Computação Gráfica e Interfaces

2017-2018
Fernando Birra

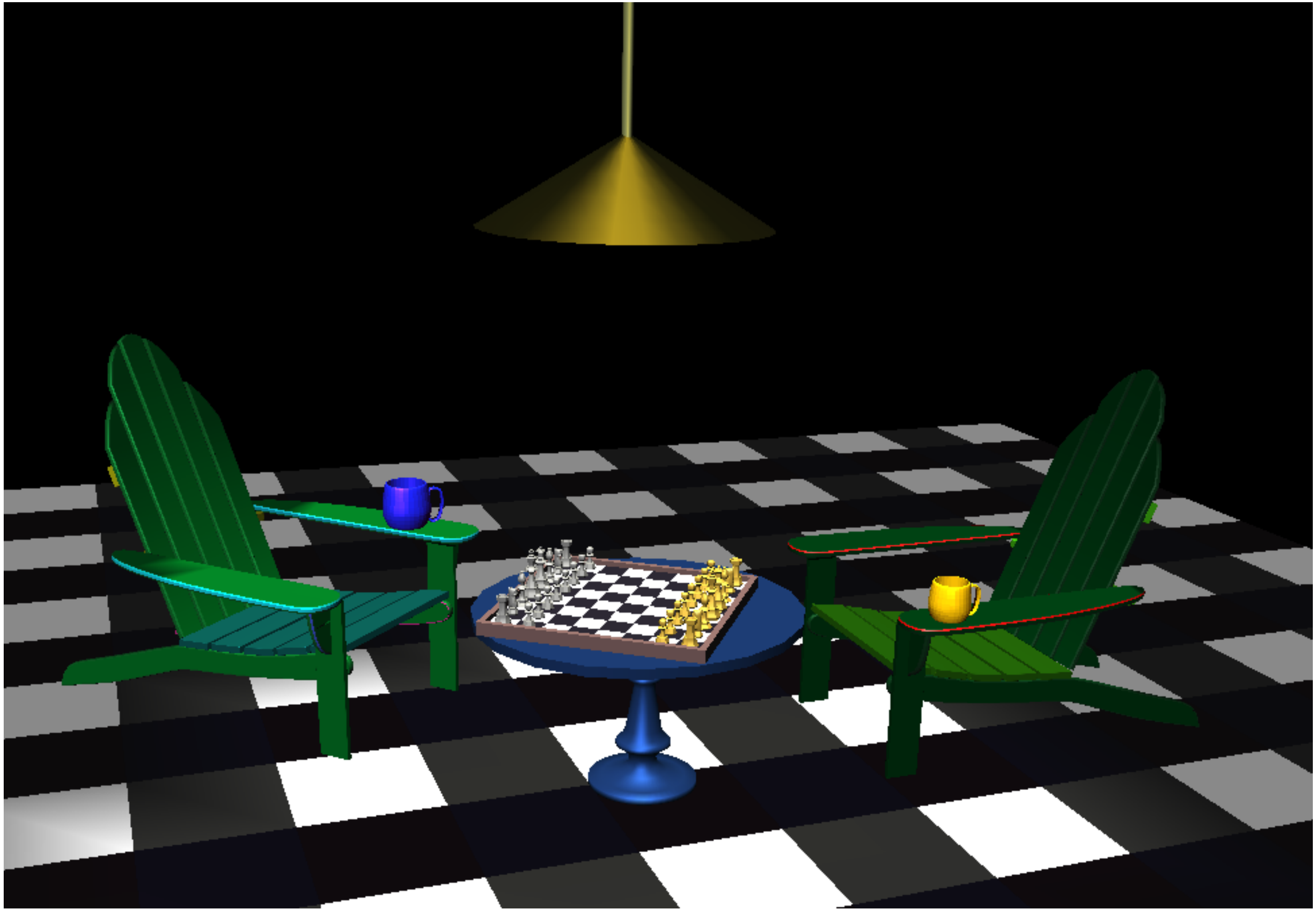
Modelação Hierárquica

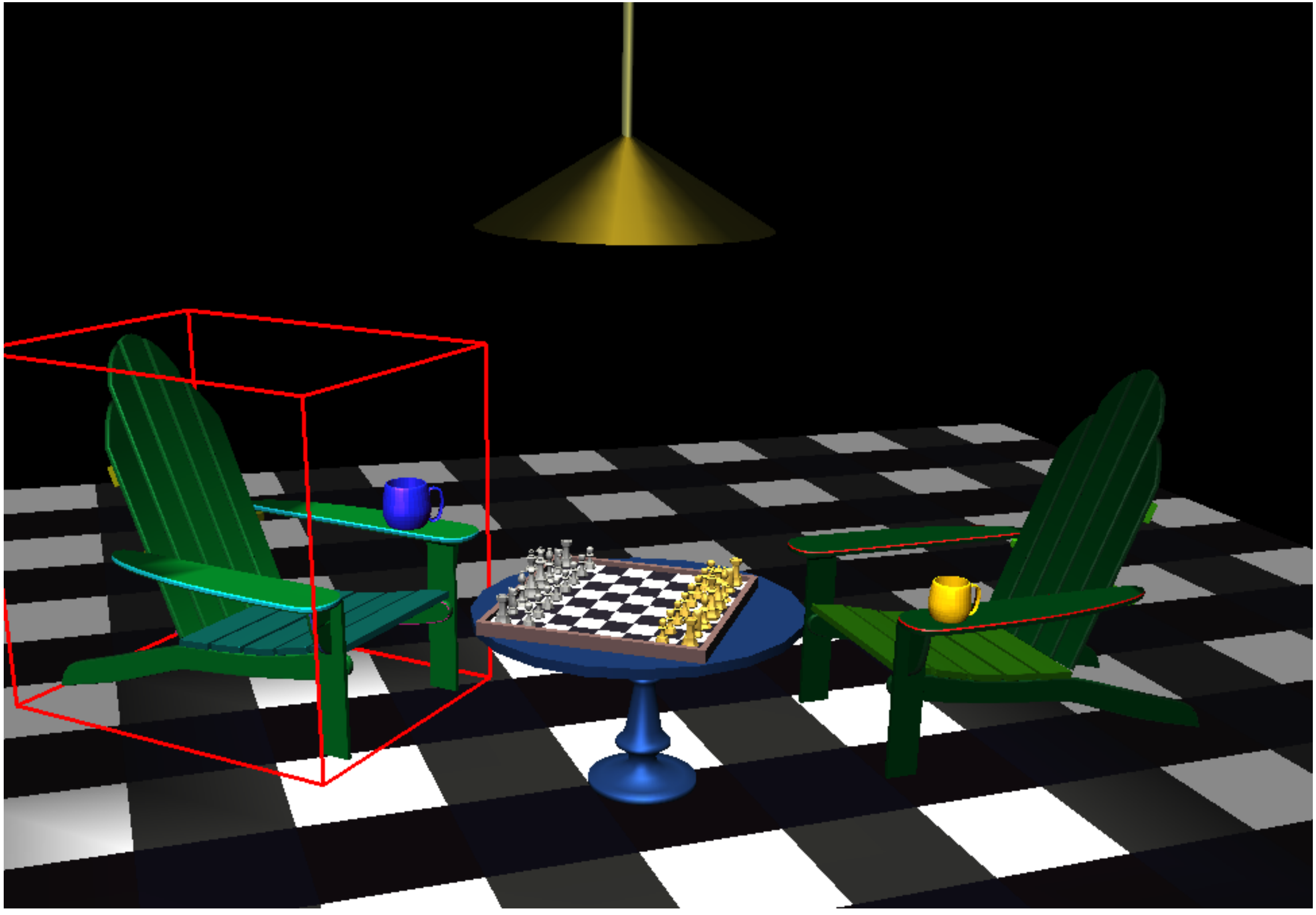
2017-2018

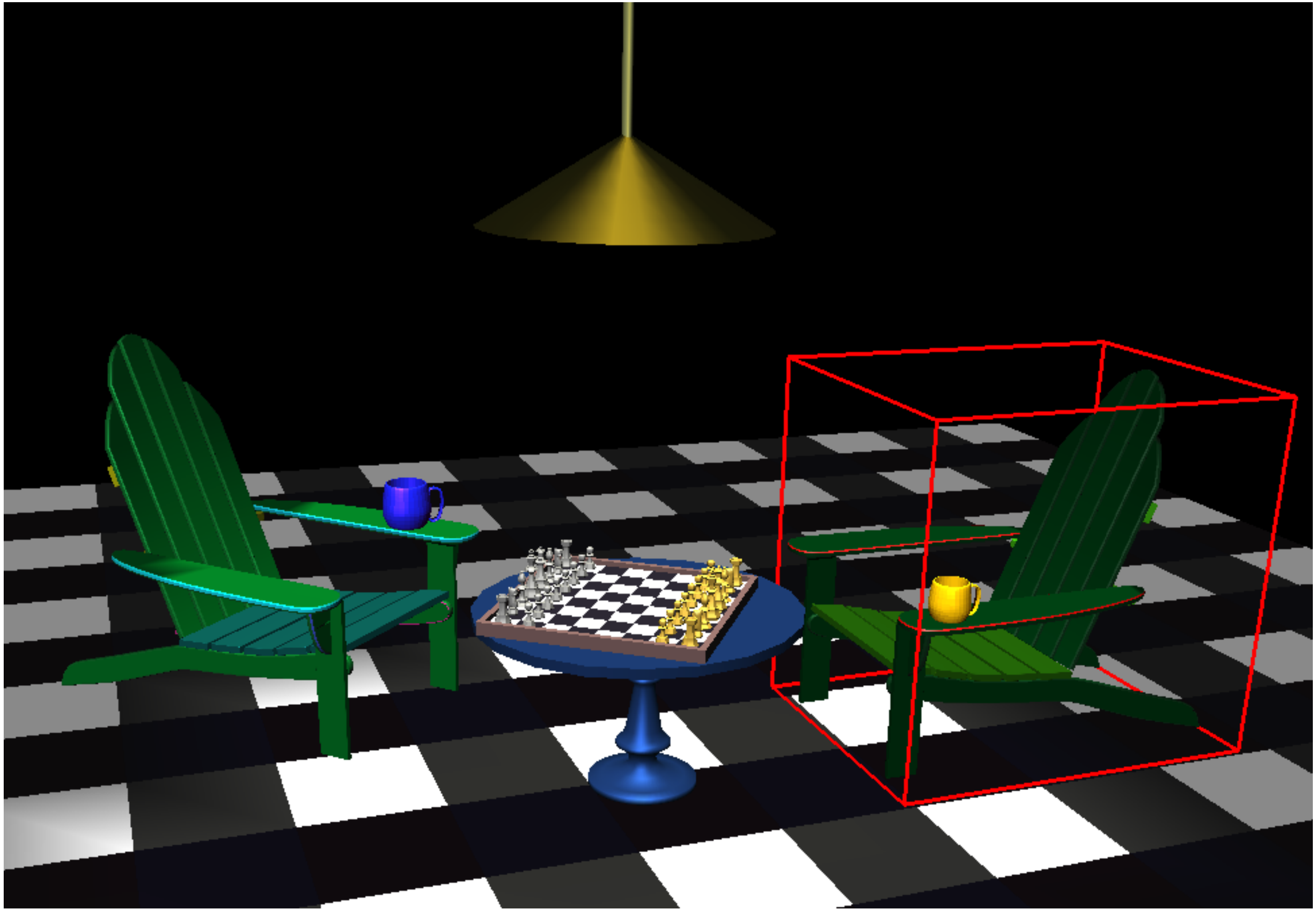
Fernando Birra

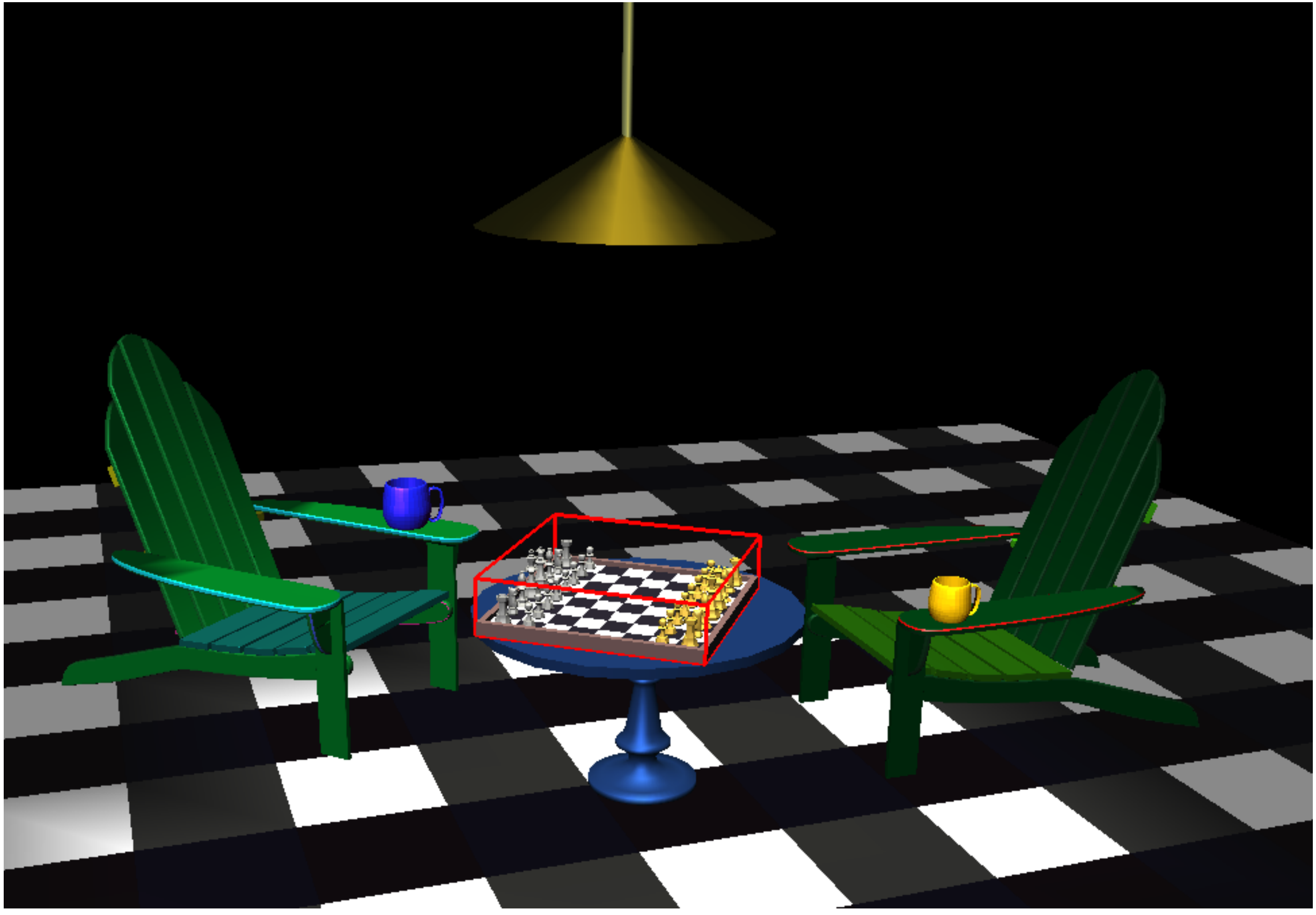
Objetivos

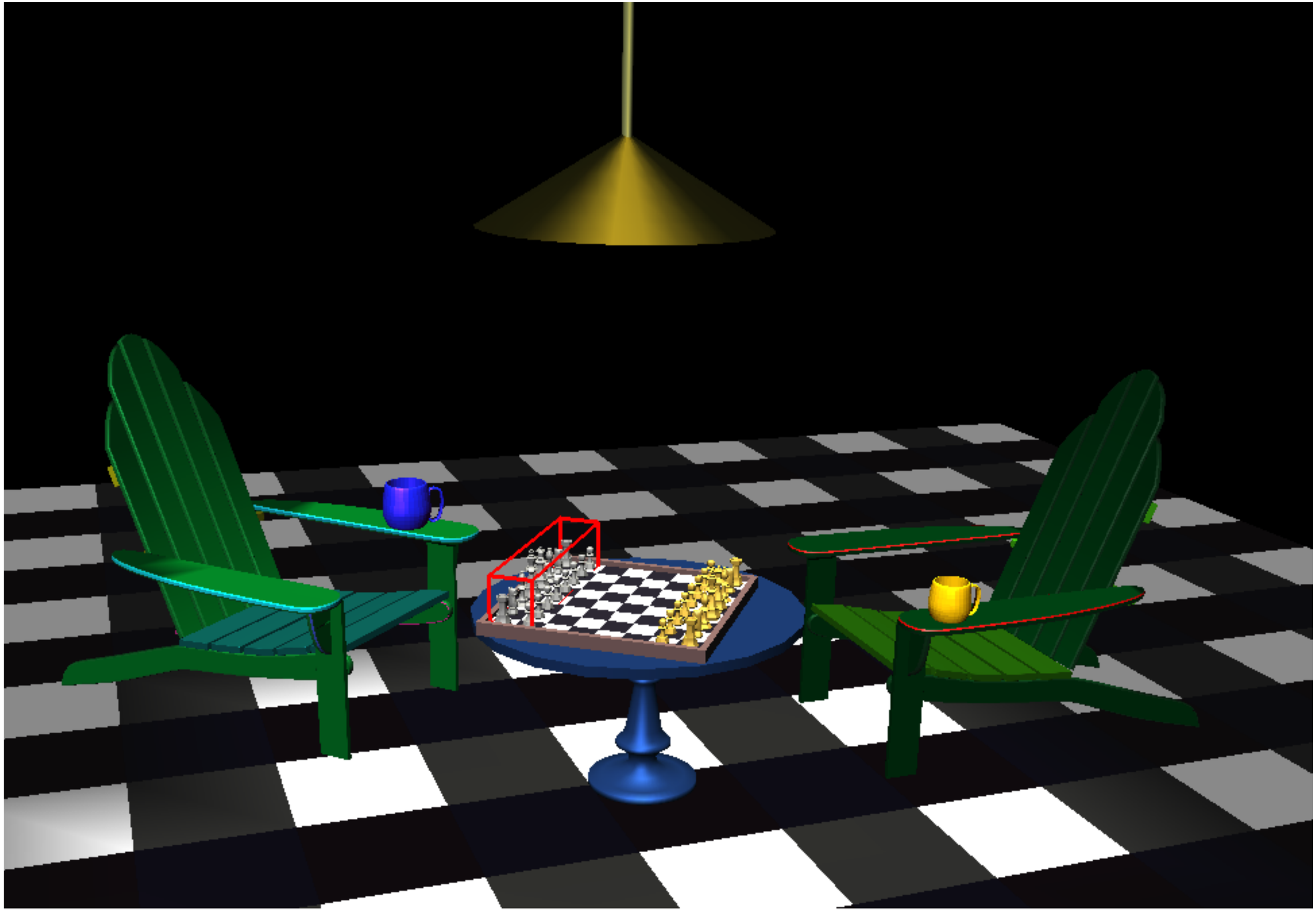
- Perceber as vantagens da modelação hierárquica de cenas
- Perceber a representação em forma de grafo da cena
- Saber escrever o código para a visualização duma cena descrita por um grafo e vice-versa.

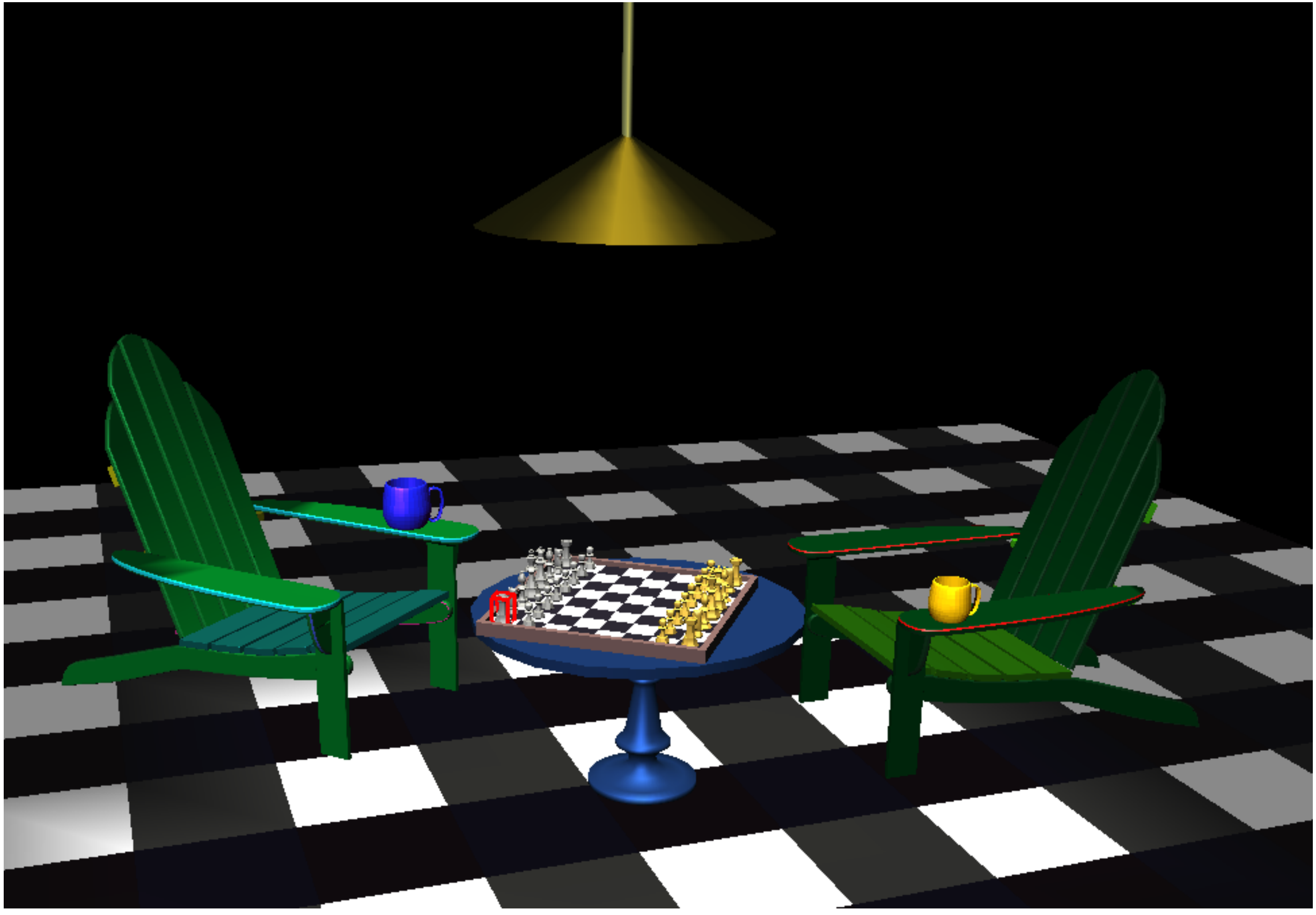






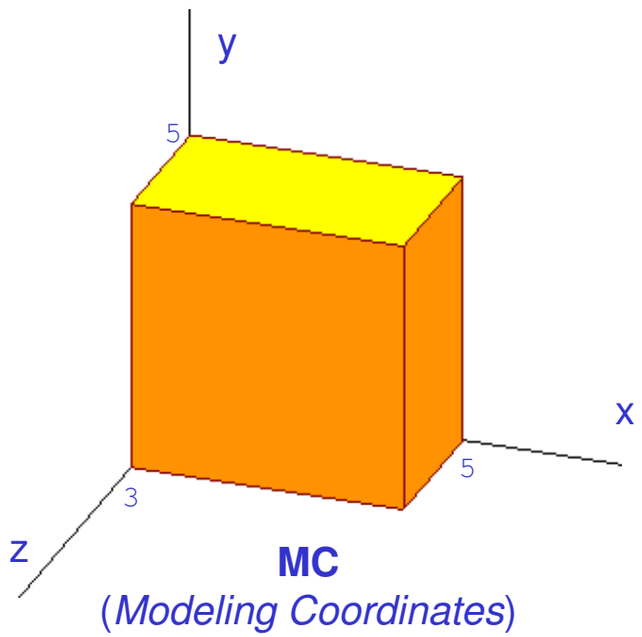




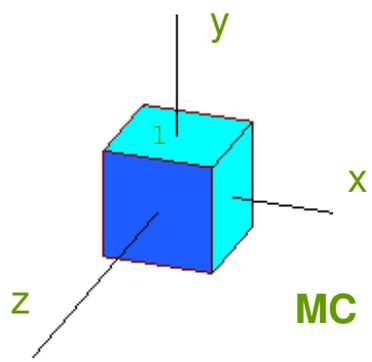


Sistemas de Coordenadas Locais, de Modelação ou do Objeto

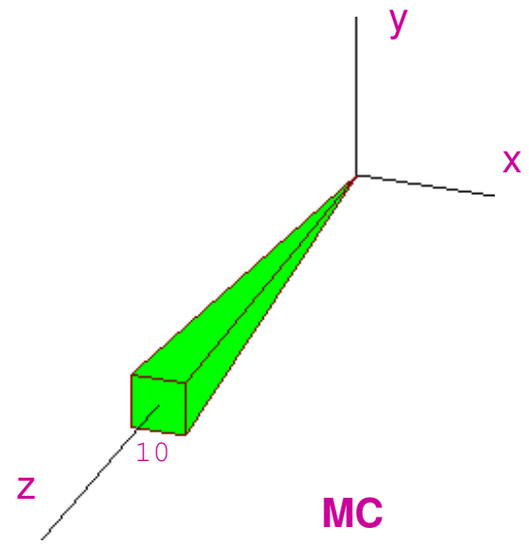
- Cada objeto primitivo tem o seu próprio referencial, no qual foi feita a sua modelação.



MC
(*Modeling Coordinates*)



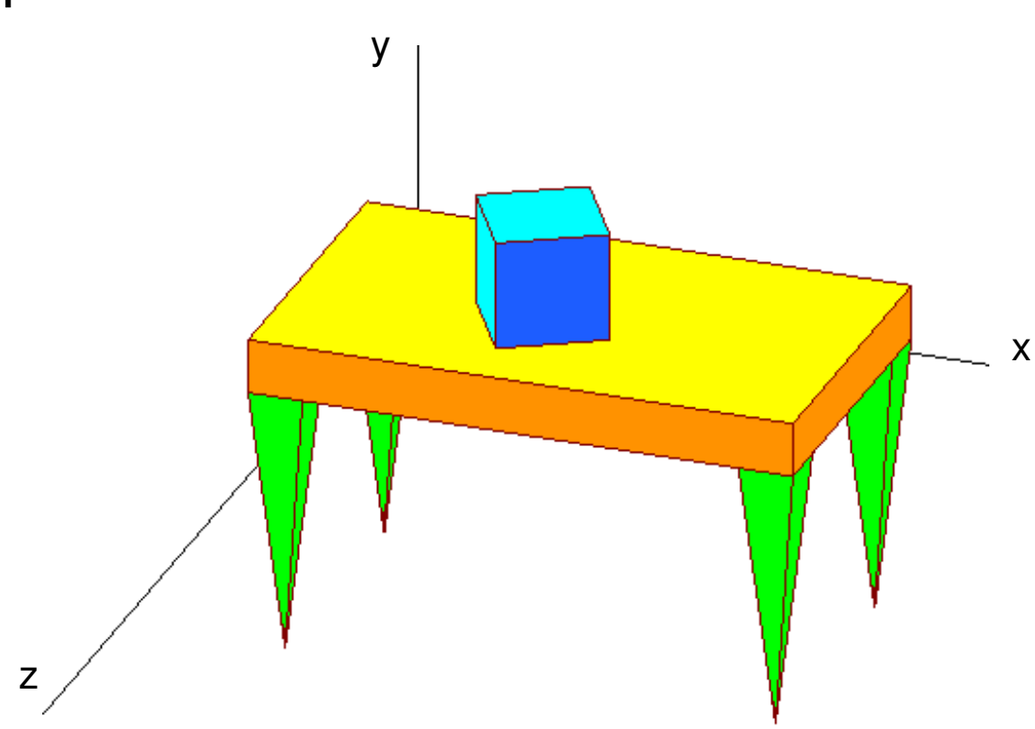
MC



MC

Sistema de Coordenadas do Mundo (Globais)

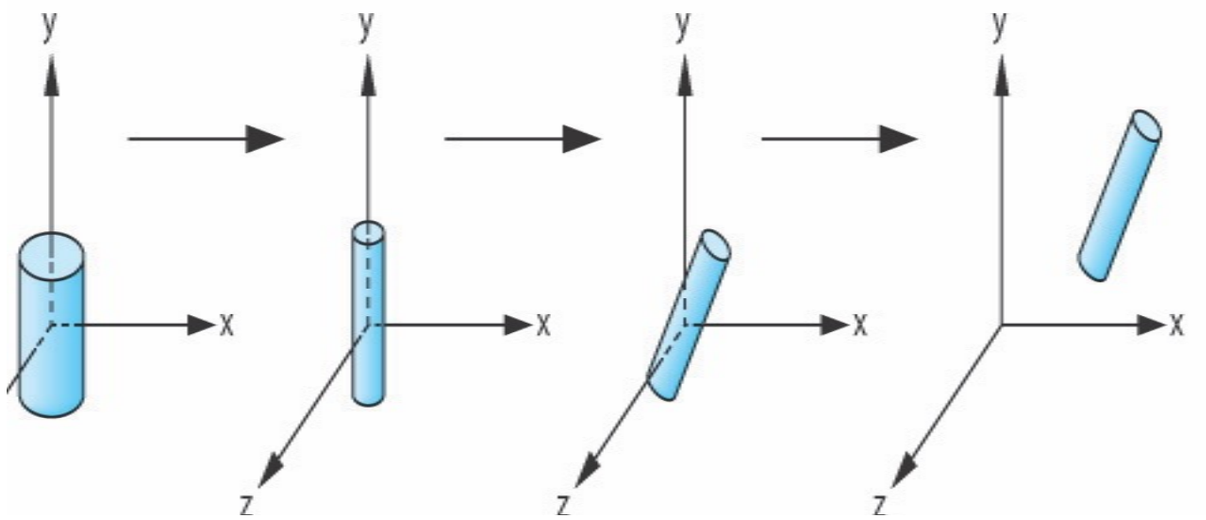
- A modelação duma cena consiste em aplicar aos objetos primitivos um conjunto de transformações geométricas para os instanciar na cena



WC
(*World Coordinates*)

Composição de Transformações

- Uma solução consiste em pensar isoladamente em cada instância de objeto primitivo e determinar isoladamente, uma composição de transformações do tipo: T.R.S



Como cada instância é pensada de forma isolada, não é fácil modificar seletivamente a cena. Exemplo: mudar o tamanho ou a posição da mesa do slide anterior.

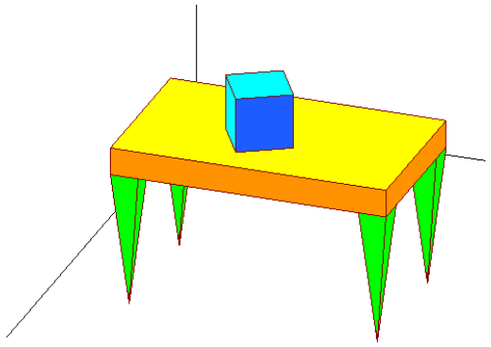
Composição de Transformações

- A alternativa consiste em modelar a cena de forma hierárquica. Agrupando instâncias de objetos primitivos, eventualmente já transformados, para assim formar objetos mais complexos.

- Estes objetos complexos poderão ser também eles transformados e agregados a outros.

- No final, ter-se-á apenas um conjunto final, representando toda a cena.

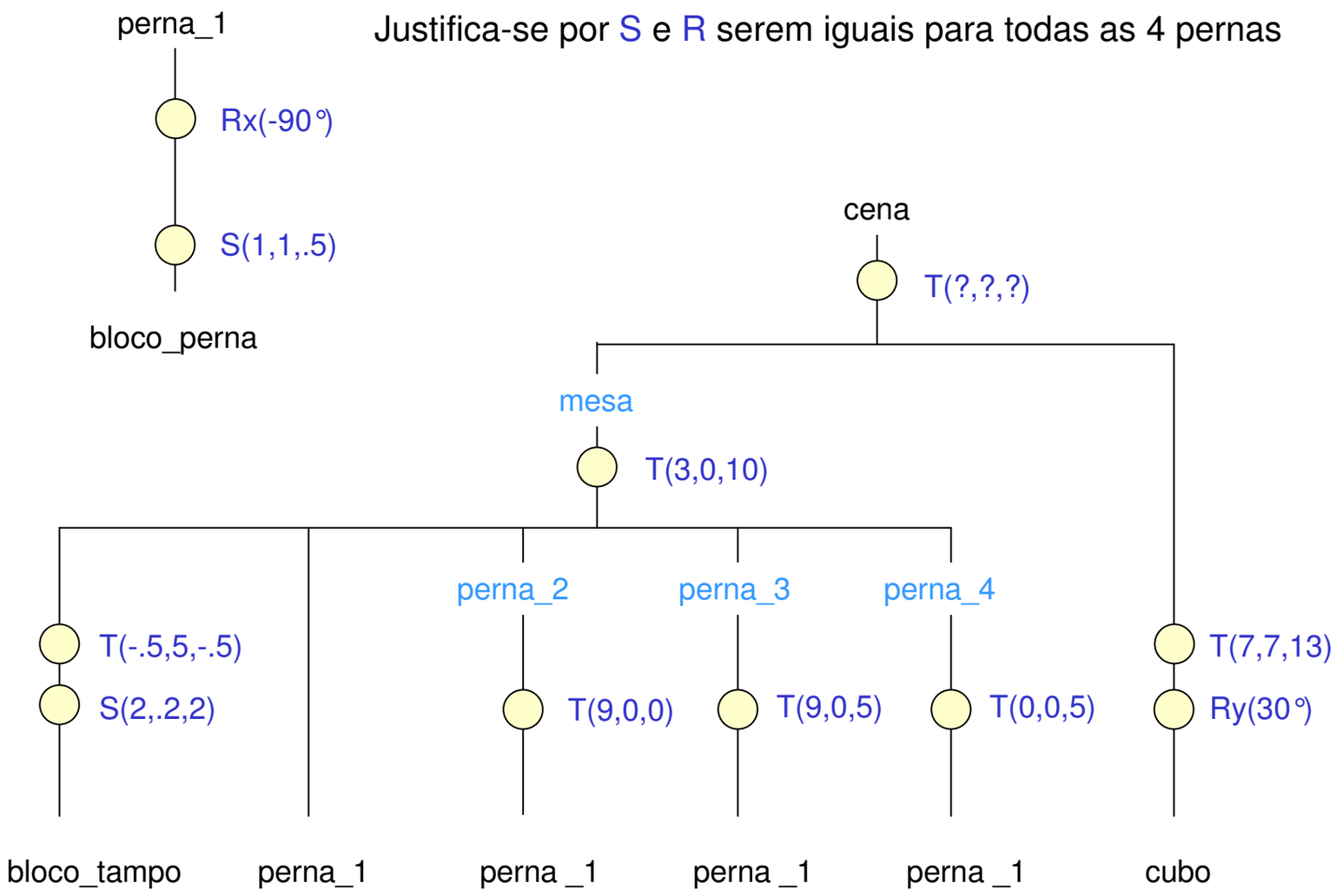
Grafo da Cena



M.Próspero

Grafo da Cena

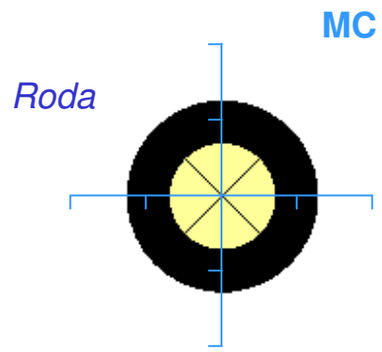
Uma alternativa possível, com base na análise do modelo, usando-se um subgrafo auxiliar:

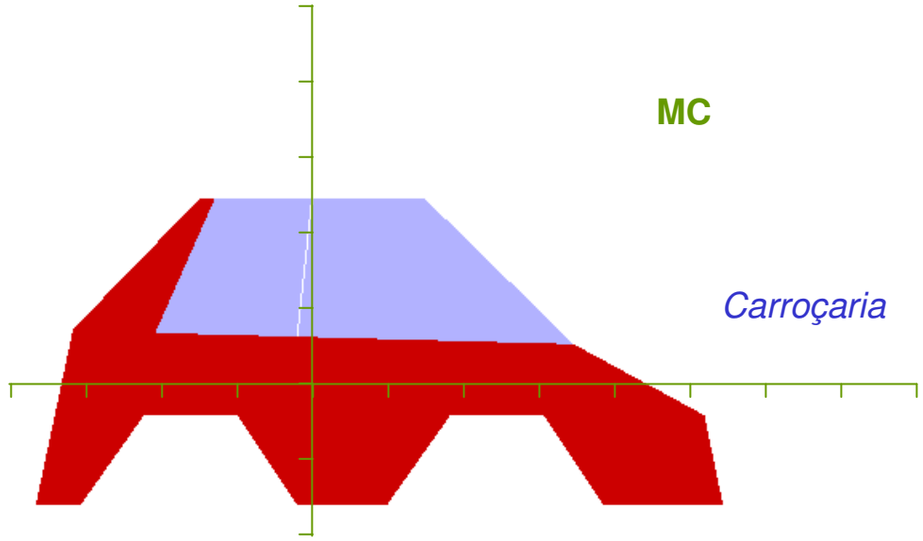


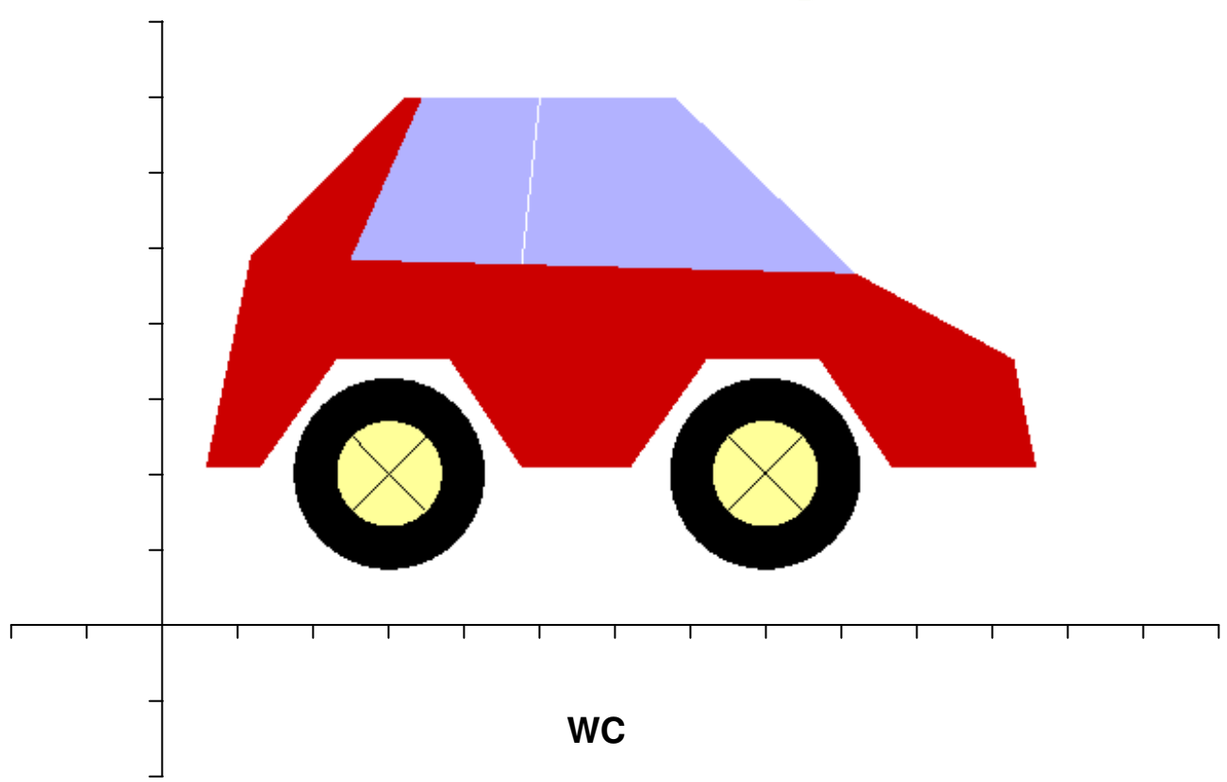
M.Próspero

Exemplo - Carro

Exemplo de um carro em 2D



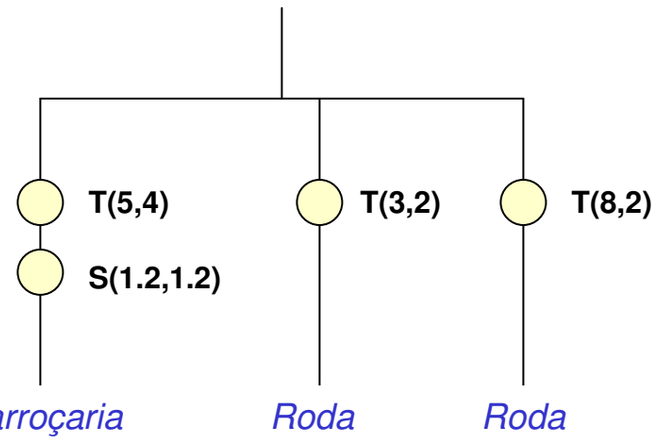




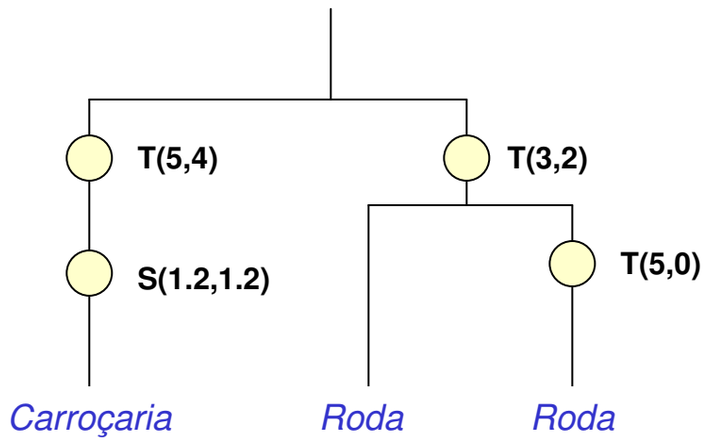
M.Próspero

Exemplo - Carro

GRAFO DA CENA



Solução alternativa:

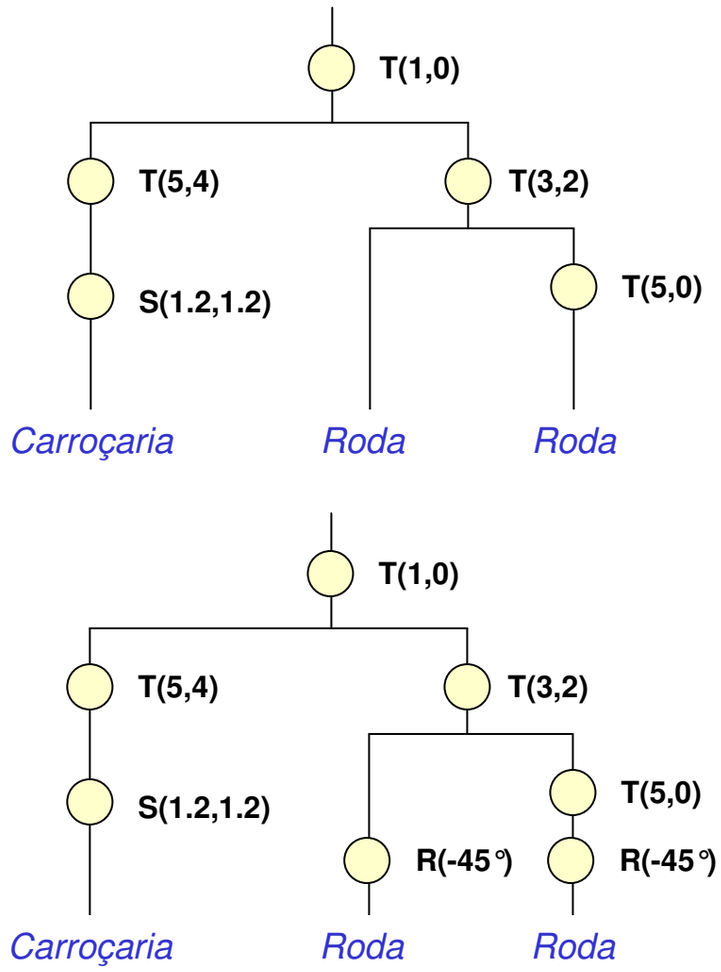


Como fazer avançar o automóvel de 1 unidade?

M.Próspero

Exemplo - Carro

GRAFO DA CENA



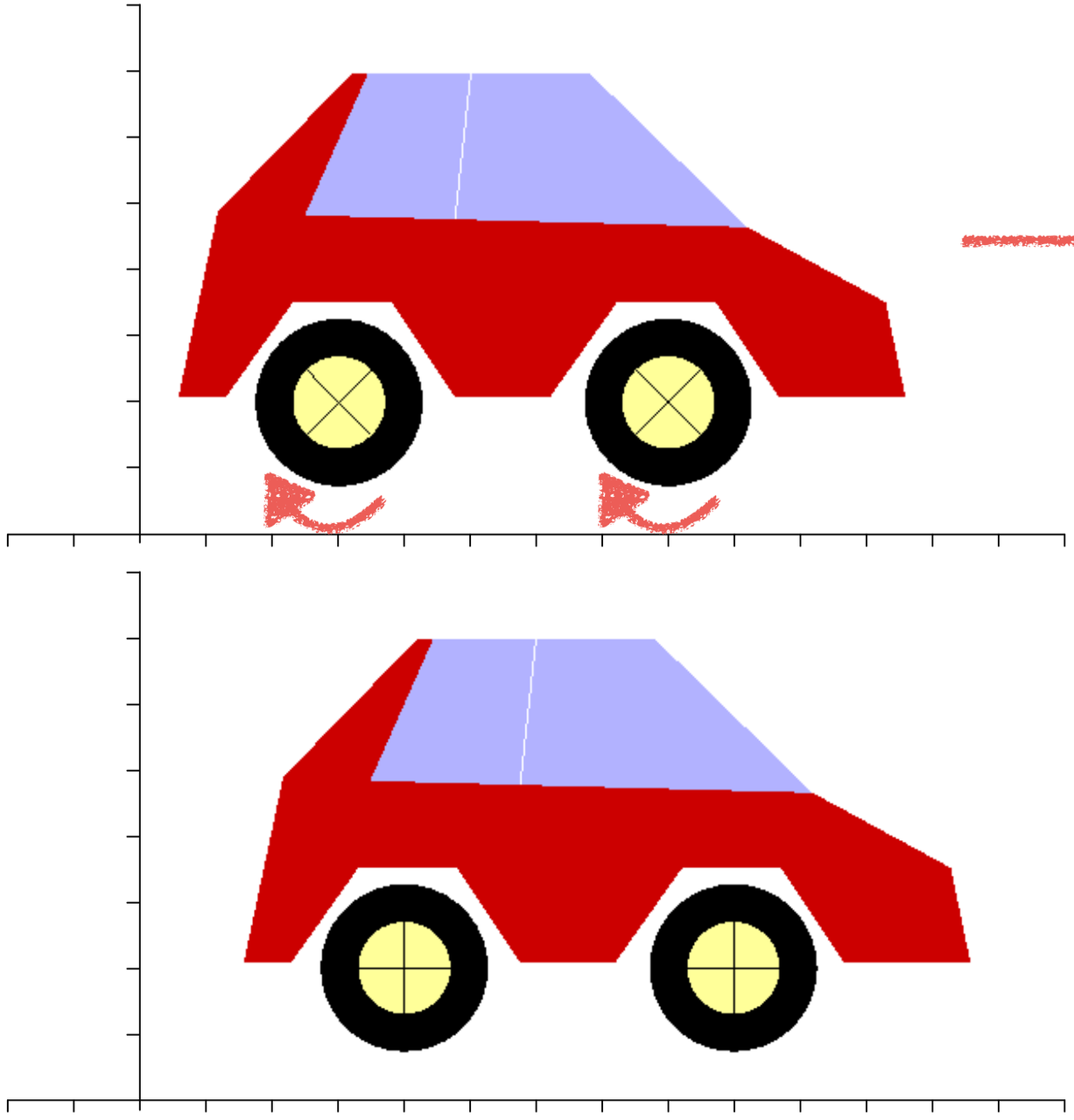
E como fazer girar as rodas, de acordo com esse movimento?

Poder-se-ia construir um subgrafo com a roda e a rotação

Exercício proposto: Qual seria a diferença no caso de um trator agrícola?

M.Próspero

Exemplo - Carro



com rotação das rodas

M.Próspero

Pilha de transformações

- As transformações que afetam um determinado ramo do grafo (uma primitiva) são aplicadas à respetiva primitiva pela ordem obtida lendo as transformações de baixo para cima
- Em cada ramificação, todas as transformações desde a raiz têm efeito sobre todos os ramos descendentes desse ponto.
- Seja M a composição de transformações M_1, M_2, …, M_n, encontradas num percurso descendente, desde a raiz até uma primitiva, então $M=M_1.M_2…M_n$ e, para um ponto P, dessa primitiva, o ponto transformado será $M.P = M_1.M_2…M_n.P$

Pilha de transformações

- A geração do código poderá ser feita através dum percurso prefixado no grafo, acumulando na matriz ModelView, por multiplicação à direita, cada uma das transformações que se vão encontrando.

- Nos pontos de bifurcação, poderemos preservar o valor atual da matriz ModelView empilhando-a numa pilha de transformações

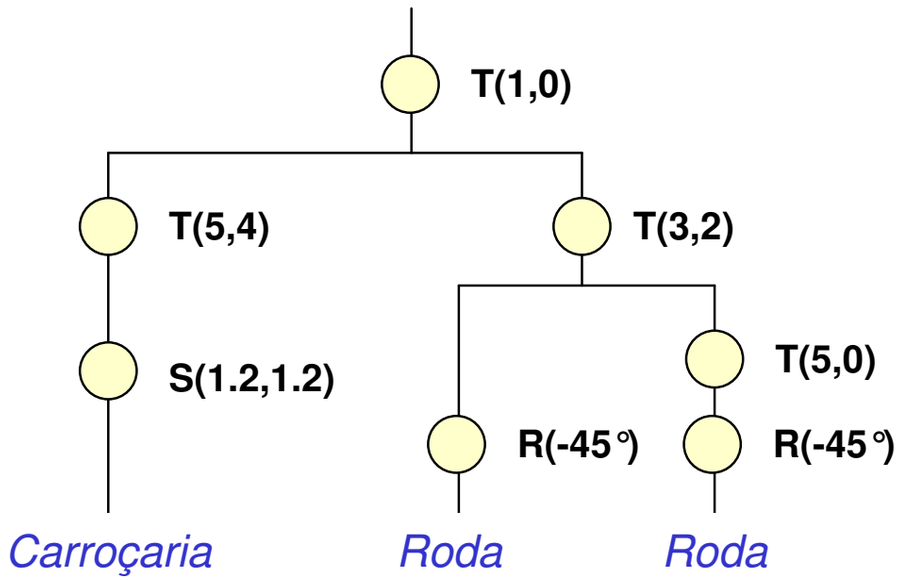
- Num determinado nível do grafo, ao regressar-se dum ramo, recupera-se o valor anterior da matriz ModelView desempilhando-o da pilha, antes de prosseguir para um ramo irmão.

Tradução para pseudo código (direta)

A inicialização de MV é geralmente efetuada chamando a função lookAt (Mview)

MV representa a matriz ModelView corrente. Presume-se que cada invocação duma primitiva usa a matriz MV para transformar os vértices.

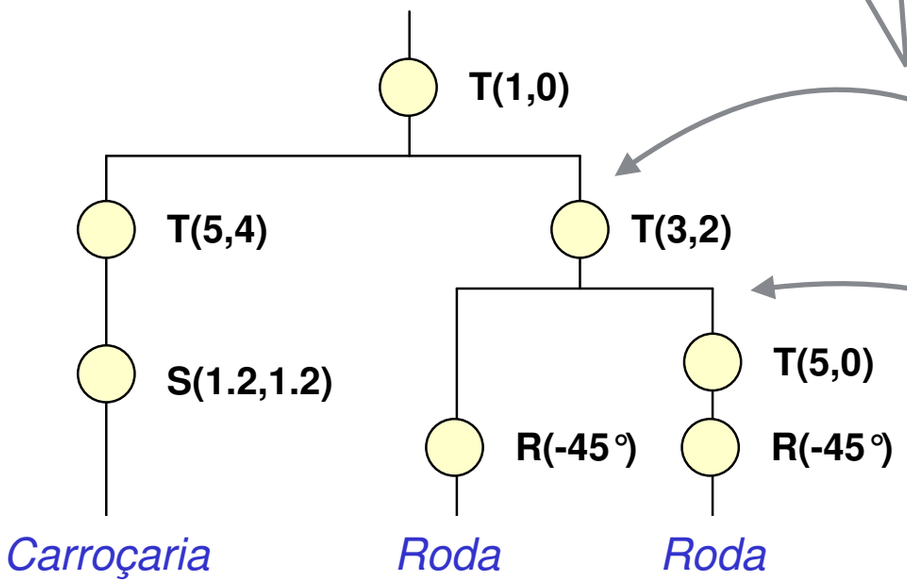
```
MV = I
MV = MV.T(1,0)
Push(MV)
    MV = MV.T(5,4)
    MV = MV.S(1.2, 1.2)
    Carroçaria
MV = Pop()
Push(MV)
    MV = MV.T(3,2)
    Push(MV)
        MV = MV.R(-45)
        Roda
    MV = Pop()
    Push(MV)
        MV = MV.T(5,0)
        MV = MV.R(-45)
        Roda
    MV = Pop()
MV = Pop()
```



(Travessia prefixada do grafo)

Tradução para pseudo código (otimizada)

Não há necessidade de preservar a composição de transformações atual, contida na matriz ModelView, sempre que se vai descer pelo ramo mais à direita!



(Travessia prefixada do grafo)

```
MV = I
MV = MV.T(1,0)
Push(MV)
    MV = MV.T(5,4)
    MV = MV.S(1.2, 1.2)
    Carroçaria
MV = Pop()
Push(MV)
    MV = MV.T(3,2)
    Push(MV)
        MV = MV.R(-45)
        Roda
    MV = Pop()
    Push(MV)
        MV = MV.T(5,0)
        MV = MV.R(-45)
        Roda
    MV = Pop()
MV = Pop()
```

Tradução para código WebGL (com MV.js)

```
MV = I
MV = MV.T(1,0)
Push(MV)
    MV = MV.T(5,4)
    MV = MV.S(1.2, 1.2)
    Carroçaria
MV = Pop()
~~Push(MV)~~
    MV = MV.T(3,2)
    Push(MV)
        MV = MV.R(-45)
        Roda
    MV = Pop()
    ~~Push(MV)~~
        MV = MV.T(5,0)
        MV = MV.R(-45)
        Roda
    ~~MV = Pop()~~
~~MV = Pop()~~
```

```
var matrixStack=[];

modelView = mat4();
modelView = mult(modelView, translate(1,0,0));
matrixStack.push(modelView);
  modelView = mult(modelView, translate(5,4,0);
  modelView = mult(modelView, scalem(1.2,1.2,1);
  Carroçaria(modelView);
modelView = matrixStack.pop();
modelView = mult(modelView, translate(3,2,0));
matrixStack.push(modelView);
  modelView = mult(modelView, rotateZ(-45));
  Roda(modelView);
modelView = matrixStack.pop();
modelView = mult(modelView, translate(5,0,0));
modelView = mult(modelView, rotateZ(-45));
Roda(modelView);
```

Nota: As primitivas, aqui representadas por funções, terão que enviar ao vertex shader o valor atual da matriz modelView, para que os respetivos pontos que as compõem possam ser transformados corretamente.

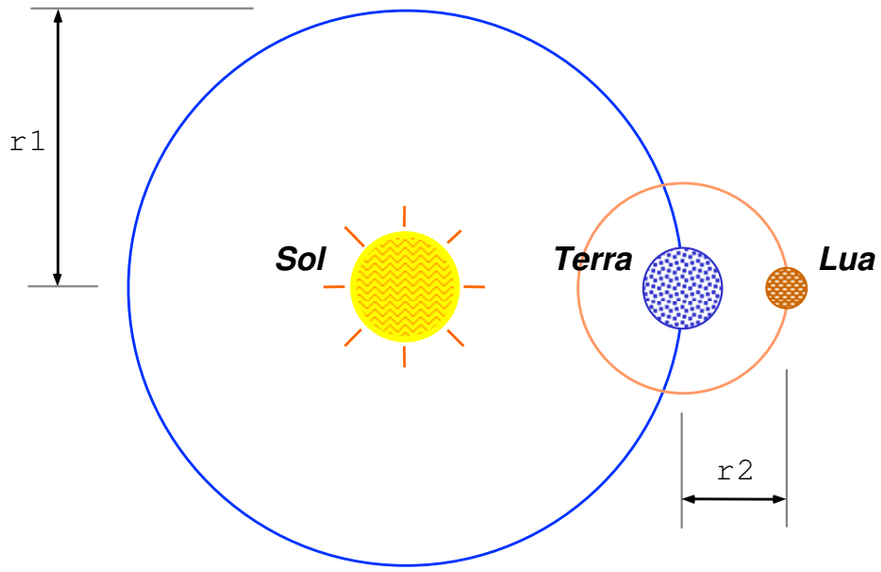
Exemplo de funções auxiliares para usar com MV.js

```
var matrixStack = [];
var modelView = mat4();

// Stack related operations
function pushMatrix() {
    mat4 m(modelView[0], modelView[1],
           modelView[2], modelView[3]);
    matrixStack.push(m);
}
function popMatrix() {
    modelView = matrixStack.pop();
}
// Append transformations to modelView
function multMatrix(m) {
    modelView = mult(modelView, m);
}
function multTranslation(t) {
    modelView = mult(modelView, translate(t));
}
function multScale(s) {
    modelView = mult(modelView, scalem(s));
}
function multRotationX(angle) {
    modelView = mult(modelView, rotateX(angle);
}
function multRotationY(angle) {
    modelView = mult(modelView, rotateY(angle);
}
function multRotatioZ(angle) {
    modelView = mult(modelView, rotateZ(angle));
}
```

```
}

multTranslation([1,0,0]));
pushMatrix();
  multTranslation(5,4,0);
  multScale(1.2,1.2,1);
  Carroçaria(modelView);
popMatrix();
multTranslation(3,2,0));
pushMatrix();
  multRotationZ(-45));
  Roda(modelView);
popMatrix();
multTranslation(5,0,0));
multRotationZ(-45));
Roda(modelView);
```

Exemplo anterior, refeito usando as funções auxiliares aqui definidas.

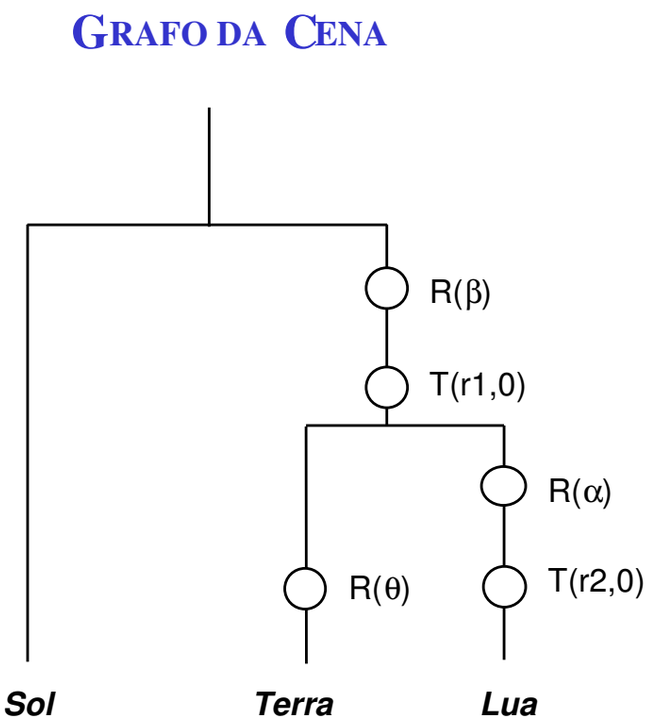
Exemplo - Sistema Planetário Simplificado



Admite-se que:

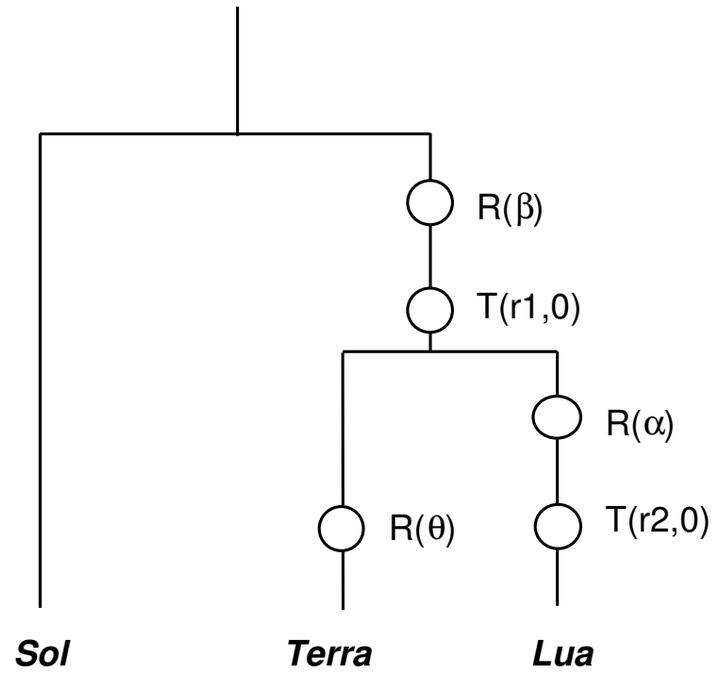
- Os objetos primitivos (2D) serão modelados com centro na origem.
- As órbitas dos planetas são circulares e encontram-se no plano XY.

Colocado na origem de WC



M.Próspero

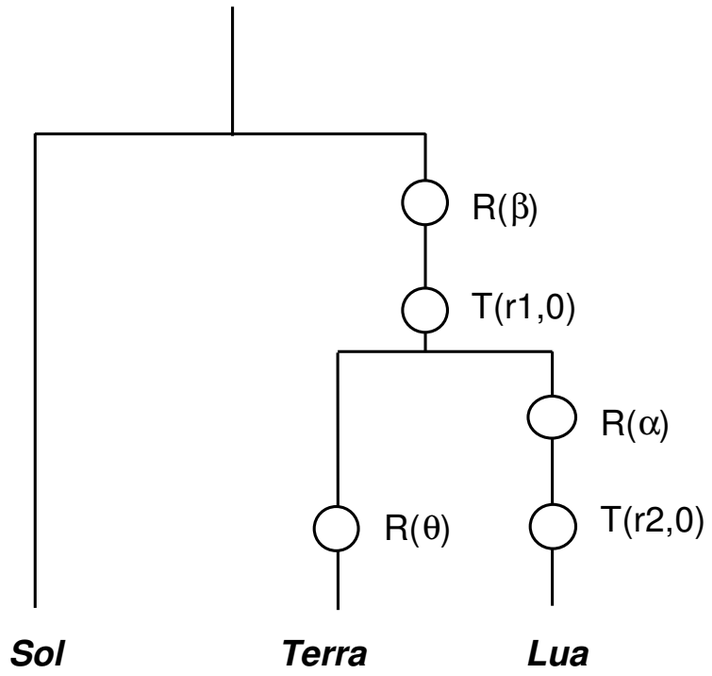
Tradução direta




```
pushMatrix();
  pushMatrix();
    Sol(modelView);
  popMatrix();
  pushMatrix();
    multRotationZ(beta);
    multTranslation([r1,0,0]);
    pushMatrix();
      multRotationZ(theta);
      pushMatrix();
        Terra(modelView);
      popMatrix();
    popMatrix();
    pushMatrix();
      multRotationZ(alpha);
      multTranslation([r2,0,0]);
      pushMatrix();
        Lua(modelView);
      popMatrix();
    popMatrix();
  popMatrix();
popMatrix();
```

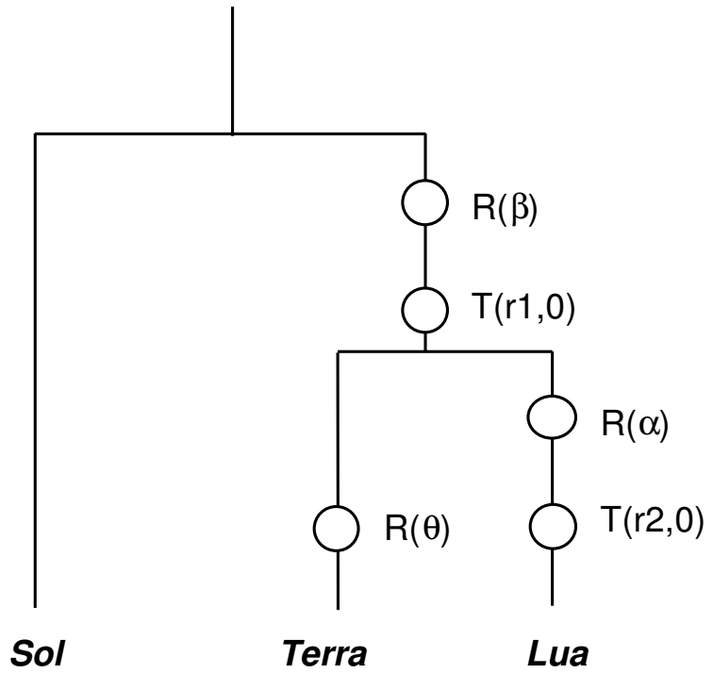
Esta solução foi escrita supondo que **nada** se conhece sobre a programação dos objetos primitivos, não sendo de excluir que aí possam ter sido utilizadas **transformações geométricas** (ver as instruções Push e Pop como parênteses de cada um desses objetos).

Tradução otimizada




```
pushMatrix();
  pushMatrix();
    Sol(modelView);
  popMatrix();
popMatrix();
pushMatrix();
  multRotationZ(beta);
  multTranslation(r1,0,0);
  pushMatrix();
    multRotationZ(theta);
    pushMatrix();
      Terra(modelView);
    popMatrix();
  popMatrix();
  pushMatrix();
    multRotationZ(alpha);
    multTranslation(r2,0,0);
    pushMatrix();
      Lua(modelView);
    popMatrix();
  popMatrix();
popMatrix();
```

Tradução otimizada




```
pushMatrix();
  Sol(modelView);
popMatrix();
multRotationZ(beta);
multTranslation(r1,0,0);
pushMatrix();
  multRotationZ(theta);
  Terra(modelView);
popMatrix();
multRotationZ(alpha);
multTranslation(r2,0,0);
Lua(modelView);
```