Bibliotecas úteis para C#

Autor:
Tomas Tamantini

Belo Horizonte - Jan 2019

Sumário

1 Vectors **3**
1.1 Vector 3

2 Movables **4**
2.1 Movable 4

3 CGI **8**
3.1 Ray 8
3.2 Light Source 9
3.2.1 LightSourceDiffuse 10
3.2.2 LightSourceInfinity 10
3.2.3 LightSourcePoint 11
3.3 Chip 11
3.4 OpaqueObject:Movable 12
3.4.1 Plane 13
3.4.2 Disco 14
3.4.3 Triangle 14
3.4.4 Sphere 15
3.4.5 Cylinder 16
3.4.6 Cone 17
3.4.7 Paraboloid 18
3.4.8 Ellipsoid 19
3.5 Frame 20
3.6 Camera:Movable 21

4 Bibliografia **23**

Lista de Figuras

1 Exemplos de operações vetoriais 3
2 Objeto no espaço 5
3 Reflexão ao longo do eixo j' 6
4 Deslocamento do objeto de 4 unidades ao longo dos eixos j e j' 7
5 Rotação de 34º ao redor dos eixos k e k' 8
6 Caminho percorrido por raio de luz 9
7 Diagrama de classe das diferentes fontes de luz 9
8 Fonte de luz difusa 10
9 Fonte de luz no infinito 10
10 Fonte de luz num ponto finito 11
11 Elemento de área (chip) para reflexão difusa da luz 11
12 Objeto posicionado e orientado no espaço, e um elemento de área destacado 12
13 Plano finito e infinito 13
14 Equação do disco, e exemplo de um dico no espaço sem e com os eixos de referência evidenciados 14
15 Equação do triângulo, e exemplo de um triângulo no espaço sem e com vértices e eixos de referência evidenciados 15
16 Equação da esfera, esfera *Beachball* e esferas de cores sólidas 16
17 Equação do cilindro, e exemplo de cilindro no espaço 17
18 Equação do cone, e exemplo de cone no espaço 18
19 Equação do parabolóide, e exemplo de parabolóide no espaço 19
20 Equação do elipsóide, e exemplo de elipsóide no espaço 20
21 Conversão da matriz de intensidades em imagem *Bitmap*. No exemplo dado, a constante multiplicativa é igual à 180 21
22 Diagrama da câmera. No exemplo dado a resolução é 5x4 pixels 22

1 Vectors

A biblioteca foi criada para o uso de instâncias e operações vetoriais

1.1 Vector

Classe para a representação de vetores tri-dimensionais, e suas operações básicas. Alguns exemplos são dados na figura 1 dados abaixo.

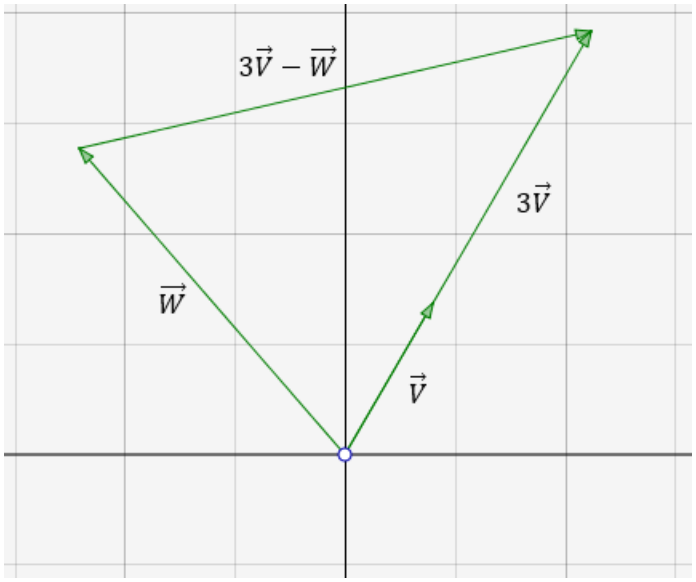


Figura 1: Exemplos de operações vetoriais

Propriedades

1. X (*double*) (Get/Set) - Coordenada X do vetor
2. Y (*double*) (Get/Set) - Coordenada Y do vetor
3. Z (*double*) (Get/Set) - Coordenada Z do vetor

Métodos

1. LengthSq () (*double*) - Comprimento do vetor ao quadrado
2. Length () (*double*) - Comprimento do vetor
3. Unit () (*Vector*) - Vetor unitário paralelo ao vetor dado (se não houver retorna *Null*)

Métodos estáticos

1. * (double, Vector),(Vector,double) (*double*) - Vetor multiplicado por escalar
2. * (Vector, Vector) (*double*) - Produto escalar de dois vetores

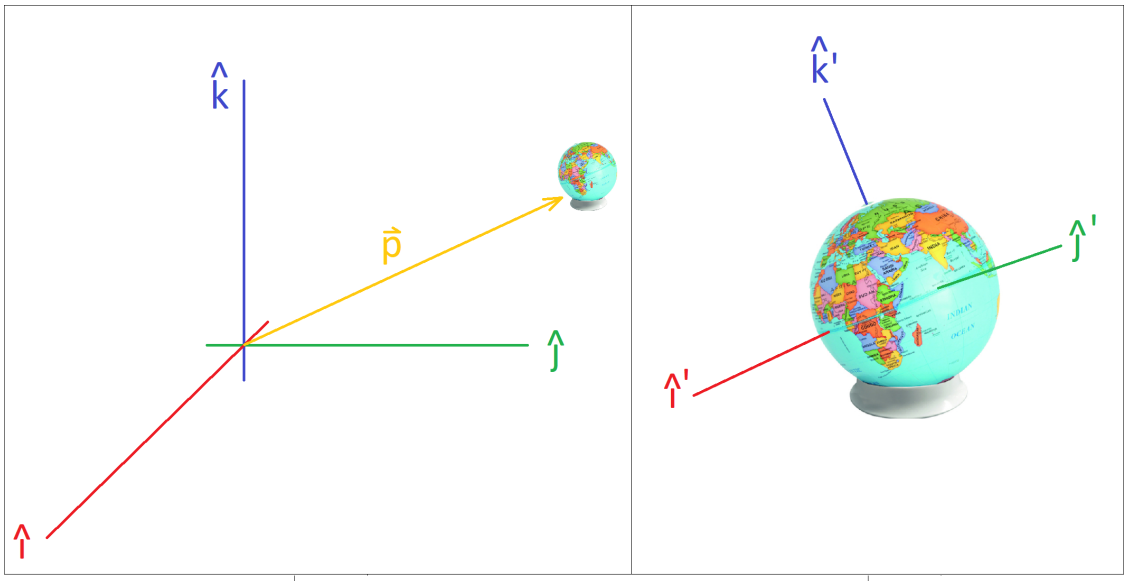
3. + (Vector, Vector) (*Vector*) - Soma de dois vetores
4. - (Vector, Vector) (*Vector*) - Diferença de dois vetores
5. - (Vector) (*Vector*) - Inverso de um vetor
6. | (Vector, Vector) (*Vector*) - Produto vetorial de dois vetores

2 Movables

A biblioteca foi criada para operações com objetos posicionados e orientados no espaço tridimensional.

2.1 Movable

Super classe abstrata para objetos no espaço tridimensional (figura 2), com posição dada, e seu próprio eixo de coordenadas de referência, além das operações para movimentar, rotacionar e refletir o objeto.



$\vec{p} \rightarrow$ *Posição do objeto (centro do seu sistema de coordenadas)*

$\vec{r} \rightarrow$ *Ponto descrito no sistema de coordenadas absoluto* (i, j, k)

$$\vec{r} = \begin{bmatrix} \vec{r} \cdot \hat{\imath} \\ \vec{r} \cdot \hat{\jmath} \\ \vec{r} \cdot \hat{k} \end{bmatrix}$$

$\vec{r}' \rightarrow$ *Ponto descrito no sistema de coordenadas do objeto* (i', j', k')

$$\vec{r}' = \begin{bmatrix} (\vec{r} - \vec{p}) \cdot \hat{\imath}' \\ (\vec{r} - \vec{p}) \cdot \hat{\jmath}' \\ (\vec{r} - \vec{p}) \cdot \hat{k}' \end{bmatrix}'$$

$M \rightarrow$ *Matriz de orientação do objeto*

$$M = \begin{bmatrix} i'_x & j'_x & k'_x \\ i'_y & j'_y & k'_y \\ i'_z & j'_z & k'_z \end{bmatrix}$$

$$M^{-1} = M^T$$

Transformações:

$$\vec{r}' = M^T(\vec{r} - \vec{p})$$
$$\vec{r} = M\vec{r}' + \vec{p}$$

Figura 2: Objeto no espaço

Propriedades

1. Pos (*Vector*) (Get/Set) - Posição do objeto (Origem do seu sistem de coordenadas)
2. Iprime (*Vector*) (Readonly) - Eixo i no sistema de coordenadas do objeto
3. Jprime (*Vector*) (Readonly) - Eixo j no sistema de coordenadas do objeto
4. Kprime (*Vector*) (Readonly) - Eixo k no sistema de coordenadas do objeto

Métodos protegidos

1. MyPoint (Vector) (*Vector*) - Converte as coordenadas do ponto dado para o referencial do objeto. Leva em conta a posição do objeto
2. MyVec (Vector) (*Vector*) - Converte as coordenadas do vetor dado para o referencial do objeto. Não leva em conta a posição do objeto, só sua orientação

3. AbsPoint (Vector) (*Vector*) - Converte as coordenadas do ponto dado para o referencial absoluto. Leva em conta a posição do objeto
4. AbsVec (Vector) (*Vector*) - Converte as coordenadas do vetor dado para o referencial absoluto. Não leva em conta a posição do objeto, só sua orientação

Métodos públicos

1. Reflect () (*void*) - Reflete o objeto ao redor do eixo j'
2. Move (bool, Vector) (*void*) - Move o objeto na direção dada. A direção pode ser dada no referencial absoluto ou no referencial do objeto.
3. Orient (Vector, Vector) (*void*) - Reorienta os objetos para os eixos dados
4. Rotate (bool, Vector, double) (*void*) - Rotaciona o objeto dado eixo e ângulo. O eixo pode ser dado no referencial absoluto ou no referencial do objeto
5. Rotate (bool, Vector, Vector) (*void*) - Rotaciona o objeto dado matriz de rotação (dada pelas suas duas primeiras colunas). A matriz pode ser dada no referencial absoluto ou no referencial do objeto

As figuras 3, 4 e 5 que se seguem ilustram os métodos públicos apresentados acima:

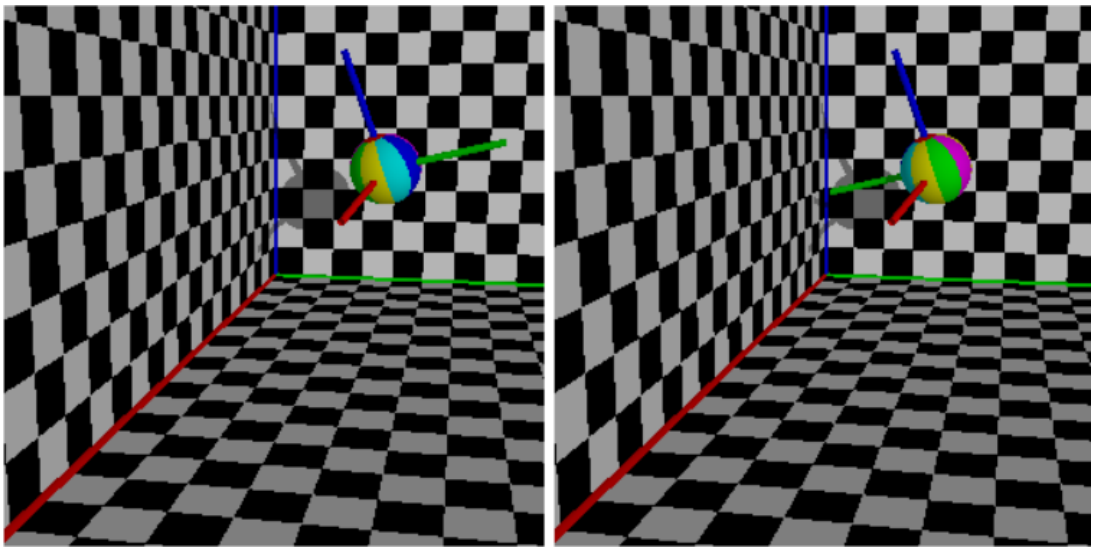


Figura 3: Reflexão ao longo do eixo j'

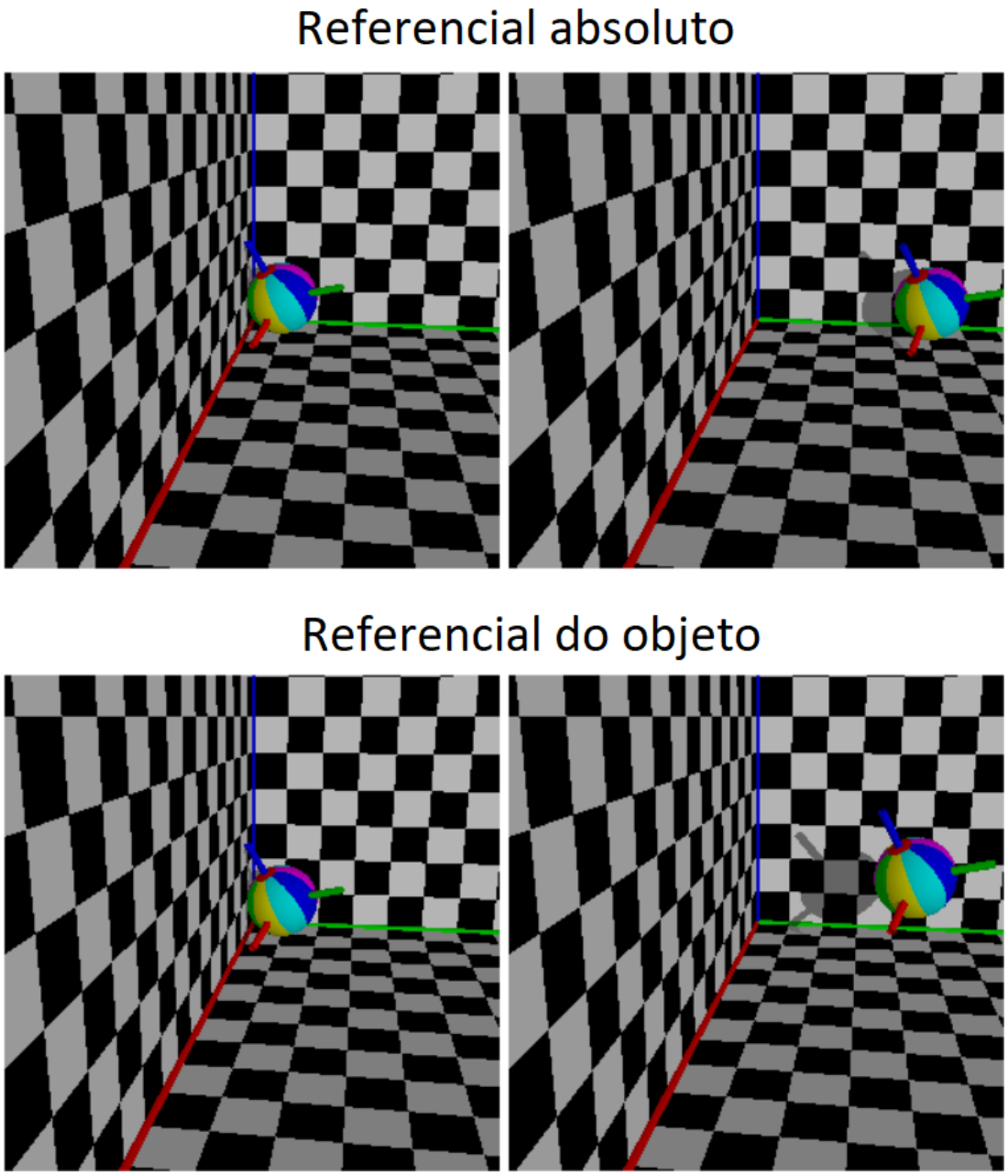


Figura 4: Deslocamento do objeto de 4 unidades ao longo dos eixos j e j'

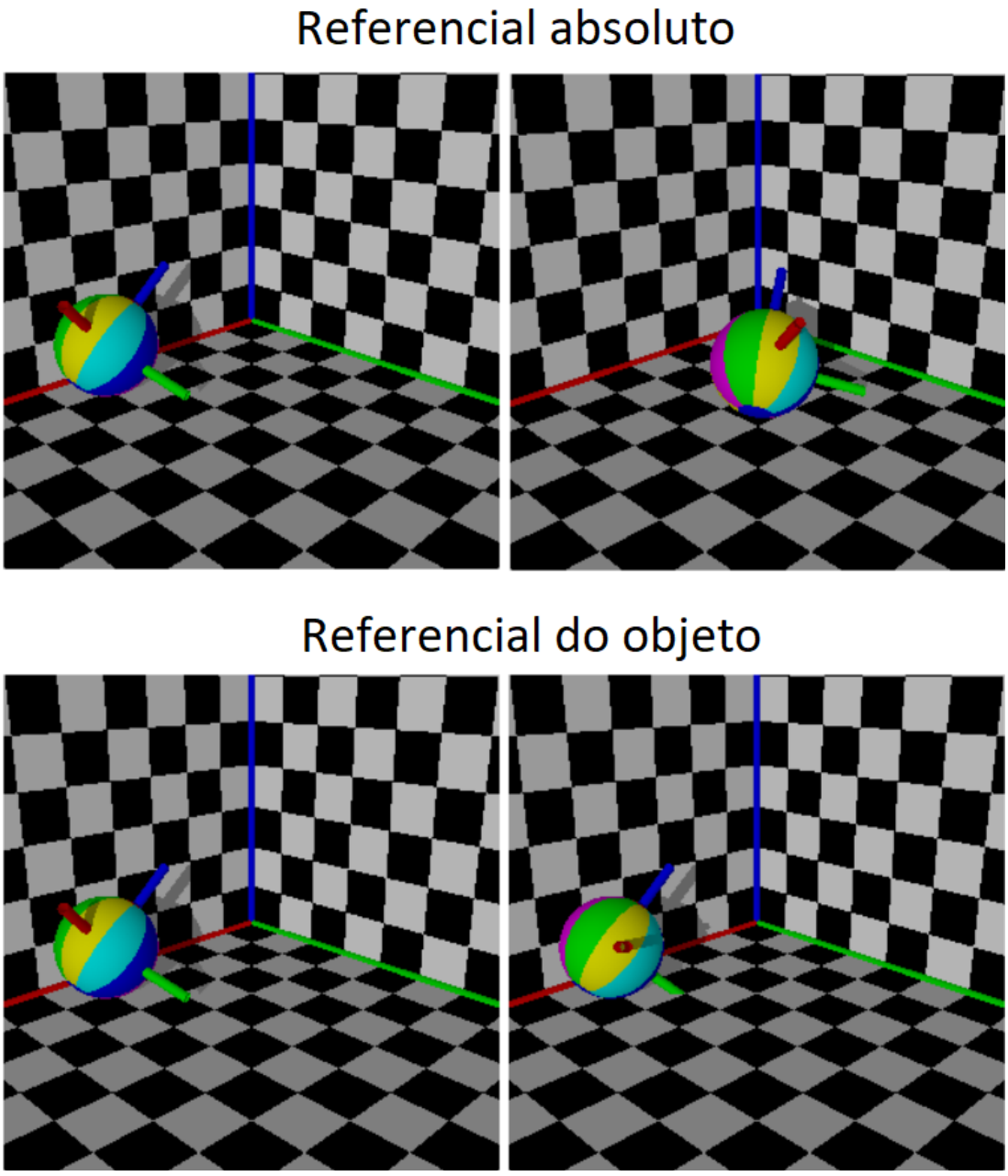


Figura 5: Rotação de 34º ao redor dos eixos k e k'

3 CGI

A biblioteca foi criada para a geração de imagens tri dimensionais computadorizadas. As imagens incluem iluminação, e objetos opacos.

3.1 Ray

Classe para a representação dos caminhos percorridos pela luz (retas no espaço tri dimensional). Vide figura 6 abaixo:

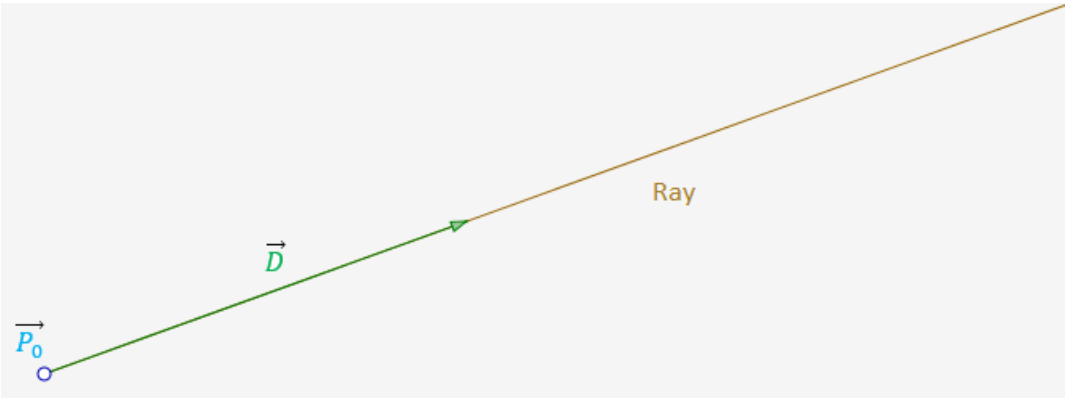


Figura 6: Caminho percorrido por raio de luz

Propriedades

1. P0 (*Vector*) (Readonly) - Ponto inicial do raio
2. D (*Vector*) (Readonly) - Direção do raio

3.2 Light Source

Super classe (abstract/mustinherit) para a representação das fontes de luz. Tem três subclasses que são instanciáveis (figura 7):

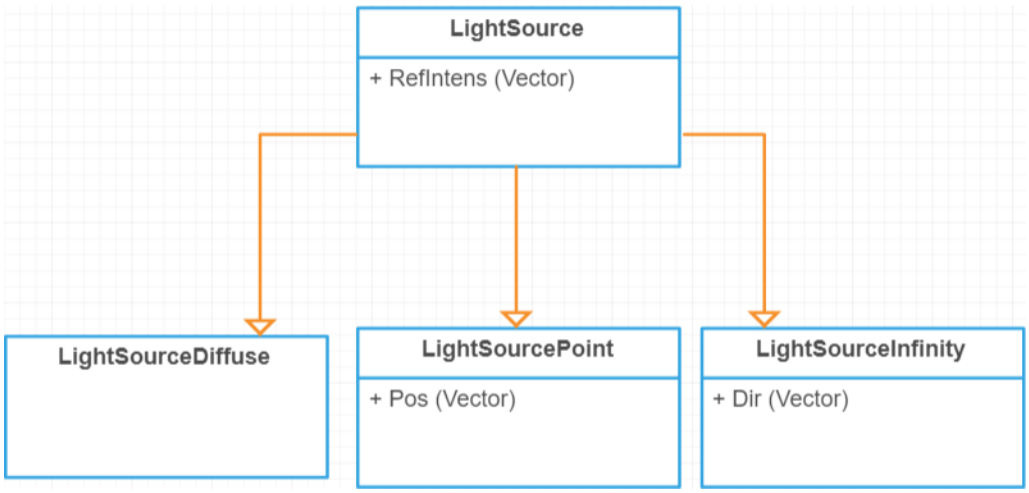


Figura 7: Diagrama de classe das diferentes fontes de luz

Propriedades RefIntens (*Vector*) (Get/Set) - Referência de intensidade da fonte de luz. As três coordenadas correspondem a Red, Green e Blue (RGB).

3.2.1 LightSourceDiffuse

Sub-classe de LightSource, correspondente a luz difusa no ambiente. É uma fonte não-direcional, e que, portanto, não pode ser bloqueada por um obstáculo (vide figura 8). A classe apresentada aqui só permite a instanciação de tais fontes com uma intensidade (e cor) constantes ao longo de todo o espaço, mas modificações simples podem ser feitas para que essa intensidade se torne função da posição e do tempo.

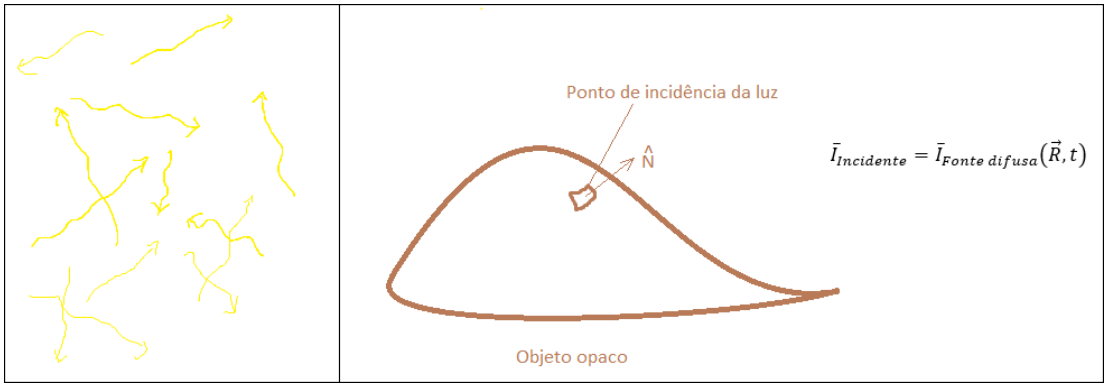


Figura 8: Fonte de luz difusa

3.2.2 LightSourceInfinity

Sub-classe de LightSource, correspondente a luz vinda de uma fonte no infinito (raios incidentes paralelos). É uma fonte de intensidade constante. Além disso é direcional, e pode ser bloqueada por um obstáculo (vide figura 9).

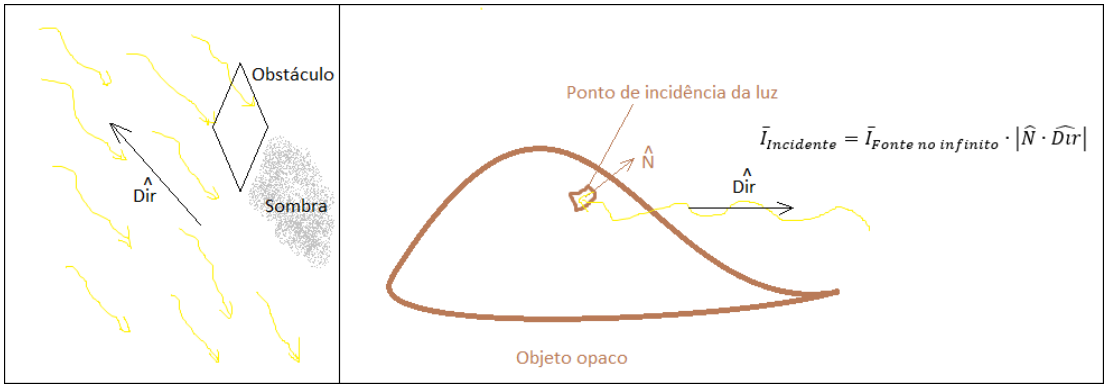


Figura 9: Fonte de luz no infinito

Propriedades Dir (*Vector*) (Get/Set) - Direção de onde vêm os raios de luz. É sempre um vetor unitário. Se o usuário fornecer um vetor inválido (o vetor (0,0,0)), o default considerado é (0,0,1).

3.2.3 LightSourcePoint

Sub-classe de LightSource, correspondente a luz vinda de uma fonte num ponto finito (raios incidentes com simetria esférica). É uma fonte de intensidade dependente da distância da fonte. Além disso é direcional, e pode ser bloqueada por um obstáculo (vide figura 10).

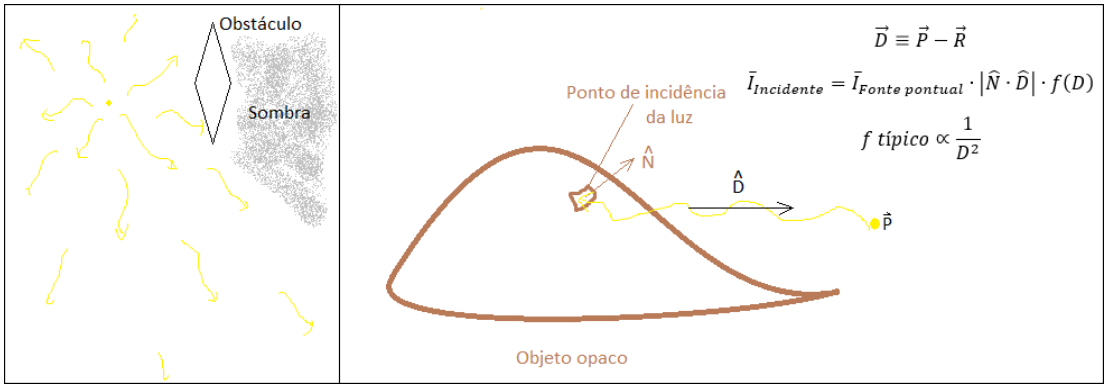


Figura 10: Fonte de luz num ponto finito

Propriedades Pos (*Vector*) (Get/Set) - Posição da fonte de luz pontual.

Métodos AttenuatedIntes (double) (*Vector*) - Intensidade de referência atenuada pela distância.

3.3 Chip

Classe para a representação do elemento de área (chip) do objeto onde a luz incide. É esse elemento de área, por reflexão difusa, que sensibiliza a câmera (figura 11):

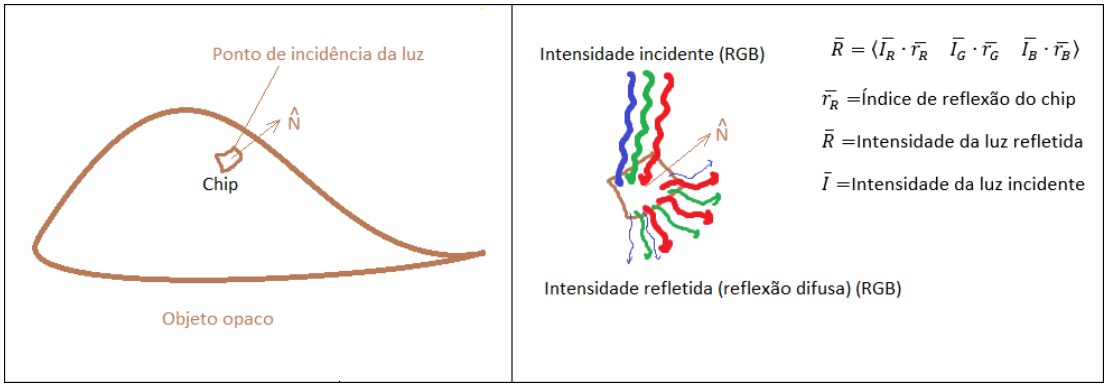


Figura 11: Elemento de área (chip) para reflexão difusa da luz

Propriedades

1. Pos (**Vector**) (Readonly) - Posição do chip
2. N (*Vector*) (Readonly) - Vetor normal ao chip
3. RefIdx (*Vector*) (Readonly) - Índice de reflexão do chip para cada uma das três cores (R,G,B). Cada uma das três entradas deve ser um número entre 0 e 1 (não reflete/reflete totalmente a cor).

3.4 OpaqueObject:Movable

Super classe abstrata para objeto posicionado e orientado no espaço tridimensional, opaco (não há reflexão especular ou refração da luz incidente, apenas reflexão difusa). Vide figura 12 abaixo:

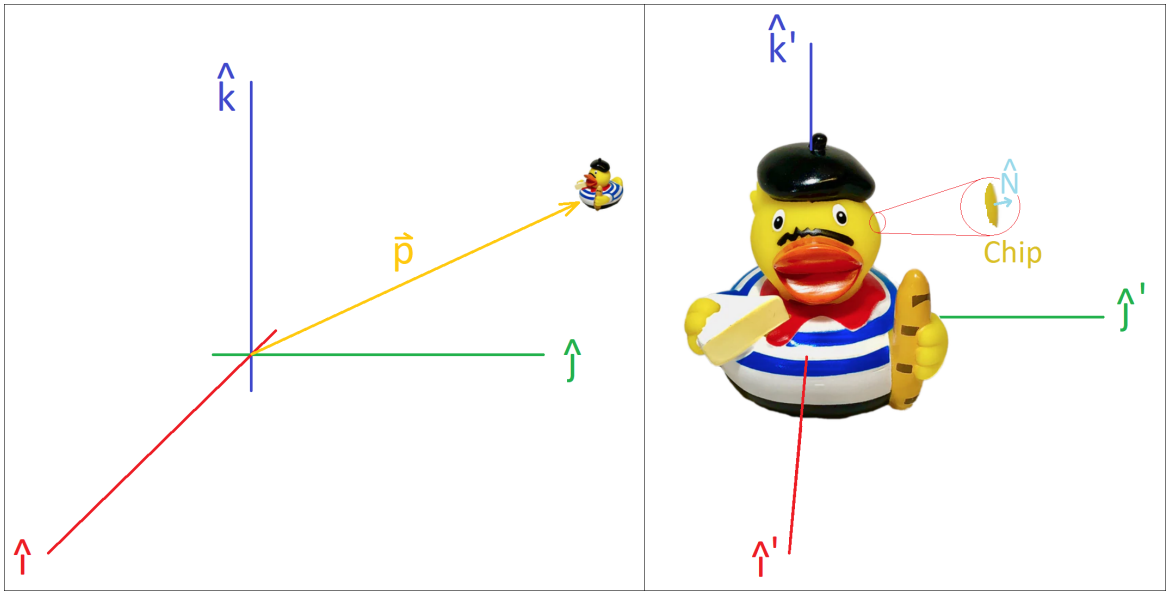


Figura 12: Objeto posicionado e orientado no espaço, e um elemento de área destacado

Propriedades

1. RefIdx (*Vector*) (Get/Set) - Índice de reflexão *default* do objeto para cada uma das três cores (R,G,B). Cada uma das três entradas deve ser um número entre 0 e 1 (não reflete/reflete totalmente a cor).

Métodos abstratos

1. OwnTime (Ray) (*double*) - Tempo que um raio (dado no referencial do próprio objeto) demora a atingir o objeto. Retorna -1 se o raio não atinge o objeto.
2. OwnChip (Vector) (*Chip*) - Elemento de área (chip) correspondente à posição. O input e output são dados no referencial do próprio objeto.

Métodos públicos

1. TimeInt (Ray) (*double*) - Tempo que um raio (dado no referencial absoluto) demora a atingir o objeto. Retorna -1 se o raio não atinge o objeto. (O método simplesmente converte o raio para o referencial do objeto e chama o método OwnTime())
2. OwnChip (Vector) (*Chip*) - Elemento de área (chip) correspondente à posição dada no referencial absoluto. (O método simplesmente converte a posição para o referencial do objeto e chama o método OwnChip())

3.4.1 Plane

Plano finito ou infinito. Pode ter um padrão quadriculado para ajudar a referenciar outros objetos (figura 13)

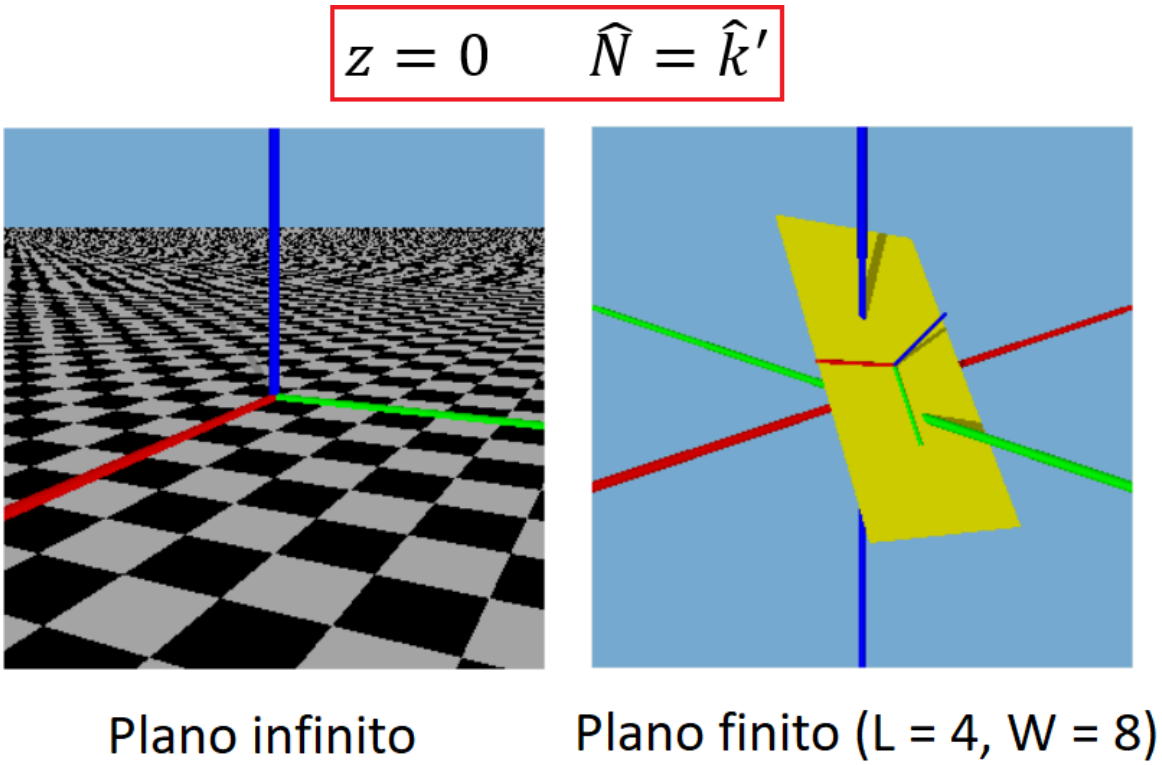


Figura 13: Plano finito e infinito

Propriedades

1. Checkered(*bool*) (Get/Set) - Indica se o plano deve apresentar padrão quadriculado preto e branco ao invés da cor sólida dada. Os quadrados tem sempre lado de tamanho unitário.
2. Length(*double*) (Get/Set) - Comprimento do plano ao longo do eixo i'.
3. Widthdouble) (Get/Set) - Largura do plano ao longo do eixo j'.

Métodos públicos

1. SetBounds (double, double) (*void*) - Reseta os limites para o plano.
2. SetInfinite () (*void*) - Retira os limites do plano, tornando-o infinito.

3.4.2 Disco

Disco plano de raio dado. Vide figura 14 abaixo

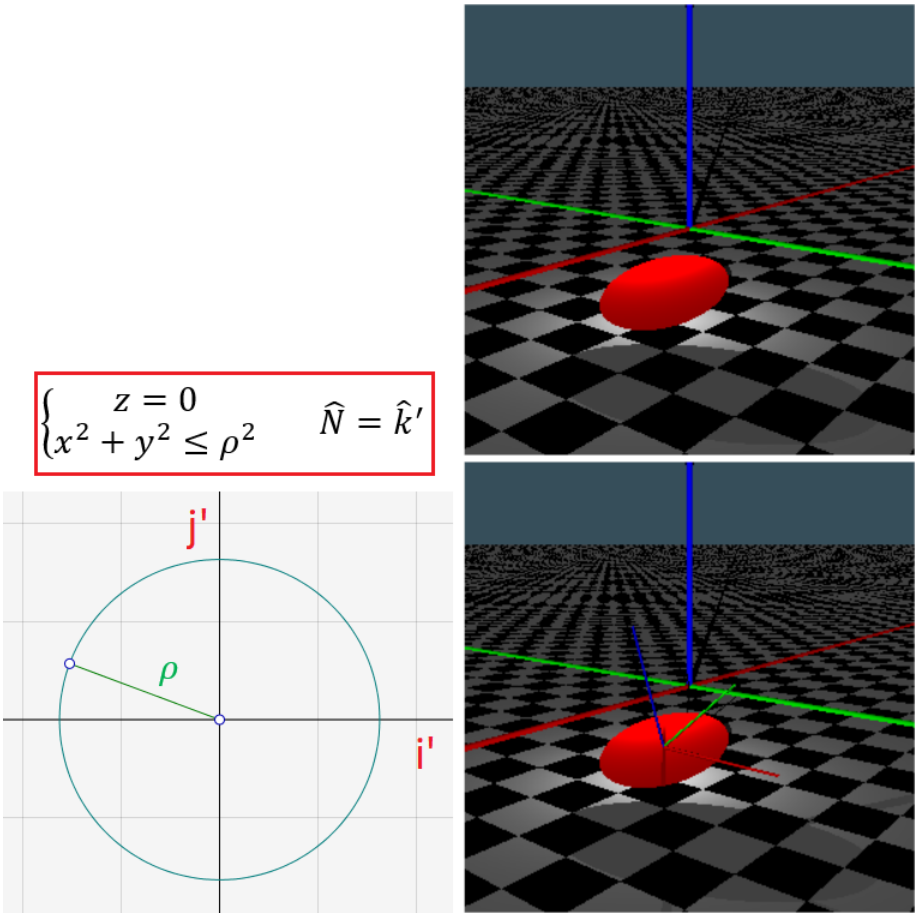


Figura 14: Equação do disco, e exemplo de um dico no espaço sem e com os eixos de referência evidenciados

Propriedades Ro(*double*) (Get/Set) - Raio do disco.

3.4.3 Triangle

Triângulo construído a partir de três pontos no espaço. Vide figura 15:

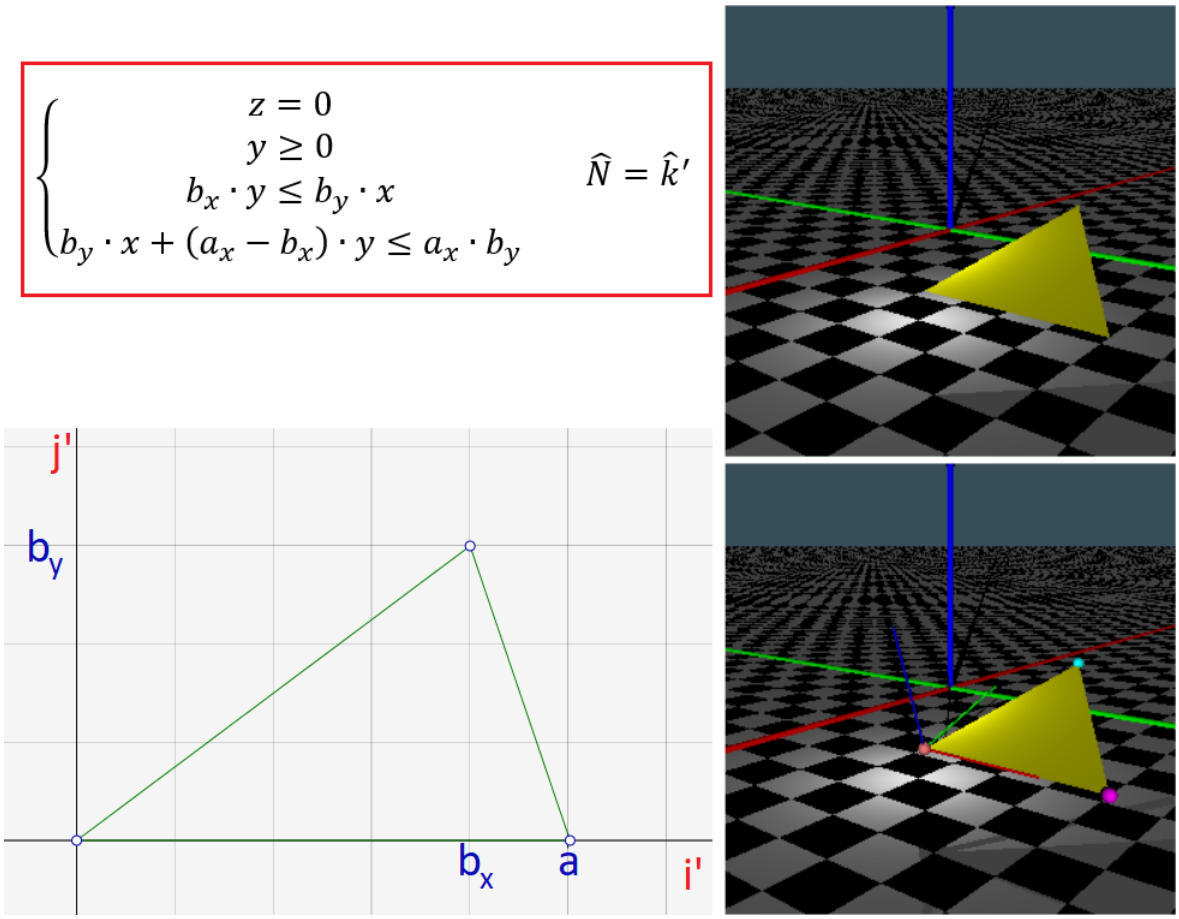


Figura 15: Equação do triângulo, e exemplo de um triângulo no espaço sem e com vértices e eixos de referência evidenciados

3.4.4 Sphere

Esfera de raio dado. Pode ter um padrão *Beachball* para facilitar a visualização de rotações (figura 16)

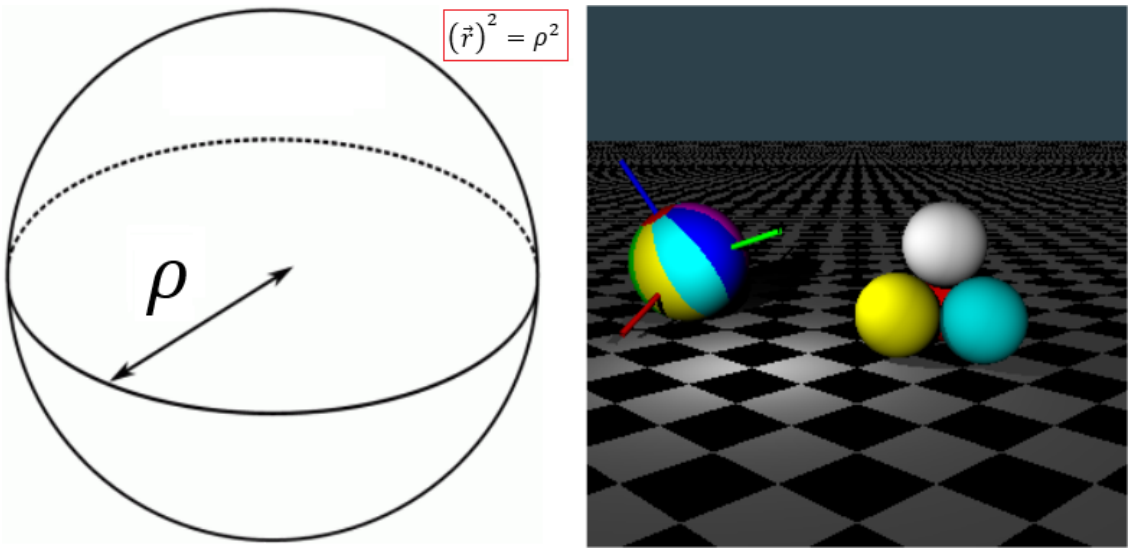


Figura 16: Equação da esfera, esfera *Beachball* e esferas de cores sólidas

Propriedades

1. BeachBall(*bool*) (Get/Set) - Indica se a esfera deve apresentar padrão multicolorido ao invés da cor sólida dada, para facilitar visualização de rotações.
2. Ro(*double*) (Get/Set) - Raio da esfera.

Métodos públicos estáticos MinPosRoot (double, double, double) (*double*) - Retorna a menor raiz positiva para a equação quadrática: $ax^2 + 2bx + c = 0$. Se não houver raiz positiva, retorna -1.

3.4.5 Cylinder

Cilindro de raio e altura dados. Vide figura 17:

$$\begin{cases} -\frac{H}{2} \leq z \leq \frac{H}{2} \\ x^2 + y^2 = \rho^2 \end{cases} \qquad \vec{N} = \begin{pmatrix} x \\ y \\ 0 \end{pmatrix}$$

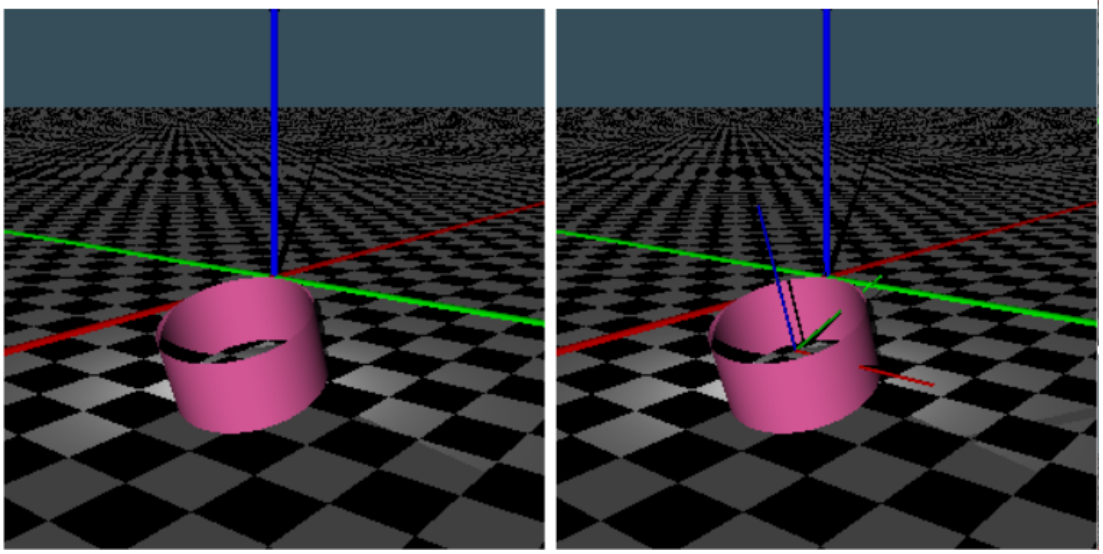


Figura 17: Equação do cilindro, e exemplo de cilindro no espaço

Propriedades

1. Ro(*double*) (Get/Set) - Raio do cilindro.
2. Height(*double*) (Get/Set) - Altura do cilindro.

3.4.6 Cone

Cone de raio e altura dados. Vide figura 18:

$$\begin{cases} 0 \leq z \leq H \\ \rho^2 z^2 = H^2(x^2 + y^2) \end{cases} \qquad \vec{N} = \begin{pmatrix} H^2 x \\ H^2 y \\ -\rho^2 z \end{pmatrix}$$

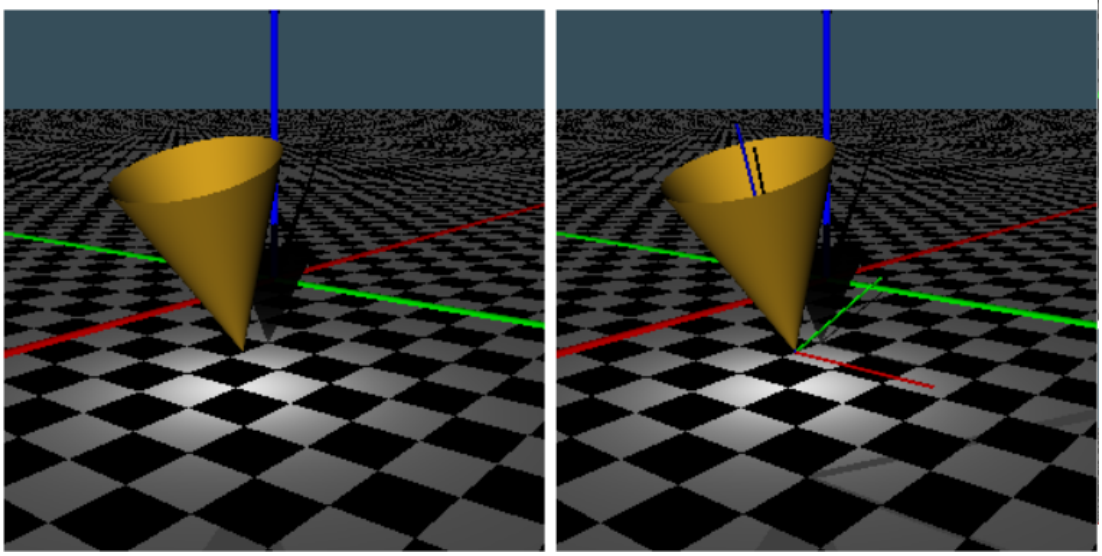


Figura 18: Equação do cone, e exemplo de cone no espaço

Propriedades

1. Ro(*double*) (Get/Set) - Raio do cone.
2. Height(*double*) (Get/Set) - Altura do cone.

3.4.7 Paraboloid

Parabolóide na forma $z = ax^2 + by^2$. Vide figura 19 que segue:

$$\begin{cases} -z_{Max} \leq z \leq z_{Max} \\ z = ax^2 + by^2 \end{cases} \qquad \vec{N} = \begin{pmatrix} ax \\ by \\ -0.5 \end{pmatrix}$$

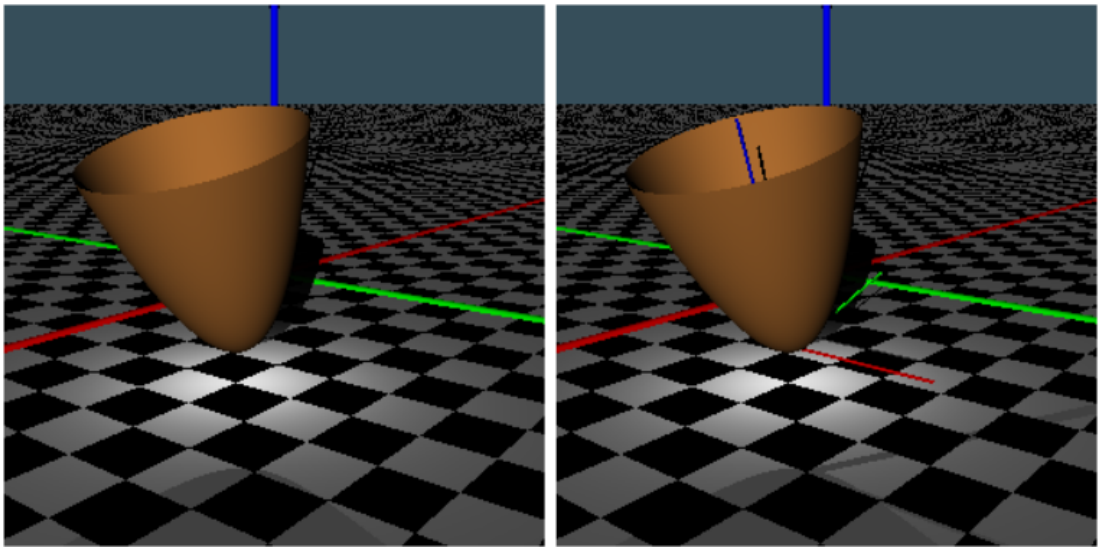


Figura 19: Equação do parabolóide, e exemplo de parabolóide no espaço

Propriedades Zmax(*double*) (Get/Set) - Valor máximo de z (em módulo).

3.4.8 Ellipsoid

Elipsóide com as larguras dos três eixos dadas $((x/a)^2+(y/b)^2+(z/c)^2 = 1)$. Vide figura 20:

$$\begin{cases} \left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 + \left(\frac{z}{c}\right)^2 = 1 \rightarrow \\ \rightarrow lx^2 + my^2 + nz^2 = p \end{cases} \qquad \vec{N} = \begin{pmatrix} lx \\ my \\ nz \end{pmatrix}$$

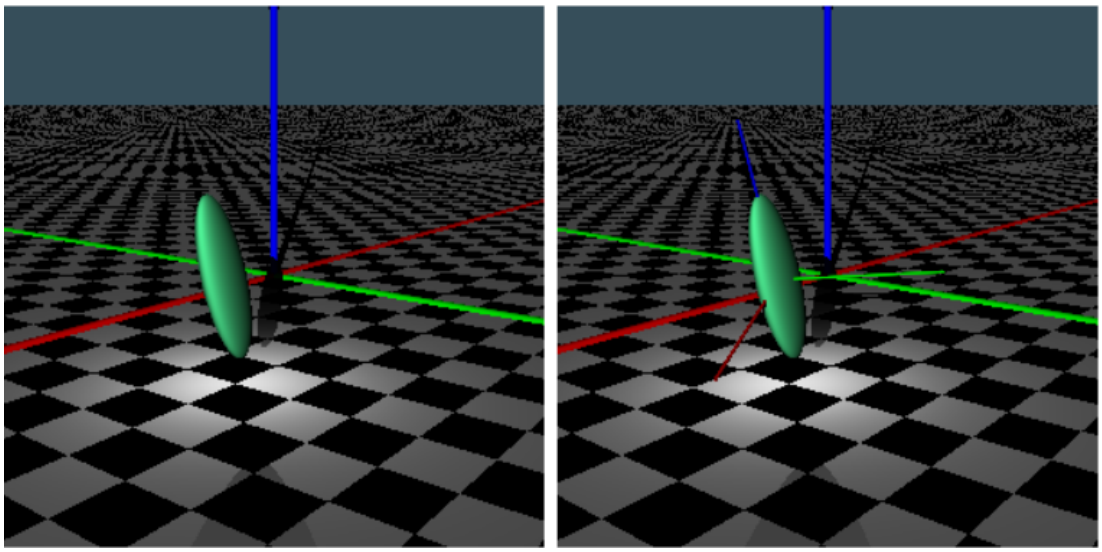


Figura 20: Equação do elipsóide, e exemplo de elipsóide no espaço

Propriedades

1. a(*double*) (Get/Set) - Valor máximo de x (em módulo). Corresponde ao dobro da largura ao longo do eixo i').
2. b(*double*) (Get/Set) - Valor máximo de y (em módulo). Corresponde ao dobro da largura ao longo do eixo j').
3. c(*double*) (Get/Set) - Valor máximo de z (em módulo). Corresponde ao dobro da largura ao longo do eixo k').

3.5 Frame

Classe para a construção da matriz de intensidades e da imagem no formato *Bitmap* a partir de cada pixel dado (Vide figura 21 abaixo).

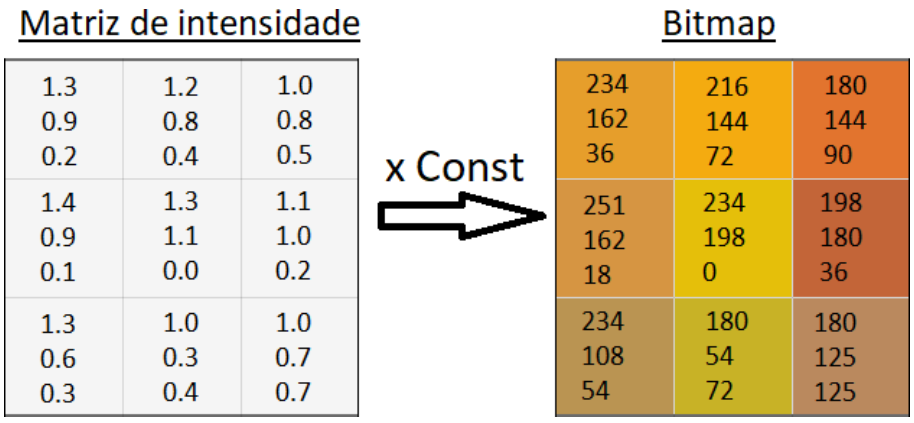


Figura 21: Conversão da matriz de intensidades em imagem *Bitmap*. No exemplo dado, a constante multiplicativa é igual à 180

Propriedades MaxIntens (*double*) (Readonly) - Maior valor na matriz de intensidades. No exemplo dado na figura 21, MaxIntens = 1.4.

Métodos públicos

1. Reflect () (*void*) - Reflete o objeto ao redor do eixo j'
2. WritePixel (int, int, Vector) (*void*) - Escreve o vetor de intensidades na posição dada na matriz.
3. Picture (double) (*Bitmap*) - Converte a matriz de intensidades numa imagem no formato *Bitmap*, a partir de uma constante multiplicativa dada (vide figura 21 acima). A constante multiplicativa que dá a maior abrangência de cores na imagem é 255/MaxIntens.

3.6 Camera:Movable

Classe para a inicialização e posicionamento da câmera *pinhole* que recebe a imagem dos objetos e fontes de luz espalhados pelo espaço, a partir de um algoritmo de *ray tracing* feito pixel a pixel. (Vide figura 22 abaixo). O filme fica sempre a distância unitária da posição da câmera, ao longo do eixo i'.

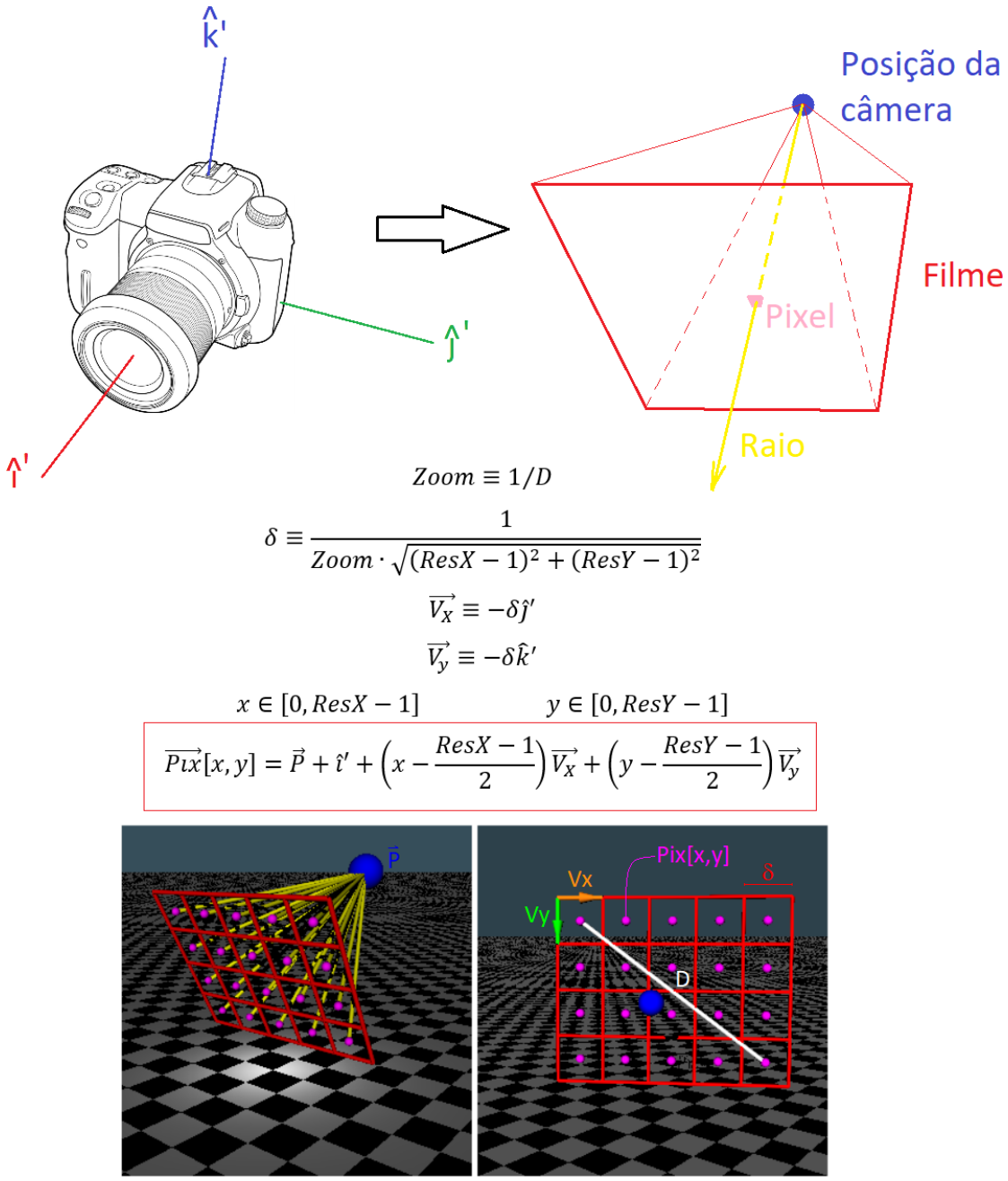


Figura 22: Diagrama da câmera. No exemplo dado a resolução é 5x4 pixels

Propriedades

1. ResX (*int*) (Get/Set) - Número de pixels no eixo horizontal da imagem (eixo j').
2. ResY (*int*) (Get/Set) - Número de pixels no eixo vertical da imagem (eixo k').
3. Zoom (*double*) (Get/Set) - Inversamente proporcional ao tamanho da diagonal do rêtangulo onde a imagem é formada (vide figura 22).

Métodos públicos Frame (List:OpaqueObject, List:LightSource , Vector) (*Frame*) - Cria a matriz de intensidades a partir dos objetos e fontes de luz dados. O último input é a intensidade *default* para pixels que não recebem imagem de nenhum objeto.

4 Bibliografia

[1] https://docs.microsoft.com