**Camera:**

De maneira a conseguir uma aplicação o mais modular possível decidimos que a manipulação da camera seria feita numa classe à parte da *engine*. Assim sendo criamos uma classe *Camera* que tinha as seguintes funções:

1. Movimento horizontal (esquerda e direita)
2. Movimento vertical (para cima e para baixo)
3. Aproximação/afastamento
4. Velocidade nos movimentos regulável.

Para que este componente conseguisse realizar estas manipulações precisava de guardar os seguintes parâmetros:

* Distancia à origem (raio)
* Ponto com as coordenadas atuais da camera
* Ângulo para a movimentação horizontal (alpha)
* Ângulo para a movimentação vertical (beta)
* Valor que define a velocidade

Após definirmos os parâmetros necessários construímos os métodos necessários. Como o tratamento das teclas pressionadas é feito na classe *engine*, nesta classe apenas precisávamos de alterar as coordenadas da posição da câmera.

Explicaremos de seguida 2 métodos uma vez que os seguintes são construídos utilizando a mesma lógica.

Movimento vertical:

void Camera :: rodarCima() {

beta +=0.1\*velocidade;

}

Quando a tecla *Up* é pressionada/rato faz movimento vertical a engine invoca este método que irá aumentar o ângulo beta responsável por esse mesmo movimento.

Aproximar a camera:

void Camera::afastarCamera() {

raio = raio + 0.5f;

}

Este método é utilizado para encurtar a distância da posição atual à origem (foco).

Posição atual:

Ponto\* Camera::getPosicao() {

float camX = raio \* cos(beta) \* sin(alpha);

float camY = raio \* sin(beta);

float camZ = raio \* cos(beta) \* cos(alpha);

posicao->setX(camX);

posicao->setY(camY);

posicao->setZ(camZ);

return posicao;

}

Este método é constantemente chamado na render scene da engine e devolve a posição atual da câmera após todas as alterações feitas, ou seja, com as teclas/rato estamos constantemente a alterar os parâmetros da camera e consequentemente a sua posição no entanto essa posição só é realmente atualizada quando a imagem é impressa ( só aqui é que é importante -> este “entre parêntesis” vai ser para por?).