LNA LABORATÓRIO NACIONAL DE ASTROFÍSICA

Notas Técnicas do Laboratório Nacional de Astrofísica

Telescópio em ambiente virtual: representação digital e estudo de movimento do Perkin-Elmer 1,60m

Orlindo Wagner Soares Pereira

Ramon Carlos Gargalhone

LNA-NT-2025-28

Jan/2025

Telescópio em ambiente virtual: representação digital e estudo de movimento do Perkin-Elmer 1,60m

Orlindo Wagner Soares Pereira [1]

Ramon Carlos Gargalhone [2]

[1] *Laboratório Nacional de Astrofísica, osoares@lna.br*

[2] *Instituto Tecnológico de Aeronáutica, ramones@ita.br*

Resumo: Este trabalho apresenta o desenvolvimento de um *script* no ambiente de programação p5.js para montagem e simulação 3D do telescópio Perkin Elmer 1,60m. O projeto integra dados em tempo real provenientes de um arquivo `JSON` para realizar cálculos de coordenadas equatoriais, permitindo o posicionamento preciso do telescópio em ambiente virtual. O *script* organiza as partes do telescópio, aplicando transformações de posição e rotação em um *loop* contínuo podendo ser visualizado diretamente em um navegador *web*. Além disso, o estudo explora perspectivas futuras, como a criação de simuladores baseados em *APIs* e destaca o potencial para aplicação em outros contextos astronômicos.

Abstract: *This work presents the development of a script in the p5.js programming environment for 3D assemby and simulation of the Perkin Elmer 1.60m telescope. The project integrates real-time data from a JSON file to perform calculations of equatorial coordinates, enabling precise positioning of the telescope in a virtual environment. The script organizes the telescope's parts, applying transformations in position and rotation in a continuous loop, which can be visualized directly in a web browser. Furthermore, the study explores future prospects, such as the creation of API-based simulators, and highlights the potential for application in other astronomical contexts.*

Palavras-chave/*Keywords*: Telescópio Perkin Elmer 1.60m, p5.js, montagem 3D, simulação / *Perkin Elmer 1.60m telescope, p5.js, 3D assembly, simulation.*

Submetido em: Jan/2025

Revisado por:

Eder Martioli
Márcio Vital de Arruda
Saulo Roberly Gargaglioni
Vanessa Bawden de P. Macanhan de Arruda

Sumário

Lista de Figuras

1 Introdução

O desenvolvimento de modelos 3D em softwares de *design* é uma tecnologia bastante empregada no desenvolvimento de projetos e simulação de funcionamento de mecanismos. O uso de bibliotecas digitais como o `p5.js` para montagem de modelos 3D possibilita a visualização, interatividade e exploração do movimento destes modelos em ambiente virtual diretamente em um navegador *web*.

O presente trabalho tem por objetivo utilizar o `p5.js` para a montagem de um modelo 3D detalhado e interativo do telescópio Perkin-Elmer 1,60m, incluindo a simulação de seus movimentos com base em dados compilados em um arquivo JSON.

Ao longo deste documento, serão detalhadas as técnicas empregadas na montagem do modelo 3D e na incorporação dos dados de movimento, destacando os desafios enfrentados e as soluções desenvolvidas para capturar com fidelidade as características e o comportamento do telescópio. Cada seção do estudo apresenta os passos necessários para o desenvolvimento do projeto.

A relevância deste trabalho reside no entendimento do funcionamento e operação do telescópio a partir de suas coordenadas observacionais, oferecendo uma ferramenta interativa que representa a estrutura física do telescópio e também simula seu funcionamento, sendo de grande utilidade para pesquisadores, estudantes e entusiastas em astronomia. Além disso, este trabalho oferece uma solução escalável e replicável, com implicações para outros projetos relacionados à astronomia e ao estudo de mecanismos.

2 Sobre o p5.js

O p5.js é uma biblioteca de JavaScript criada para tornar o código acessível e inclusivo para artistas, *designers*, educadores e iniciantes. É uma biblioteca de código aberto que torna a criação gráfica e de som no navegador fácil e intuitiva. Com p5.js, é possível desenhar usando funções que são simples de entender e usar [1]. A biblioteca é baseada no *Processing*, um ambiente de programação visual desenvolvido para ensinar os fundamentos da programação em um contexto visual.

Nas seções seguintes, serão apresentadas as formas de carregar e exibir modelos 3D no formato .obj usando *p5.js* com WebGL, além de aplicar transformações de posição, rotação baseadas em informações extraídas de um arquivo JSON. O projeto deve ser executado em um servidor local para que a imagem seja renderizada em um navegador *web*.

2.1 Configurando o *workspace*

Nesta seção serão apresentados os métodos e recursos necessários para configuração do *workspace*, isto é, os requisitos necessários para desenvolvimento do projeto.

2.1.1 Estrutura de arquivos

O diretório do projeto pode seguir a seguinte estrutura de arquivos:

```
/p5js-telescope-project
  /assets
    base.obj
    eixo.obj
    tubo.obj
    config.json
    inconsolata.otf
  index.html
  sketch.js
```

2.1.2 Criação do arquivo index.html

O arquivo index.html segue a seguinte estrutura:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <link rel="stylesheet" type="text/css" href="style.css">
  <script src="p5.js"></script>
  <meta charset="utf-8" />
</head>

<body>
  <main>
  </main>
  <script src="sketch.js"></script>
</body>

</html>
```

Script 1: Estrutura básica do HTML para a página web.

2.1.3 Criação do arquivo JSON (`config.json`)

O arquivo `config.json` segue, inicialmente, a seguinte estrutura:

```
{
    "tag": "PE160",
    "date": "2024-02-16",
    "hour": "09:38:32",
    "ah": "-00 00 00.06",
    "dec": "-22 32 04.5"
}
```

Script 2: Arquivo JSON.

2.1.4 Criação do arquivo em JavaScript (`sketch.js`)

No arquivo `sketch.js`, realiza-se o pré-carregamento dos arquivos presentes no diretório `/assets`. Sendo estes, os modelos 3D no formato `.obj`, o arquivo `config.json` e o arquivo referente à fonte utilizada para o letreiro (`inconsolata.otf`):

```
// Declaracao de variaveis globais
let base
let eixo
let tubo
let inconsolata
let data = {}
let date, hour, ah, dec

async function getJSONData() {
  data = await loadJSON('assets/config.json')
}

// Funcao de pre-carregamento dos 'assets'
function preload() {
  base = loadModel("assets/base.obj", false)
  eixo = loadModel("assets/eixo.obj", false)
  tubo = loadModel("assets/tubo.obj", false)
  inconsolata = loadFont('assets/inconsolata.otf')
  data = loadJSON('assets/config.json')
  setInterval(getJSONData, 1000) // Sincroniza a cada 1000 ms
}

// Funcao de configuracao geral do script
function setup() {
  createCanvas(windowWidth, windowHeight, WEBGL)
  textFont(inconsolata)
  textSize(height / 35)
  textAlign(CENTER, CENTER)
}

// Funcao ciclica do ambiente de desenho
function draw() {
    // O codigo principal estara aqui!
}
```

Script 3: Estrutura básica do *script* principal para teste.

O objetivo é aplicar as transformações dentro da função `draw()` de acordo com as informações do arquivo `config.json`. Este processo será melhor explicado na Seção 3.

2.2 Configuração de um servidor local (*localhost*)

Para configurar e executar um servidor local existem algumas soluções: usar o Python ou Node.js. Outra alternativa bastante recomendada é o uso do VSCode.

A seguir, serão apresentados alguns métodos para que a página web seja acessada no endereço http://127.0.0.1:5500 (que corresponde ao acesso à um servidor local na porta 5500).

2.2.1 Usando o Python

Com o Python 3 instalado, basta abrir o terminal no diretório do projeto e executar:

```
$ python -m http.server 5500
```

2.2.2 Usando o Node.js

Com o Node.js instalado, basta instalar o pacote http-server :

```
$ npm install -g http-server
```

E depois iniciar o servidor no diretório do projeto e executar:

```
$ http-server -p 5500
```

2.2.3 Usando o *VSCode*

O Visual Studio Code, ou simplesmente *VSCode*, é um editor de código-fonte desenvolvido pela Microsoft para funcionar em sistemas operacionais Windows, Linux e macOS [2]. As principais vantagens de usar o p5.js com VSCode são: edição, gestão e execução do servidor através de um único ambiente.

Com o *VSCode* instalado no computador, é necessário instalar duas extensões: a primeira chamada "*p5.vscode*"e a segunda chamada "*Live Server*"[2]. Para ativar o *Live Server*, basta clicar com o botão direito do mouse sobre o arquivo `index.html` e selecionar a opção `Abrir com Live Server`, como ilustra a Figura 1 abaixo.

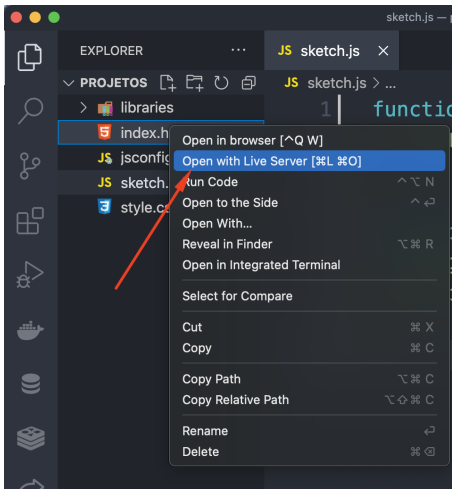

Figura 1: Ativação do servidor local via *VSCode*.
Fonte: [2]

3 Desenvolvimento do *software*

Nesta seção serão apresentadas as técnicas e estratégias utilizadas para o desenvolvimento do *script* principal da aplicação *web*.

3.1 Funcionamento do *script*

A Figura 2 a seguir apresenta o processo de funcionamento do *script* no ambiente p5.js com a inserção de modelos 3D no formato `.obj`, que representam as partes do telescópio. Há também, a leitura contínua de um arquivo de dados `JSON` que servirá de base para efetuar os cálculos de conversão de unidades correspondentes as coordenadas de apontamento do telescópio.

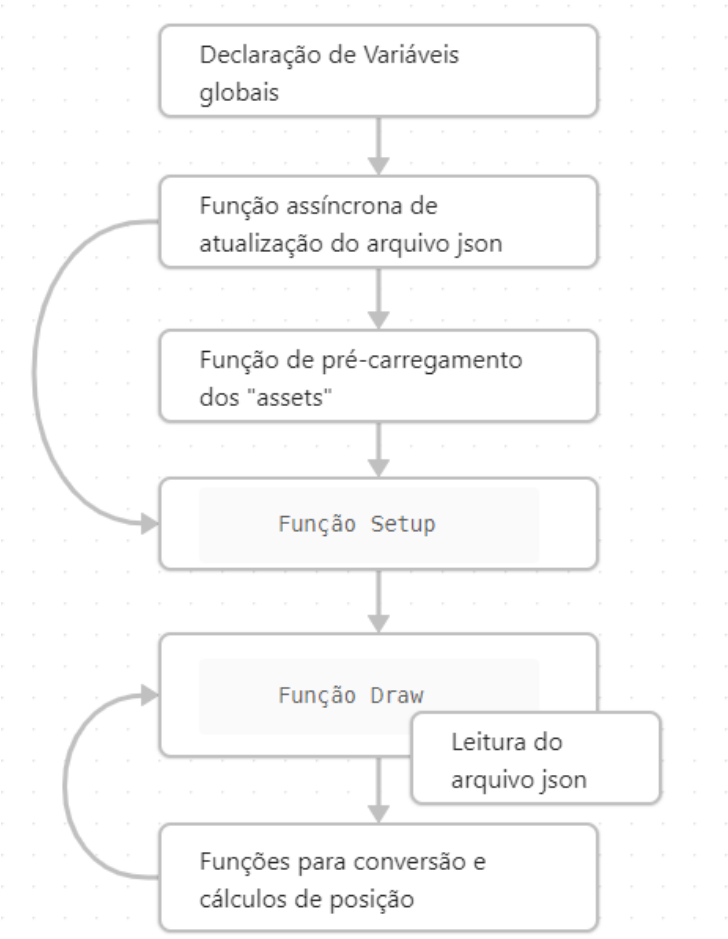

Figura 2: Visão geral de funcionamento do *script*.

A função `draw()` é a função principal do *script* uma vez que é responsável por posicionar e gerenciar o modelo 3D continuamente em um *loop* infinito.

3.2 Conteúdo da função `draw()`

A Figura 3 abaixo apresenta a sequência de ações para montar o telescópio na função `draw()` do ambiente de programação p5.js:

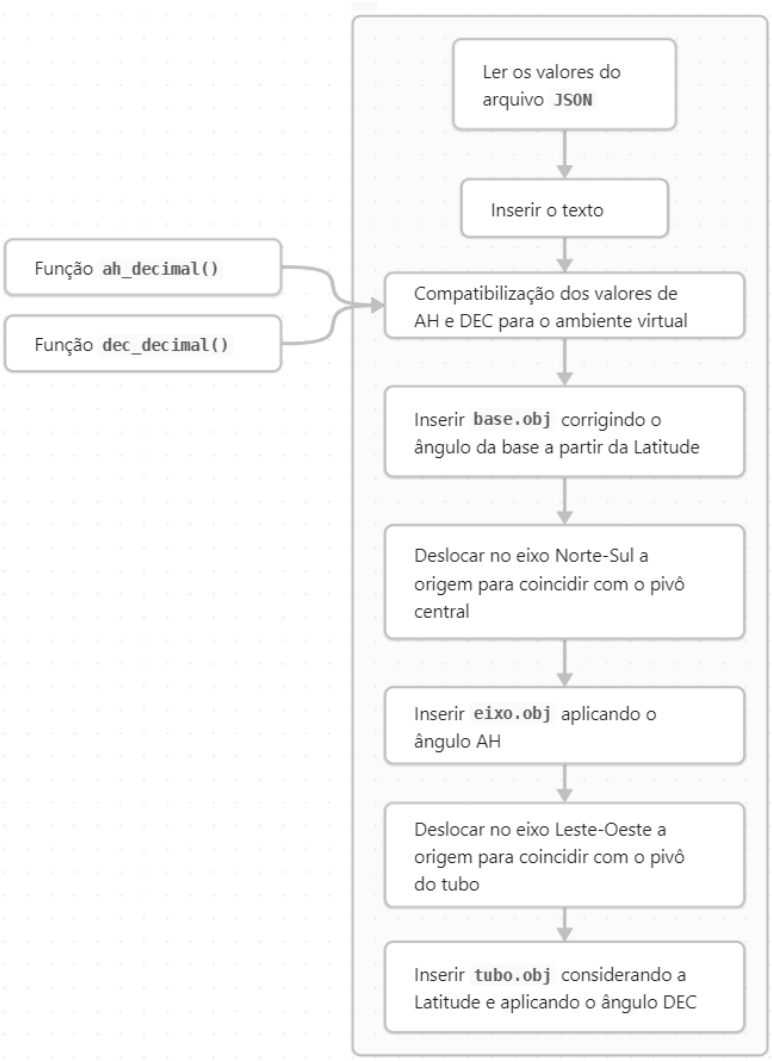

Figura 3: Diagrama de funcionamento da função `draw()`.

As funções que convertem AH e DEC para valores em decimal foram colocadas externamente pois envolvem o tratamento de *strings* e cálculos matemáticos que deixariam o *script* principal muito carregado de informações caso fossem resolvidos dentro da função `draw()`. Além disso, estas funções podem ser substituídas por recursos externos ao código, tais como uma *API* que já entrega os valores em decimal.

3.2.1 Processo de montagem das peças

As peças utilizadas neste estudo foram previamente modeladas em um *software* de modelagem 3D e exportadas para o formato `.stl`. Em seguida, os modelos foram convertidos para o formato `.obj`. Esta última etapa foi realizada por meio do *software* *meshlab* que foi fundamental na conversão dos formatos de arquivos e principalmente no posicionamento e ajustes necessários na origem do sistema de coordenadas dos modelos tridimensionais.

A técnica de montagem necessita do posicionamento do conjunto levando-se em consideração os pivôs de cada peça, ou seja, os pontos de rotação do telescópio. A posição de cada pivô foi colocada estrategicamente na origem do sistema de coordenadas *xyz* de cada peça.

No ambiente do `p5.js`, as peças foram inseridas e posicionadas tomando-se como referência as respectivas origens. A Figura 4 apresenta o processo de montagem das peças que formam o conjunto do telescópio.

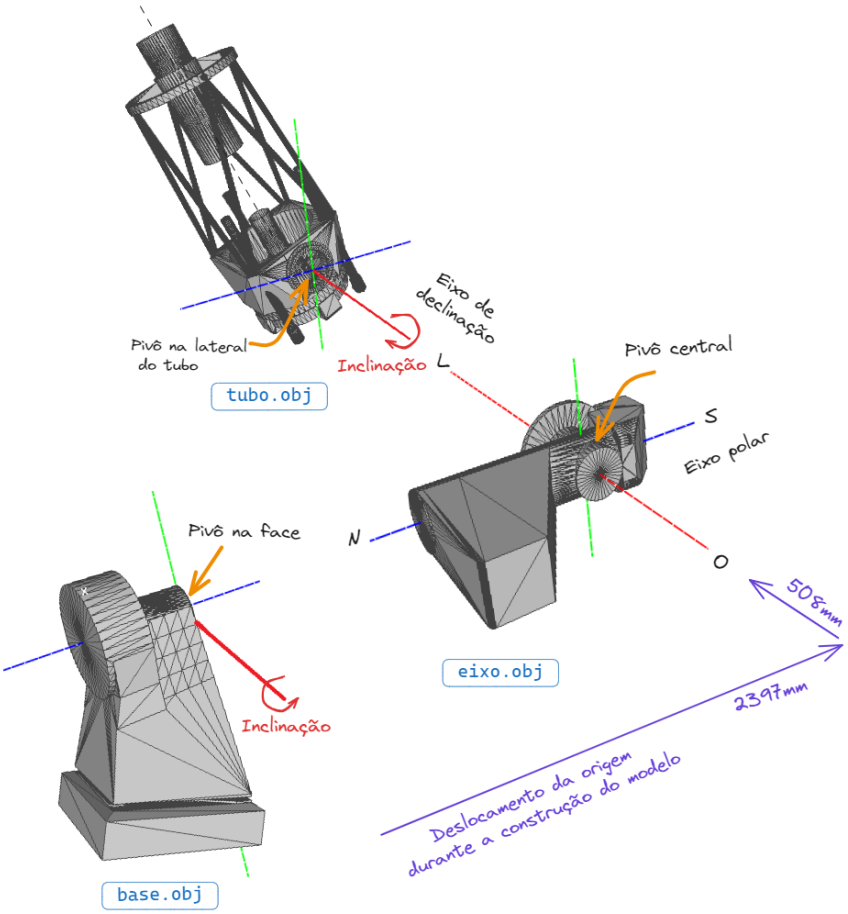

Figura 4: Processo de posicionamento dos pivôs das peças dentro do ambiente p5.js.

No modelo `base.obj` o pivô está localizado na face da peça por onde passa o eixo polar (eixo Norte-Sul). Esta peça possui uma inclinação para frente que corresponde ao valor da coordenada de latitude onde o telescópio foi instalado. Por conseguinte, a inclinação é corrigida para que o conjunto fique nivelado.

No modelo `eixo.obj` o pivô está localizado no ponto de encontro do eixo Norte-Sul, Leste-Oeste do telescópio. Esta peça é inserida e deslocando-se a origem ao longo do eixo norte-sul com o valor correspondente à distância entre o pivô da base e o pivô central.

No modelo `tubo.obj` o pivô está localizado na lateral. O modelo foi orientado originalmente com inclinação voltada para trás, cujo valor corresponde à latitude onde o telescópio foi instalado. Esta peça é inserida e desloca-se a origem ao longo do eixo Leste-Oeste com o valor correspondente à distância entre a face lateral do tubo e o pivô central. Corrige-se também a inclinação rotacionando-se o tubo para frente com o valor da latitude.

3.2.2 Resultado final da montagem

Finalmente, seguindo os passos anteriores, o modelo estará organizado de modo que o conjunto do telescópio esteja na posição zênite. Como apresenta a Figura 5 a seguir:

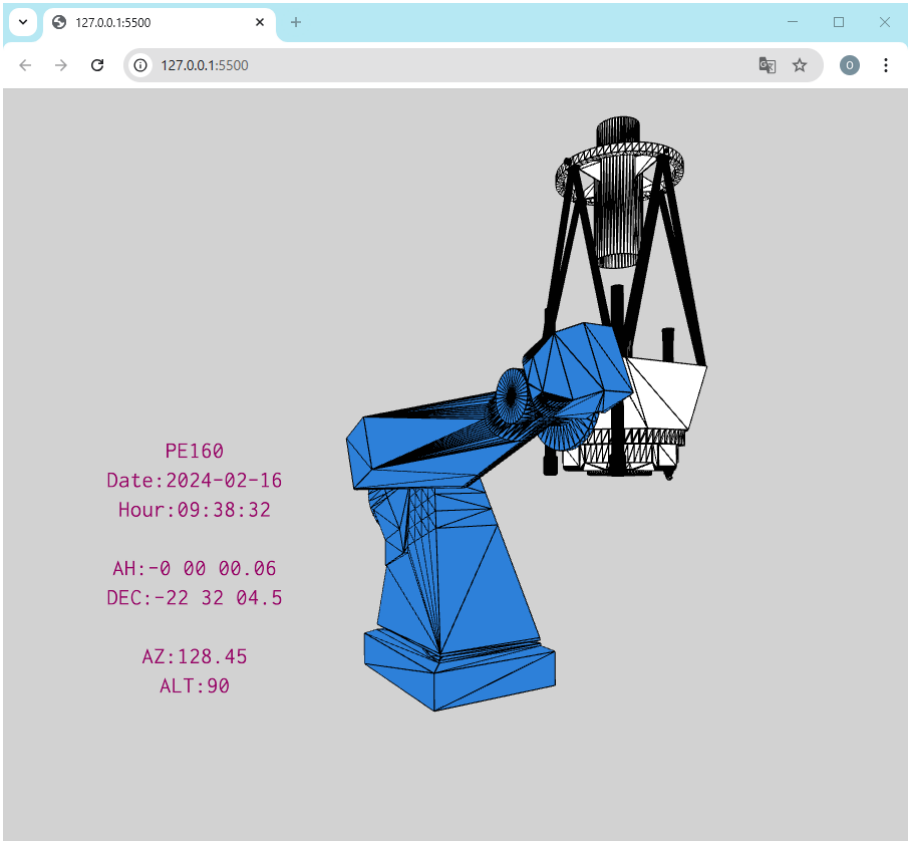

Figura 5: Modelo 3D do telescópio visualizado no navegador *web*.

A Figura 5 apresenta o resultado final de montagem do conjunto no ambiente do p5.js e com visualização disponível em um navegador *web*. Observe que o modelo segue as informações iniciais apresentadas no arquivo `config.json`. Deste modo, basta mudar os valores das coordenadas AH e DEC neste arquivo que o modelo 3D irá mudar de posição instantaneamente.

3.3 *Script* completo (`sketch.js`)

```
let base
let eixo
let tubo
let inconsolata
let data = {}
let date, hour, ah, dec

async function getJSONData() {
  data = await loadJSON('assets/config.json')
}

function preload() { // Funcao de pre-carregamento dos 'assets'
  base = loadModel("assets/base.obj", false)
  eixo = loadModel("assets/eixo.obj", false)
  tubo = loadModel("assets/tubo.obj", false)
  inconsolata = loadFont('assets/inconsolata.otf')
  data = loadJSON('assets/config.json')
  setInterval(getJSONData, 1000)
}

function setup() {
  createCanvas(windowWidth, windowHeight, WEBGL)
  textFont(inconsolata)
  textSize(height / 45)
  textAlign(CENTER, CENTER)
}

function draw() {
  background(210)

  if (data['hour']) {  // Evita o 'UNDEFINED' como resultado
    hour = data['hour']
    ah = data['ah']
    dec = data['dec']
    date = data['date']
  }

  latitude = -22.5344 // latitude do telescopio PE160
  AzElev = getAzimuthElevation(ah_decimal(), dec_decimal(), latitude)

  fill(150, 0, 100) // Atribui cor ao letreiro
  text('\nPE160\n' + 'Date:' + date + '\n' + 'Hour:' + hour + '\n\n' + '
   AH:' + ah + '\n' + 'DEC:' + dec + '\n\n'+"AZ:"+ AzElev.azimuth+ '\n'+
   "ALT:"+ AzElev.elevation+ '\n' , -260, 100)
  noFill()

  scale(.06)
  fill(45, 128, 217) // Atribui cor ao modelo
  stroke(1)
  rotateY(-150 * PI / 180) // Gira o conjunto

  rotateX((latitude-180) * PI / 180) // Correcao do pivo
  model(base)

  rotateZ(-ah_decimal()*15 * PI / 180) // Configura o valor do Eixo RA
  translate(0, 0, 2397) //Coincidir os pivos da base e do eixo
  model(eixo)
```

```
  translate(-508, 0, 0) //Coincidir os pivos do eixo e do tubo
  rotateX(latitude * PI / 180)  // Correcao do pivo
  rotateX(-1 * (dec_decimal()) * PI / 180)  // Configura o valor do DEC
  fill(255, 255, 255)// Atribui cor ao modelo
  model(tubo)
}

function ah_decimal() {
  ra_result = float(abs(ah.split(" ")[0])) + float(ah.split(" ")[1] /
   60) + float(ah.split(" ")[2] / 3600)

  if (ah.split(" ")[0] + 0.5 < 0) { // Detecta valor negativo
    ra_result = -ra_result
  }
  return ra_result
}

function dec_decimal() {
  dec_result = float(abs(dec.split(" ")[0])) + float(dec.split(" ")[1] /
    60) + float(dec.split(" ")[2] / 3600)

  if (dec.split(" ")[0] < 0) {
    dec_result = -dec_result
  }
  return dec_result
}

function windowResized() {
  resizeCanvas(windowWidth, windowHeight);
}

function getAzimuthElevation(ha, dec, latitude) { // Conversao AZ e ALT
  const DEG2RAD = Math.PI / 180;
  const RAD2DEG = 180 / Math.PI;
  const H = ha*15; //Converte ah_decimal em graus

  // Calculo da elevacao
  const sinElevation = (Math.sin(dec * DEG2RAD) * Math.sin(latitude *
   DEG2RAD)) +
                       (Math.cos(dec * DEG2RAD) * Math.cos(latitude *
   DEG2RAD) * Math.cos(H * DEG2RAD));
  let elevation = Math.asin(sinElevation) * RAD2DEG; // Altura em graus
  elevation = Math.round(elevation * 100) / 100; // Arredondamento

  // Calculo do azimute
  const cotgAzimute = Math.sin(latitude * DEG2RAD)*(1/Math.tan(H *
   DEG2RAD)) -  (( Math.cos(latitude * DEG2RAD) * Math.tan(dec * DEG2RAD
   )) / Math.sin(H * DEG2RAD))
  let azimuth = Math.atan2(1, cotgAzimute) * RAD2DEG;

  if (H < 0){ // Arredondamento
    azimuth =  Math.round((azimuth)*100)/100
  }
  else{ // Correcao do lado e Arredondamento
    azimuth = Math.round((azimuth+180)*100) / 100
  }

  return {azimuth, elevation};
}
```

3.3.1 Descrição das funções presentes no *script*

1. `getJSONData()`: Função definida como assícrona (`async function`) para leitura do arquivo JSON em intervalo de tempo pré-determinado `setInterval(getJSONData, 1000)`;

2. `preload()`: Função que realiza o pré-carregamento dos arquivos que são utilizados no código;

3. `setup()`: Função responsável pela configuração global do `script` definindo o modo de funcionamento;

4. `draw()`: Função responsável por desenhar o que será apresentado na tela. Aplica as transformações de posição, rotação e escala de acordo com o processamento das informações de entrada;

5. `ah_decimal(), dec_decimal()`: Funções responsáveis por receber uma *string* de posição, processar e retornar o valor correspondente em decimal.

6. `windowResized()`: Ajusta o tamanho da tela quando a janela é redimensionada. É uma maneira de tornar a tela responsiva.

7. `getAzimuthElevation(ha, dec, latitude)`: Calcula os valores em azimute e altura à partir de ângulo horário, declinação e latitude.

3.3.2 Projeto disponível no *GitHub*

O presente projeto está disponível no Github no repositório: https://github.com/osoares-git/p5js-telescope-project. Deste modo, caso o usuário já possua o `git` instalado e configurado na máquina, o projeto pode ser clonado através dos seguintes passos:

1. **Clone do repositório do GitHub:** É necessário abrir um terminal ou *prompt* de comando e executar o seguinte comando:

```
git clone https://github.com/osoares-git/p5js-telescope-project.git
```

2. **Checagem do diretório do projeto:** É conveniente acessar o diretório onde o projeto foi clonado e conferir o conteúdo:

```
$ cd ./p5js-telescope-project/

$ ls

README.md  assets/  index.html  p5.js  p5.sound.min.js  sketch.js
   style.css
```

3. **Configuração do projeto:** Visto que todos os arquivos necessários estão no lugar o passo seguinte é configurar o servidor para que o projeto seja executado diretamente em um navegador *web*.

4. **Interação com a simulação:** O usuário poderá visualizar a montagem do telescópio diretamente no navegador e, ao alterar/salvar os valores das coordenadas no arquivo `json`, o modelo 3D irá reproduzir automaticamente a posição do telescópio.

4 Estudo de movimento do telescópio

A montagem do telescópio Perkin Elmer 1,60m é do tipo equatorial e os parâmetros básicos para posicionamento do telescópio são as coordenadas em ângulo horário (AH) e Declinação (DEC). O posicionamento do telescópio é feito pelo TCSPD (*Telescope Control System Pico dos Dias*) que gera um *Log* de registro do que aconteceu durante a operação do telescópio.

O estudo de movimento pode ser conduzido apenas com a inserção das coordenadas de AH e DEC dos *Logs* de registro no arquivo `config.json`, ou qualquer outro sistema que forneça essas coordenadas.

4.1 Reprodução das posições críticas do telescópio

No desenvolvimento dos trabalhos no Observatório do Pico dos Dias, sempre foi de interesse, da equipe o conhecimento dos limites do telescópio. Ou seja, as posições de *"Security zone"* do TCSPD que fazem com que o telescópio pare imediatamente o movimento e sinalize inclinação excessiva.

Em 2022, foi conduzida uma operação cuidadosa de testes, por meio da movimentação de um eixo por vez, visando saber os limites do telescópio nas direções Norte, Sul, Leste e Oeste. E registros de *Logs* de operação de 2023 e 2024 também sinalizam os limites do telescópio pelo TCSPD. Os valores encontrados estão representados na Figura 6 a seguir:

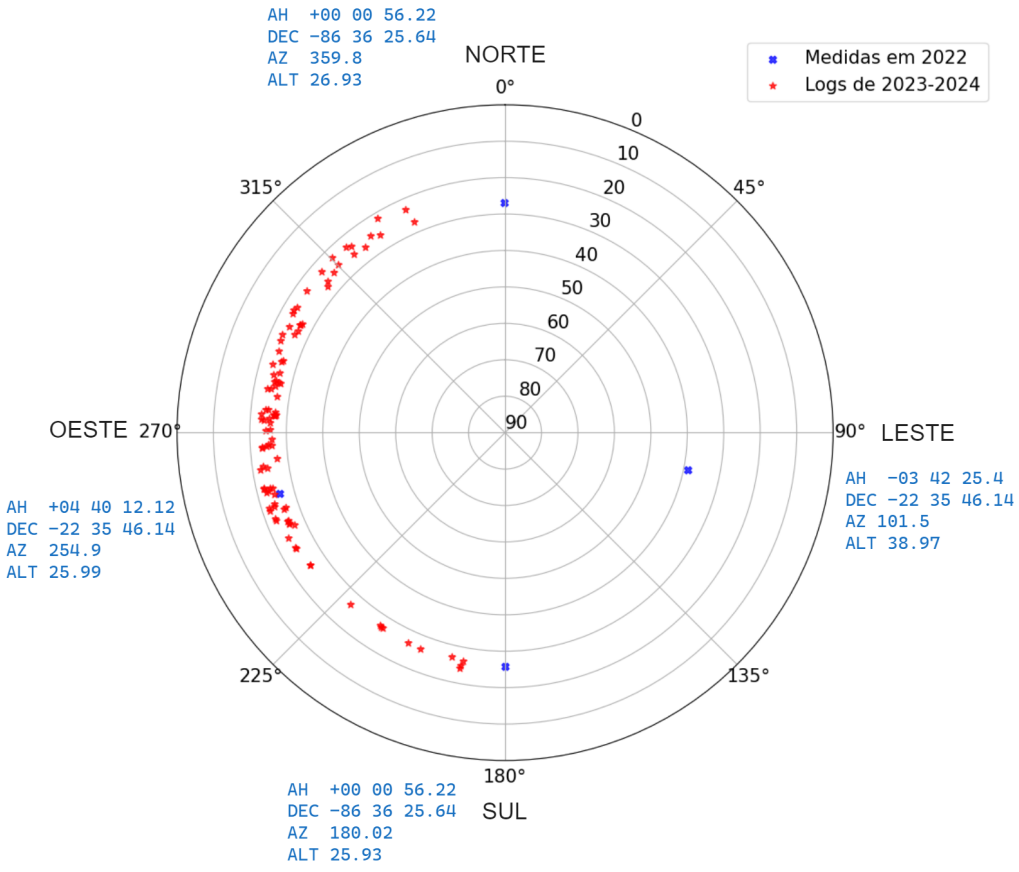

Figura 6: Radar com as posições criticas do telescópio.

O radar apresentado na Figura 6 foi produzido com base nos dados coletados durante os testes realizados em 2022 (em azul) e complementados por *Logs* de observações astronômicas realizadas em 2023 e 2024 (em vermelho). Estes últimos também são pontos de notificação de inclinação excessiva alertados pelo TCSPD que, por medida padrão de segurança, efetuou a parada imediata do telescópio nestas condições.

É importante ressaltar que os valores de azimute e altura (na Figura 6) não foram fornecidos pelo TCSPD. Os mesmos foram obtidos através de cálculos de conversão de coordenadas horárias para coordenadas horizontais através das expressões matemáticas descritas no Apêndice deste trabalho.

Uma vez que é muito arriscado e pouco conveniente repetir essas posições com o telescópio real. O estudo de movimento através do modelo 3D se torna um recurso interessante para reproduzir a posição do telescópio em ambiente virtual. A Figura 7 a seguir apresenta as posições críticas do telescópio tomando por base os valores de inclinação excessiva nas direções Norte, Sul, Leste e Oeste (medidas de 2022) apenas com o objetivo de apresentar o potencial oferecido pelo uso deste novo recurso.

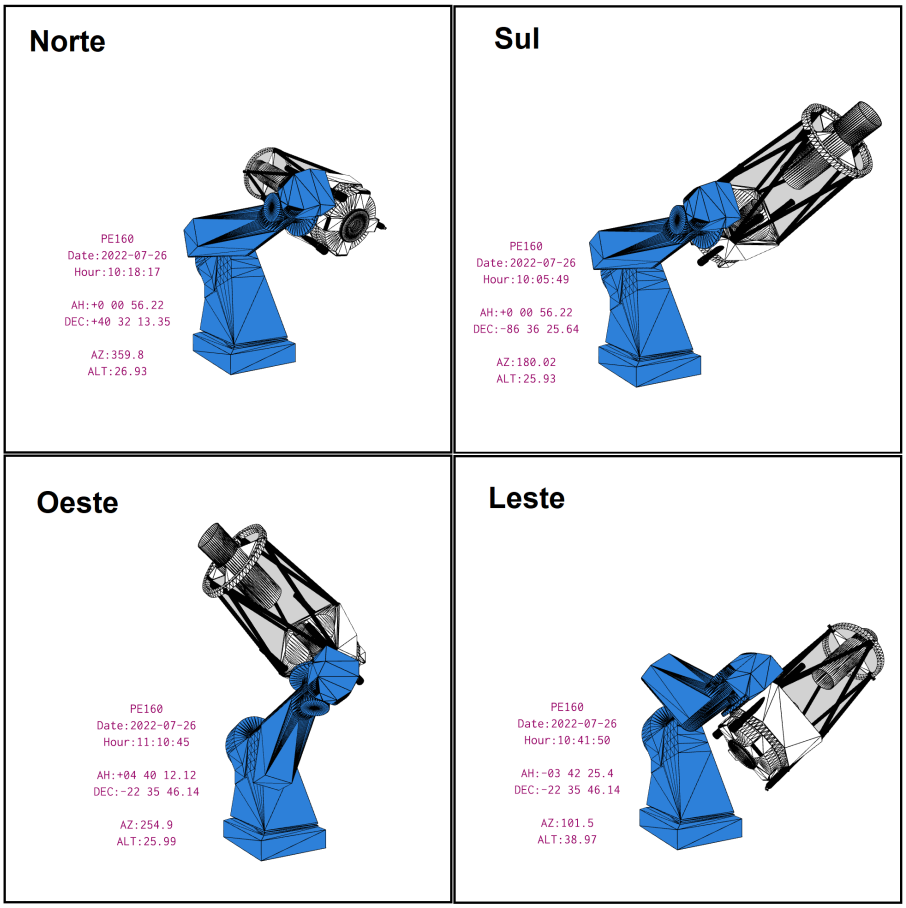

Figura 7: Reprodução das posições críticas do telescópio 1,60m.

Note que tanto o radar apresentado na Figura 6 quanto a posição indicada na Figura 7 são recursos que, se colacados em um *dashboard*, ajudariam no monitoramento da operação do telescópio.

4.2 *Logs* de apontamentos e posição atual do telescópio

A operação do TCSPD gera, a cada segundo, informações de apontamentos que são armazenadas em arquivos de *Log* no formato `.txt`. Esses arquivos podem ser processados, mesmo durante as observações astronômicas, no objetivo de obter e atualizar as informações no arquivo `config.json`. As últimas modificações do TSCPD utilizam o sistema de comunicação *ZeroMQ* com informações de *status* instantâneo de operação do telescópio, sendo assim, outra possível fonte de alimentação das informações do arquivo `config.json`.

O desafio aqui é o processamento dos *Logs* de apontamento em tempo real do telescópio e atualização das informações. Deste modo, o modelo 3D funciona reproduzindo as informações atuais do telescópio, porém com um ligeiro atraso (*delay*) inerente ao próprio processamento e atualização das informações. A primeira versão de teste já pode ser acessada na página: https://coopd.lna.br:8090/pe160. No entanto, os *scripts* em *python* de sincronização do arquivo `config.json` ainda não foram impelentados até a data de submissão do presente trabalho.

Durante as observações astronômicas não é permitido haver iluminação no ambiente onde o telescópio opera. Deste modo, a visualização da estrutura por meio de uma câmera de vídeo é complicada e, por vezes, inviável. A grande vantagem da presente aplicação é a visualização, mesmo com atraso, da posição atual do telescópio.

4.3 Simulador de apontamentos

Com o objetivo de aumentar a segurança operacional dos telescópios do OPD, os autores do presente trabalho desenvolveram, em 2023, um simulador de apontamentos para o telescópio Perkin-Elmer 1,60m usando parte do *script* apresentado na Seção 3.3.

O simulador combina as técnicas empregadas neste trabalho com *scripts* em *python*, e a integração com *APIs* para permitir a visualização, em tempo real, das posições do telescópio geradas pela *interface* de controle.

Esse simulador já está em uso em situações práticas, como consultas sobre a viabilidade e segurança das posições do telescópio para eventos específicos. Mais informações sobre este projeto podem ser obtidas na mátéria publicada na revista LNA em dia nº 63.

4.4 Perspectivas futuras

As técnicas empregadas nesse estudo podem ser reproduzidas em outros telescópios e mecanismos (plataforma, cúpula e trapeira), desde que sejam desenhados e referenciados os respectivos modelos tridimensionais e que haja as informações de posição desses mecanismos.

A plataforma, por exemplo, é um dos fatores críticos de risco, pois possui seu próprio movimento e é controlada manuamente por um sistema eletro-hidráulico. Uma primeira versão de testes com o modelo da plataforma já foi construída e está nas primeiras etapas de desenvolvimento. Os primeiros efeitos podem ser constatados na Figura 8.

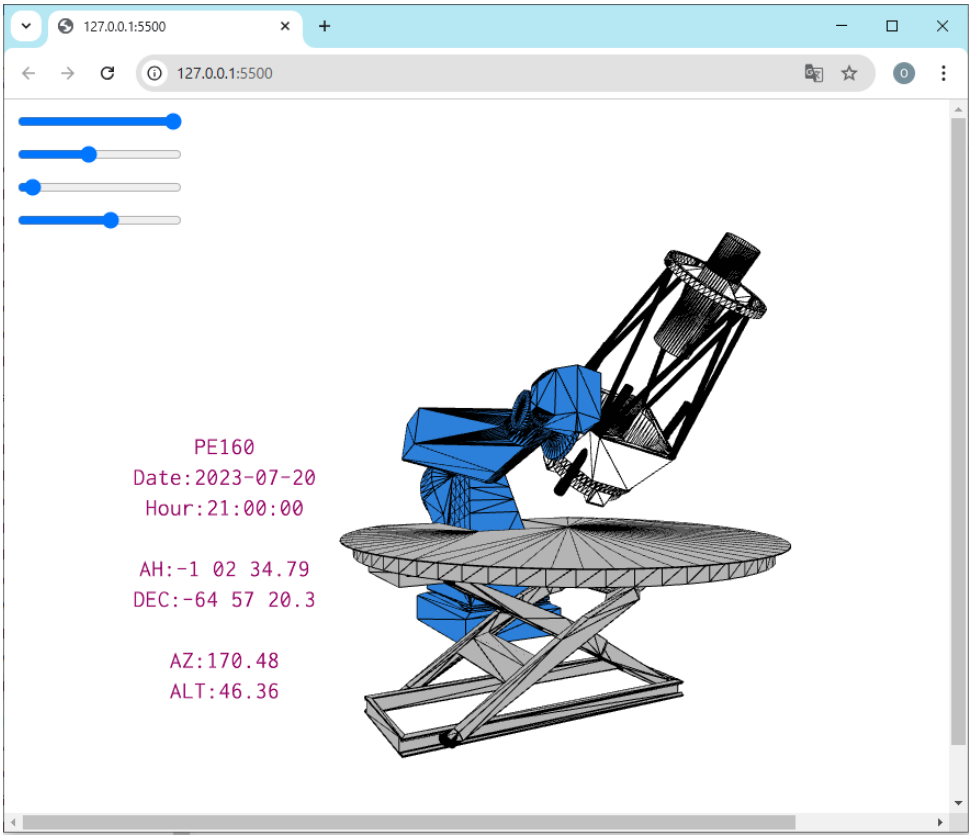

Figura 8: Versão inicial de um projeto do telescópio com a plataforma.

Atualmente, existem sensores ultrassônicos com o microcontrolador arduino e antena ESP8266 instalados embaixo da plataforma, que fornecem uma *string* com as informações de posição da plataforma via rede sem fio. A integração desses dados ao modelo 3D permitirá simulações mais precisas e a definição de zonas de risco dinâmicas, ajudando a evitar situações em que a movimentação do telescópio ou da plataforma possam causar danos irreversíveis aos mecanismos e adjacências do sistema.

Neste sentido, um aspecto essencial a ser incorporado ao modelo é a presença de instrumentos acoplados ao telescópio. Um caso relevante é o SPARC4 (*Simultaneous Polarimeter and Rapid Camera in 4 bands*), um equipamento astronômico avançado desenvolvido em uma parceria entre o Instituto Nacional de Pesquisas Espaciais - INPE e o Laboratório Nacional de Astrofísica - LNA que, em operação, é instalado diretamente na base do tubo do telescópio. Uma operação mal sucedida, poderia ser facilmente danificado pelo movimento combinado do telescópio e da plataforma. A integração do modelo 3D deste e de outros instrumentos permitiria prever com maior precisão possíveis interferências físicas e aprimorar a segurança operacional durante as observações astronômicas, principalmente no contexto de observações remotas onde há a necessidade de clareza para o observador remoto da situação em que o equipamento se encontra.

Por fim, outro objetivo futuro é estudar as características dos arquivos no formato `.obj` para aplicar, dentro do ambiente do `p5js`, efeitos de contato entre as peças. Isso possibilitaria um maior controle de cada peça inserida no ambiente virtual. Sobretudo em cenários onde o telescópio corre o risco de atingir algum obstáculo. Com esta característica, o sistema notificaria um situação de risco apenas pelo contato com uma zona de risco pré-determinada.

5 Conclusão

O trabalho realizado demonstra como a integração entre modelos 3D, programação e informações de posição podem resultar em ferramentas interessantes para reprodução e simulação dos movimentos de telescópios. A modularidade do *script* permite o estudo para diferentes cenários e aplicações em outros tipos de mecanismos.

As técnicas desenvolvidas podem ser ampliadas para estudos mais complexos, envolvendo arquiteturas de rede e monitoramento remoto de ambientes. Esse estudo reforça a importância do uso de tecnologias abertas e acessíveis, como o `p5.js` e o *MeshLab*, para promover a inovação e integração tecnológica em diferentes áreas do conhecimento.

O resultado final deste trabalho permite uma interação mais dinâmica e realista deste icônico instrumento, que é o telescópio Perkin-Elmer 1,60m, do Laboratório Nacional de Astrofísica, estabelecendo um convite para desenvolvimento de futuras soluções para a infraestrutura do Observatório do Pico dos Dias.

Agradecimentos

Gostaríamos de expressar a mais sincera gratidão a todas as pessoas e instituições que tornaram possível a realização deste trabalho. Sobretudo aos profissionais da equipe de técnicos do Observatório do Pico dos Dias que sempre se dispuseram à explicar os detalhes sobre as operações do telescópio Perkin-Elmer 1,60m. Essas informações foram muito importantes durante o desenvolvimento da solução apresentada neste trabalho.

Referências

[1] *What is p5.js*. Disponível em: <https://p5js.org/> Acessado em: 02/07/2024.

[2] Alura. *P5 no VScode: Passo a passo*. Disponível em: <https://www.alura.com.br/artigos/p-5-no-vscode?srsltid=AfmBOooicw-fevcpTUe_9tpFhAS5ecQa_28kHfUw87wdztjMIM-RWVQJ> Acessado em: 09/01/2025.

[3] ARANA J. M. *Transformação de coordenadas*. ASTRONOMIA DE POSIÇÃO: NOTAS DE AULA, UNESP. 2000. p.28–30. <https://edisciplinas.usp.br/pluginfile.php/5557610/mod_resource/content/1/NotasDeAulaUNESP.pdf> Acesso em: 14/01/2025.

[4] ZANETTI M.A.Z.; VEIGA L.A.K. *Transformações entre sistemas de coordenadas (por trigonometria esférica)*. GA160 TÓPICOS EM ASTRONOMIA, UFPR. 2018. p.25–26. <https://cartografica.ufpr.br/wp-content/uploads/2018/08/GA160-T%C3%B3picos-em-Astronomia-unidades-1-e-2.pdf> Acesso em: 14/01/2025.

[5] SANTIAGO B. *Astronomia Esférica*. APOSTILA DE ASTRONOMIA GEODÉSICA, UFRGS. 2005. <https://www.if.ufrgs.br/oei/santiago/fis2005/textos/esferast1.htm> Acesso em: 14/01/2025.

Apêndice

Transformação de coordenadas

O problema de transformação de coordenadas pode ser solucionado pela trigonometria esférica ou por matrizes ortogonais por rotação de eixos [3]. Neste trabalho, optou-se por utilizar a trigonometria esférica através da análise do triângulo de posição que nada mais é do que um triângulo esférico.

Medidas do triângulo esférico

Os ângulos do triângulo esférico são medidos pelos diedros do triedro enquanto que os lados do triângulo esférico são medidos pelos ângulos planos das faces do diedro [4]:

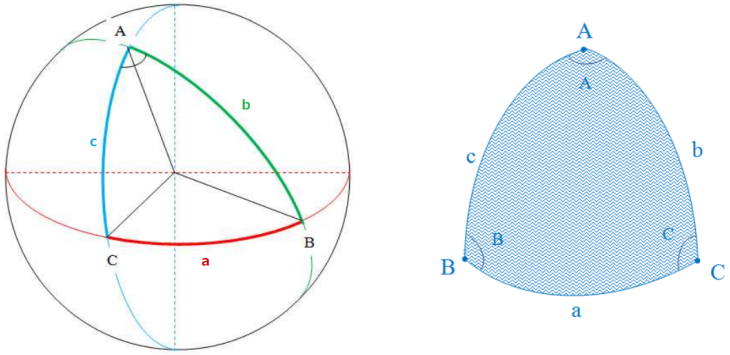

Figura 9: Medidas de ângulos e lados de um triângulo esférico.
Fonte: [4], adaptado.

Equações fundamentais

Existem algumas equações em trigonometria esférica que relacionam todas as medidas apresentadas na Figura 9. O conjunto dessas equações, fundamentais, são apresentadas a seguir:

Fórmula dos 4 elementos relativa a lados ou lei dos cossenos (3 lados e 1 ângulo):

$$\begin{aligned} cosa &= (cosb \times cosc) + (senb \times senc \times cosA) \\ cosb &= (cosa \times cosc) + (sena \times senc \times cosB) \\ cosc &= (cosb \times cosa) + (senb \times sena \times cosC) \end{aligned} \tag{1}$$

Fórmula dos 4 elementos relativa a ângulos ou lei dos cossenos (3 ângulos e 1 lado):

$$\begin{aligned} cosA &= -(cosB \times cosC) + (senB \times senC \times cosa) \\ cosB &= -(cosA \times cosC) + (senA \times senC \times cosb) \\ cosC &= -(cosB \times cosA) + (senB \times senA \times cosc) \end{aligned} \tag{2}$$

Analogia dos senos (2 ângulos e 2 lados opostos):

$$\frac{senA}{sena} = \frac{senB}{senb} = \frac{senC}{senc} \tag{3}$$

Fórmula das cotangentes (2 ângulos e 2 lados não opostos):

$$\begin{aligned}
sena \times cotgc &= (senB \times cotgC) + (cosa \times cosB) \\
sena \times cotgb &= (senC \times cotgB) + (cosa \times cosC) \\
senb \times cotga &= (senC \times cotgA) + (cosb \times cosC) \\
senb \times cotgc &= (senA \times cotgC) + (cosb \times cosA) \\
senc \times cotga &= (senB \times cotgA) + (cosc \times cosB) \\
senc \times cotgb &= (senA \times cotgB) + (cosc \times cosA)
\end{aligned} \tag{4}$$

Fórmula dos cinco elementos (3 lados e 2 ângulos):

$$\begin{aligned}
senb \times cosC &= (sena \times cosc) - (cosa \times senc \times cosB) \\
senb \times cosA &= (senc \times cosa) - (cosc \times sena \times cosB) \\
sena \times cosB &= (senc \times cosb) - (cosc \times senb \times cosA) \\
sena \times cosC &= (senb \times cosc) - (cosb \times senc \times cosA) \\
senc \times cosA &= (senb \times cosa) - (cosb \times sena \times cosC) \\
senc \times cosB &= (sena \times cosb) - (cosa \times senb \times cosC)
\end{aligned} \tag{5}$$

Algumas propriedades trigonométricas aplicáveis:

$$\begin{aligned}
sen(90 - x) &= +cos(x) \\
cos(90 - x) &= +sen(x) \\
cotg(90 - x) &= +tan(x) \\
\\
sen(180 - x) &= +sen(x) \\
cos(180 - x) &= -cos(x) \\
cotg(180 - x) &= -cotg(x)
\end{aligned} \tag{6}$$

Aplicação para o triângulo de posição

Considerando-se a esfera celeste, na qual o objeto esteja referido simultaneamente aos sistemas de coordenadas horizontal e horária. Tem-se a formação de um triângulo esférico denominado triângulo de posição formado pelos vértices: PNC, Z e E que correspondem ao pólo norte celeste, o zênite e ao astro respectivamente. Enquanto que os lados deste triângulo correspondem ao complemento da latitude do observador, à distância zenital e à distância polar respectivamente. O triângulo de posição de uma estrela e as respectivas medidas são apresentadas na Figura 10.

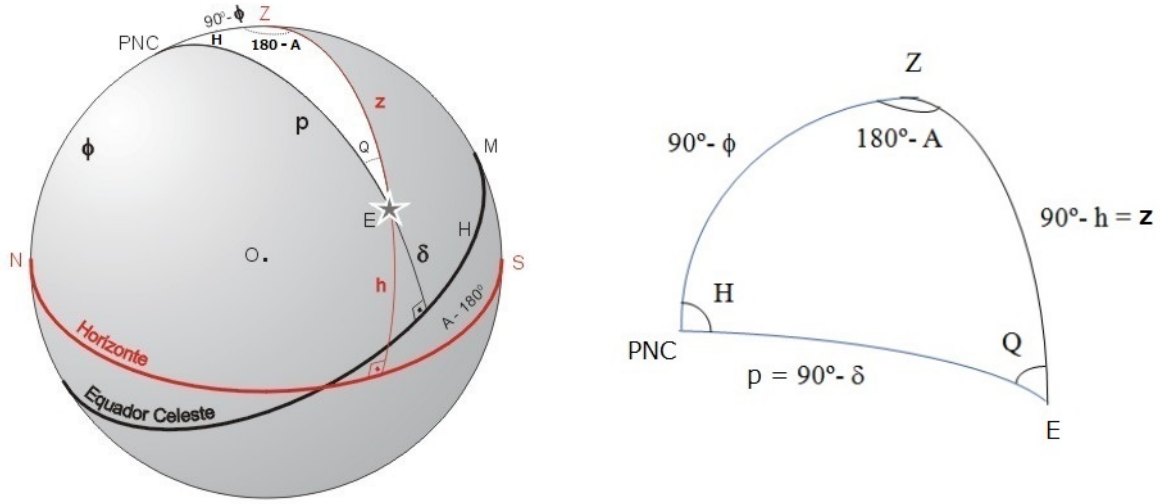

Figura 10: Triângulo de posição de uma estrela.
Fontes: [5] e [4], adaptado.

Onde,

$$
\begin{aligned}
Z &= \text{zênite} \\
z &= \text{distância zenital} \\
h &= \text{altura (ou elevação)} \\
\delta &= \text{declinação} \\
p &= \text{distância polar} \\
\phi &= \text{latitude do observador} \\
H &= \text{ângulo horário} \\
A &= \text{azimute} \\
PNC &= \text{pólo norte celeste}
\end{aligned}
$$

Conversão: Declinação e Ângulo horário e Latitude » Altura e Azimute

Usando a Fórmula dos 4 elementos (eq.1) correspondente calcula-se a **Altura**:

$$\begin{gathered} cos(90-h) = cos(90-\phi)cos(90-\delta) + sen(90-\phi)sen(90-\delta)cos(H) \\ \boxed{cos(z) = sen(h) = sen(\delta)sen(\phi) + cos(\delta)cos(\phi)cos(H)} \end{gathered} \tag{7}$$

Usando a Fórmula das cotangentes (eq.4) correspondente calcula-se o **Azimute**:

$$\begin{gathered} sen(90-\phi)cotg(90-\delta) = sen(H)cotg(180-A) + cos(90-\phi)cos(H) \\ cos(\phi)tan(\delta) = sen(H)[-cotg(A)] + sen(\phi)cos(H) \\ \boxed{cotg(A) = sen(\phi)cotg(H) - \frac{cos(\phi)tan(\delta)}{sen(H)}} \end{gathered} \tag{8}$$

Conversão: Altura, Azimute e Latitude » Declinação e Ângulo horário

Usando a Fórmula dos 4 elementos (eq.1) correspondente calcula-se a **Declinação**:

$$\begin{gathered} cos(90-\delta) = cos(90-\phi)cos(90-h) + sen(90-\phi)sen(90-h)cos(180-A) \\ sen(\delta) = sen(\phi)sen(h) + cos(\phi)cos(h)[-cos(A)] \\ \boxed{sen(\delta) = sen(\phi)cos(z) - cos(\phi)sen(z)cos(A)} \end{gathered} \tag{9}$$

Usando a Fórmula das cotangentes (eq.4) correspondente calcula-se o **Ângulo horário**:

$$\begin{gathered} sen(90-\phi)cotg(z) = sen(180-A)cotg(H) + cos(90-\phi)cos(180-A) \\ cos(\phi)tan(z) = sen(A)cotg(H) + sen(\phi)[-cos(A)] \\ \boxed{cotg(H) = sen(\phi)cotg(A) - \frac{cos(\phi)cotg(z)}{sen(A)}} \end{gathered} \tag{10}$$