo