UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

ANDRÉ CARRASCO RODRIGUES

Estudo do Sistema de Posicionamento Global (GPS)

São Carlos
2011

ANDRÉ CARRASCO RODRIGUES

Estudo do Sistema de Posicionamento Global (GPS)

Trabalho de Conclusão de Curso
apresentado à Escola de Engenharia de São
Carlos, da Universidade de São Paulo

Curso de Engenharia de Computação com
ênfase em Telecomunicações e
Computação Móvel

ORIENTADOR: Prof. Dr. Amílcar Careli César

São Carlos

2011

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

Rodrigues, André Carrasco.

R696e Estudo do sistema de posicionamento global (GPS)/ André Carrasco Rodrigues ; orientador Amílcar Careli César -- São Carlos, 2011.

Monografia (Graduação em Engenharia de Computação com ênfase em Telecomunicações e Computação Móvel) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2011.

1. GPS. 2. Posicionamento global. 3. Satélites. 4. *OpenGL*. 5. Longitude. 6. Latitude. I. Titulo.

FOLHA DE APROVAÇÃO

Nome: André Carrasco Rodrigues

Título: "Estudo do Sistema de Posicionamento Global (GPS)"

Trabalho de Conclusão de Curso defendido e aprovado em 28 / 11 / 2011,

com NOTA 10 (Dez, zuo), **pela comissão julgadora:**

Marcelo Andrade da Costa Vieira

Prof. Dr. Marcelo Andrade da Costa Vieira - SEL/EESC/USP

Eduardo M. G. de Queiroz

Msc. Eduardo Martinelli Galvão de Queiroz - SEL/EESC/USP

Prof. Associado Evandro Luís Linhari Rodrigues
Coordenador pela EESC/USP do
Curso de Engenharia de Computação

SUMÁRIO

Lista de Figuras

Lista de Tabelas

Resumo

Com o grande avanço tecnológico e simultâneo crescimento da capacidade computacional, o Sistema de Posicionamento Global (GPS) popularizou-se e, atualmente, é utilizado por diversos tipos de equipamentos eletrônicos em todo o mundo. Desenvolvido pelo Departamento de Defesa americano com fins inicialmente militares, o GPS começou a ser utilizado na comunidade civil na década de 90, porém, com precisão limitada. Alguns anos depois, com o aumento da precisão disponível ao uso civil e barateamento dos equipamentos, resultantes do acelerado desenvolvimento da microeletrônica, a utilização do sistema GPS tornou-se uma nova facilidade da sociedade moderna. O texto descreve os diferentes segmentos que compõem o sistema, discute os principais conceitos do seu funcionamento, analisa como é feita a transmissão dos sinais pelos satélites e descreve o formato das palavras binárias, além de alguns dos possíveis métodos matemáticos fundamentais para a determinação da posição do usuário na Terra. Com essas informações, foi desenvolvida uma aplicação em OpenGL que simula o funcionamento do GPS, que descreve em 3 dimensões o movimento orbital dos satélites e calcula a posição de um usuário próximo da superfície terrestre através dos métodos matemáticos descritos no texto. O aplicativo apresenta em um ambiente gráfico dinâmico os movimentos da constelação de satélites GPS com a rotação terrestre, os valores das posições dos satélites e calcula a posição do usuário a cada 100 ms pelo sistema GPS, tanto nas coordenadas cartesianas quanto por longitude, latitude e altitude, confirmando a funcionalidade dos métodos propostos.

Palavras-chave: GPS, posicionamento, global, satélites, OpenGL, longitude, latitude.

Abstract

With the large technological progress and simultaneous growth of computational power, the Global Positioning System (GPS) has become more popular and is currently used by several types of electronic equipments throughout the world. Developed by the U. S. Department of Defense initially for military purposes, the GPS started to be used in the civil community in the 90's despite its limited precision. A few years later, due to the precision increase available to civilian use and cheaper equipments, result from the fast development of microelectronic in recent years, the use of GPS has become a new ease of modern society. The text describes the different segments which compose the system, discusses the key concepts of its operation, analyzes how the signals transmission from the satellites works and demonstrates the format of the binary words applied, further some of the available mathematical methods that are fundamental to determine the user's position on Earth. Given these information, it was developed an OpenGL application that simulates the GPS operation, which describes the orbital movement of satellites in 3D and calculates the position of a user close to the Earth surface through these mathematical methods described in the text. The application presents through a dynamic graphical environment the GPS satellite constellation movements with the Earth's rotation, the values of the satellites and calculates the user positions each 100 ms by GPS, either in Cartesian coordinates or longitude, latitude and altitude, confirming the functionality of the proposed methods.

Key-words: GPS, positioning, global, satellites, OpenGL, longitude, latitude.

1 - Introdução

Os séculos XIV e XV ficaram marcados como a Era das Grandes Navegações e Descobrimentos Marítimos, pela quantidade de viagens marítimas feitas pelos europeus, principalmente portugueses e espanhóis, que resultou no descobrimento de grandes territórios e acumulação de grandes riquezas. Essas navegações tornaram-se possíveis pelo avanço científico da época que resultou, principalmente, na criação de caravelas mais ágeis e utilização da bússola, além de outras ferramentas, que auxiliavam na navegação.

Posteriormente com as diversas mudanças sociais ocorridas ao longo dos séculos, os diferentes interesses e anseios de certos grupos tornaram o incentivo à pesquisa e desenvolvimento tecnológicos uma importante ferramenta para obtenção de vantagens militares sobre seus inimigos. Assim, a necessidade de informações sobre a localização de tropas e controle de outros recursos militares incentivou o desenvolvimento de um sistema que provesse essas informações. Nesse cenário, foram criados sistemas pioneiros na utilização de ondas para descrever localizações, como o Sonar e o Radar [1].

Com a utilização da radiocomunicação desde a 1ª Guerra Mundial, sistemas baseados em ondas de rádio para navegação começaram a aparecer há algumas décadas depois. Os primeiros sistemas utilizavam estações terrestres fixas com antenas que emitiam sinais constantes para as antenas receptoras que, através de características do sinal recebido, determinam suas próprias posições.

Um desses exemplos é o sistema de navegação DECCA, desenvolvido na 2ª Guerra pelos Aliados, que determina a localização da antena receptora analisando a diferença de fase de ondas de frequências baixas, entre 70 e 130 kHz, emitidas a partir das estações fixas. Essas estações consistem de uma estação mestre e três estações secundárias, embora duas estações secundárias sejam suficientes para a determinação do posicionamento. O alcance do sistema depende de vários fatores distintos, entretanto, é tipicamente descrito na ordem de 240 milhas náuticas (440 km) à noite e alcançando o dobro da distancia durante o dia [1], [2].

Outro sistema é o LORAN, inicialmente dividido na versão C para usuários civis e na versão D para usuários militares, mas unificados em um único sistema posteriormente. Foi implantado nas zonas costeiras americana e canadense e opera

de forma semelhante ao DECCA, através de uma estação mestre e três ou quatro estações secundárias. Transmitem ondas na frequência de 100 kHz e a localização é feita medindo o tempo de chegada do sinal vindo da estação mestre e comparando-o com o tempo de chegada das estações secundárias [1]. A utilização do sistema LORAN está completamente descontinuada desde fevereiro de 2010 quando as transmissões do seu sinal foram cessadas [3].

Além desses, o sistema ÔMEGA também consiste de um sistema de navegação via rádio com estações terrestres fixas, entretanto, utiliza-se de oito transmissores que emitem sinais com fases sincronizadas em frequências abaixo de 20 kHz. Devido à baixa frequência, os sinais sofrem diversas reflexões entre a Terra e a ionosfera, tornando o alcance do sistema na escala global, apesar da baixa precisão de posicionamento [1].

Posteriormente, através de experimentos realizados com o primeiro satélite artificial SPUTINIK 1 colocado em órbita do planeta terra, um grupo de trabalho da Universidade John Hopkins desenvolveu um novo método de determinação de posicionamento analisando a ocorrência do *efeito Doppler* na emissão e recepção de sinais por estações terrestres de localização conhecida. Através dele, foi possível determinar a posição aproximada do satélite e, então, perceberam que poderiam usar satélites emissores de ondas eletromagnéticas com posições pré-determinadas para descobrir a localização de um receptor na Terra. Esse sistema foi desenvolvido pela *Applied Physics Laboratory* da Universidade John Hopkins e implantado pela marinha norte-americana, recebendo o nome de sistema TRANSIT. Inicialmente, era composto de sete satélites de órbitas baixas e aproximadamente circulares, entrando em operação em 1964 para uso militar e, anos mais tarde, para a comunidade civil. Além dos satélites, o sistema contava com 18 estações terrestres ao redor do planeta para auxiliar na determinação do posicionamento [1].

Em virtude das limitações desse sistema, como o fornecimento da posição em apenas duas dimensões (latitude e longitude), baixa precisão, disponibilidade de posicionamento em apenas algumas horas do dia e dependente da região onde o receptor está, entre outros; a força aérea americana iniciou, em 1973, o desenvolvimento de um novo sistema de posicionamento global através de satélites artificiais com posicionamento determinado. Esse sistema foi denominado *NAVigation System with Time And Ranging & Global Positioning System* (NAVSTAR GPS) e, atualmente, é controlado pelo Departamento de Defesa dos EUA (DoD) [1],[4]-[6].

O sistema GPS apresenta diversas vantagens sobre o seu antecessor Transit. Apresenta alta precisão de posicionamento, disponibilidade ininterrupta, necessidade de um tempo curto de observação, determinação do posicionamento em quatro dimensões (latitude, longitude, altitude e tempo), possibilidade do cálculo da velocidade instantânea do equipamento receptor, além de outras vantagens. Através das informações espaciais e temporais, é possível determinar a velocidade, aceleração e direção de deslocamento do equipamento receptor, além de outras informações. Entretanto, sua precisão ou completa funcionalidade ficam comprometidas em determinadas condições climáticas desfavoráveis ao contínuo recebimento do sinal proveniente dos satélites. Apesar dessas situações especiais, o sistema GPS mostra-se muito mais adequado para uso em grande escala substituindo, assim, os outros sistemas de posicionamento no planeta consideravelmente [1]. Vale ressaltar que sistemas antigos de posicionamento ainda são utilizados em aplicações restritas e outros sistemas de posicionamento por satélites estão sendo testados e começando a operar ao redor do mundo, como o russo GLONASS e o europeu Galileo (ou “Global Navigation Satellite System” - GNSS) [4], [7].

Embora seu projeto tenha sido iniciado em 1973 com objetivo de suprir necessidades militares, seu uso tornou-se disponível para uso civil no início de 1996. Essa mudança de paradigma fez com que o sistema ganhasse uma ampla gama de possíveis aplicações.

Após ter sido liberado para o uso civil, o sistema GPS foi inicialmente utilizado por usuários com finalidades bastante específicas como levantamentos de campo de no estudo da Geodésia, levantamento de informações para aprimoramento da Cartografia, captura de informações em tempo real de desastres ecológicos, cálculos de áreas, navegação marítima por grandes embarcações e controle aéreo pelos órgãos competentes [1], [5].

A evolução da microeletrônica, principalmente na última década, permitiu que a complexidade dos circuitos aumentasse exponencialmente e as suas dimensões permanecessem constantes (ou até diminuíssem). Isso contribuiu para a popularização do GPS. Além do posicionamento, os dispositivos começaram a apresentar capacidade de desenvolver estratégias de deslocamentos sobre o planeta Terra, seja por meio terrestre, marítimo ou aéreo. Através do aprimoramento de sistemas computacionais atrelado à diminuição dos seus custos de fabricação resultou na integração desse sistema com as redes de comunicação de alto desempenho

disponíveis na maioria das regiões metropolitanas atualmente. Assim, diversos tipos de usuários aderiram às suas facilidades ao longo dos últimos anos.

Atualmente, é possível encontrar receptores GPS integrados a diversos tipos de dispositivos móveis comuns à vida contemporânea (chamados *gadgets*). Tantos dispositivos GPS dedicados quanto dispositivos multifuncionais, como os atuais *smartphones*, popularizaram-se com uma enorme rapidez na sociedade recentemente. Esses dispositivos costumam armazenar localmente informações de um território específico, descrevendo com alta fidelidade informações como nomes de ruas, estradas, relevo, rios, lagos, trilhos de trem e outros elementos que possam ser relevantes para o tipo de navegação desejado [8].

Através destas informações, diversos desenvolvedores de *software* comerciais têm apresentado diferentes alternativas de aplicativos que estão deixando a utilização desses dispositivos GPS mais atrativos e ao alcance de usuários cada vez mais simples. Funções como determinação de rotas baseadas em características configuráveis tornou esse equipamento bastante indicado para taxistas e mesmo motoristas comuns. As opções mais comuns para determinação de rota encontradas nos principais aplicativos GPS são: rota mais rápida (baseado na velocidade permitida de cada via), rota mais curta, rota considerando vias não pavimentadas. Enquanto isso, informações marítimas como profundidade do local, distância da costa, demarcação espacial de uma região em alto mar, entre outras, tornaram o uso de um equipamento GPS indispensável para a segurança de embarcações marítimas de pequeno e grande porte. Além disso, o sistema GPS atualmente é utilizado em larga escala para captura de informações de espécies de animais em extinção, sistemas antifurto de veículos, controle da frota de veículos de empresas de transporte público e de carga, agricultura de precisão, entre outras inúmeras funcionalidades novas que surgem da fusão desse sistema com as redes de comunicação [8], [9].

O objetivo do trabalho consiste em, além de estudar o funcionamento do sistema GPS, desenvolver uma simulação que ilustre o movimento orbital dos satélites que compõem o sistema juntamente com a rotação da Terra e determine o posicionamento de um usuário do sistema utilizando métodos matemáticos.

O Capítulo 2 descreve as principais características e conceitos utilizados no sistema GPS, seus segmentos, conceitos geométricos e métodos matemáticos implantados nos receptores GPS. O Capítulo 3 apresenta a simulação desenvolvida e os seus resultados são discutidos no Capítulo 4.

2 - Funcionamento do Sistema GPS

O processo de determinação de localizações na superfície terrestre pelo sistema GPS é feito, basicamente, pelo processamento dos sinais emitidos constantemente pela constelação de satélites artificiais em órbita ao redor do planeta Terra. Esses sinais contêm diversas informações do sistema GPS e do satélite emissor (efemérides) que, através do método de trilateração, torna-se possível determinar as coordenadas espaciais do receptor. As efemérides correspondem a uma importante parte das informações transmitidas pelos satélites. Consistem de um conjunto de parâmetros que descrevem o posicionamento atual e, por meio delas, torna possíveis os receptores definirem com precisão o posicionamento do satélite que a transmitiu.

Para determinação da latitude, longitude e altitude, é necessário um conjunto de três satélites dentro do diagrama de radiação da antena receptora. Adicionando-se um quarto satélite nesse conjunto, podem-se obter informações de tempo do receptor, o que permite calcular grandezas físicas como velocidade de locomoção, aceleração, direção de deslocamento, entre outras, a partir de medições sequenciais.

Esse método exige que todo o sistema esteja sincronizado o mais perfeitamente possível. Para isso, foi criado um sistema temporal de referência denominado tempo GPS e é medido por relógios atômicos de Césio ou Rubídio. Esse sistema de referência é controlado pela *Operational Control System* do *U. S. Naval Observatory* (OCS-USNO) e se mantém com uma diferença máxima de 1 µs do sistema *universal time coordinated* (UTC), utilizado como principal referência de tempo no mundo desconsiderando o "*leap second*", que consiste no ajuste de 1 segundo no tempo UTC, que ocorre no final de Junho ou Dezembro, para melhorar a aproximação com a rotação terrestre. Todo satélite contido na constelação do sistema GPS contém, pelo menos, quatro relógios desses para enviar a informação de tempo com a maior precisão possível [1], [4].

A medida de tempo do sistema GPS é feita contando-se a quantidade de segundos na semana corrente, ou seja, a partir da última meia-noite de sábado para domingo. Além disso, é feito o controle das semanas no sistema GPS através do "número de semanas GPS", sendo a semana zero iniciada em 6 de Janeiro de 1980 e codificada por 10 bits em cada mensagem emitida pelos satélites. Dessa forma, o número máximo de semanas GPS são 1023, exigindo que esse número seja reiniciado

a cada 1024 semanas, aproximadamente 20 anos, o que é chamado de "*end of week*" (EOF) ou "*roll over week*" (ROW).

2.1 – Características Operacionais

A determinação do posicionamento de um receptor GPS é feita, basicamente, através do cálculo das pseudodistâncias de três satélites visíveis de localizações predeterminadas e, através do método de triangulação, é determinado geometricamente sua posição. Um quarto satélite é utilizado, pois o relógio do receptor não é sincronizado com os relógios dos satélites, sendo sua posição utilizada para definir a posição da antena receptora dentro do espaço permitido pelas pseudodistâncias.

Os cálculos dessas pseudodistâncias são feitos determinando-se o tempo de propagação dos sinais emitidos pelos satélites até o receptor. O momento da transmissão é comparado com o tempo do relógio do receptor e, conhecida a velocidade de fase das ondas eletromagnéticas transmitidas, tem-se a distância percorrida pela onda.

Há dois tipos de serviço transmitidos ininterruptamente pelo sistema GPS. O *standard positioning service* (SPS) consiste no serviço de localização disponível para a comunidade civil e utiliza o código C/A (*coarse* ou *clear/acquisition-code*). Esse serviço apresentava até 2000 o efeito de degradação SA (s*elective availability*) do sinal para diminuir a precisão do serviço. O efeito SA consiste na manipulação proposital dos dados transmitidos pelos satélites, como o tempo de transmissão do sinal e/ou dos parâmetros contidos nas efemérides. Por outro lado, o *precise positioning service* (PPS) utiliza o código-P (*precision-code*), correspondendo ao serviço de maior precisão e disponível apenas para usuários militares e restritamente autorizados pelo DoD. Essa restrição é feita aplicando uma técnica de criptografia ao código-P gerando, assim, um novo código denominado Y. Devido a isso, o código-P é comumente escrito código P(Y) [1], [4].

Esses códigos binários pseudoaleatórios, denominados código PRN (*pseudo random noise*) são utilizados na modulação dos sinais elétricos em um método denominado *code division multiple access* (CDMA), técnica de modulação digital amplamente utilizada na telefonia terrestre móvel [1], [4].

O código-C/A corresponde a uma sequência binária gerada na frequência de 1,023 MHz, ou seja, a cada microssegundo seu conteúdo é repetido e utilizado na modulação das mensagens de navegação do serviço SPS. Enquanto isso, o código-P é gerado na frequência de 10,23 MHz, isto é, são gerados 10,23 Mbits por segundo (Mbps) para modular as mensagens de navegação referentes ao serviço PPS. Devido à sua maior frequência e, portanto, menor comprimento efetivo de onda, o código-P é capaz de apresentar melhores precisões com sua utilização. Corresponde a um código binário de valores fixos longos, repetindo-se, aproximadamente, a cada 266 dias (38 semanas). Cada satélite transmite uma parcela deste código repetidamente a cada semana, sendo reiniciada a cada nova semana GPS (0:00 UTC de sábado para domingo). Essa parcela do código-P, utilizada exclusivamente para cada satélite, recebe um número de identificação que é comumente utilizado para identificação do mesmo [1], [4]. Mais informações de como esses códigos são gerados são encontradas em [8].

Para o processo de demodulação, é necessário conhecimento do código utilizado para recuperar a mensagem de navegação. Por isso, foi implementada a técnica de encriptação do sinal no código-P denominado *anti-spoofing* (AS) para restringir a utilização do serviço PPS a somente usuários autorizados pelo DoD.

O sistema GPS utiliza duas frequências portadoras L_1 e L_2 na banda L de Micro-ondas (desconsiderando-se a nova portadora L_5 em 1379,91 MHz ainda em desenvolvimento). São derivadas da frequência fundamental f_0 de 10,23 MHz gerada pelos relógios atômicos de Cs ou Rb presentes nos satélites e estações de controle terrestres. É de suma importância destacar a necessidade de osciladores de alta precisão para a geração desta frequência, dado que a sincronização do sistema depende disso [1], [4], [8].

A portadora L_1 de 1575,42 MHz (154 x 10,23 MHz, aproximadamente 19 cm de comprimento de onda) transmite ambos os códigos C/A e P. Esses códigos são transmitidos continuamente por quadratura de fase (atraso de 90º na fase dos sinais). Enquanto isso, a portadora L_2 de 1227,60 MHz (120 x 10,23 MHz, aproximadamente 24,4 cm de comprimento de onda) transmite apenas o código P. Ambas as portadoras são transmitidas por todos os satélites em funcionamento no sistema GPS.

Por último, têm-se as mensagens de navegação que contém diversas informações, como tipo de código utilizado (C/A ou P), informações sobre a órbita e saúde do satélite transmissor, correções do relógio do satélite e de condições atmosféricas, além de outras informações momentâneas do sistema GPS. Essas

informações são estruturadas em blocos binários e gerados a uma taxa de 50 bps. Esses bits referentes à mensagem de navegação são modulados por *phase-shift keying* (PSK), que consiste em representar cada valor binário em fases diferentes. No caso de apenas dois símbolos (0 e 1, ou ±1), a técnica é denominada BPSK (binary PSK) e cada valor corresponde a uma fase distante 180º da outra [1], [4].

As mensagens de navegação são estruturadas por quadro, representado na Fig. 2.1, e transmitidas na taxa de 50 bps (muito inferior às taxas de 10,23 Mbps do código-P e 1,023 Mbps do código C/A). Cada quadro contém um total de 1500 bits levando, portanto, 30 segundos para sua transmissão. Esses bits são distribuídos em 5 sub-quadros com duração de 6 segundos cada. Estes, por sua vez, são divididos em 10 palavras de 30 bits cada [1], [4].

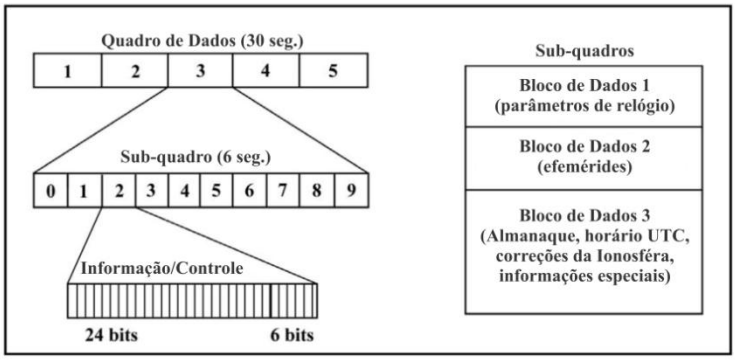


Figura 2.1 - Estrutura da mensagem de navegação GPS. Traduzido de [4].

A primeira palavra de cada sub-quadro consiste em uma *telemetry word* (TLW), que consiste em um padrão binário usado para sincronização dos sub-quadros. A segunda palavra de cada sub-quadro corresponde a *hand over word* (HOW), que contém um parâmetro denominado *contador-Z*. Este parâmetro corresponde a um valor inteiro que armazena a quantidade de intervalos de 1,5 segundo passado desde o início da semana GPS, identificando assim o instante em que a mensagem de navegação foi gerada. A função principal do contador-Z é auxiliar o acesso dos receptores autorizados ao código-P a partir do código C/A, de comprimento muito menor. Isso é necessário, pois para utilizarem o código-P, os receptores devem estar precisamente sincronizados com o tempo GPS e em um local de coordenadas conhecidas. Desta forma, é possível utilizar o código-P para determinar o posicionamento em poucos segundos de observação [1], [4].

Os quadros são divididos também em blocos de acordo com o conteúdo das palavras restantes que contém as informações úteis para o posicionamento.

O bloco de dados 1 corresponde ao primeiro sub-quadro e contém a semana GPS correspondente, parâmetros de correção do tempo GPS (correção do relógio de pouca precisão do receptor) e informações sobre o estado do satélite emissor da mensagem.

O bloco de dados 2 corresponde ao segundo e terceiro sub-quadros, contendo todos os parâmetros correspondentes à órbita do satélite emissor da mensagem para que o receptor possa calcular suas coordenadas e, a partir delas, determinar sua posição.

Por último, o bloco de dados 3 corresponde ao quarto e quinto sub-quadros que completam o quadro correspondente. Contém um conjunto de informações denominado *almanaque*. Essas informações descrevem o estado global do sistema GPS, com informações sobre os satélites disponíveis, correções no tempo GPS, informações do horário UTC, parâmetros para correções dos efeitos da ionosfera, além de informações particulares para usuários autorizados. Essas informações são de grande importância para determinar a geometria atual dos satélites, aperfeiçoando a precisão dos resultados. Diferentemente dos blocos 1 e 2, o bloco de dados 3 não repete suas informações a cada 30 segundos. O almanaque é dividido em 25 páginas que são transmitidas sequencialmente através dos dois últimos sub-quadros de cada quadro, sendo que o conteúdo completo é enviado depois de 12,5 minutos [4].

Portanto, considerando um sinal de portadora não modulado por $L(t) = A.cos(\omega t)$, o código-P de $P(t)$, o código-C/A de $C(t)$ e a mensagem de navegação de $D(t)$, os sinais das portadora L_1 e L_2 modulados podem ser descritos por $L_1(t)$ e $L_2(t)$:

$$L_{1i}(t) = A_{1i}P_i(t)D_i(t)\,\mathrm{sen}(\omega_1 t) + A_1 C_i(t)D_i(t)cos(\omega_1 t)$$

$$L_{2i}(t) = A_{2i}P_i(t)D_i(t)\cos(\omega_2 t)$$

Onde:

i: sinais referentes ao i-ésimo satélite em órbita no sistema GPS,

A_1, A_2: Amplitudes das portadoras L_1 e L_2,

$sen(\omega t)$, $cos(\omega t)$: sinal senoidal das portadoras defasado de 90º,

$P(t)$: código-P binário com valores ±1,

$C(t)$: código-C/A binário com valores ±1,

D(t): mensagem de navegação binária gerada com valores ±1.

Como a portadora L_1 transmite ambos os códigos P e C/A em quadratura de fase, seu sinal é constituído de dois sinais senoidais defasados de 90º de frequência f_1 = 1575,42 MHz com suas amplitudes moduladas pela mensagem de navegação gerada e o código utilizado. No caso da portadora L_2, está presente apenas o sinal referente ao código-P transmitido na frequência f_2 de 1227,60 MHz.

A Fig. 2.2 [4] ilustra as etapas da geração do sinal GPS emitido pelos satélites.

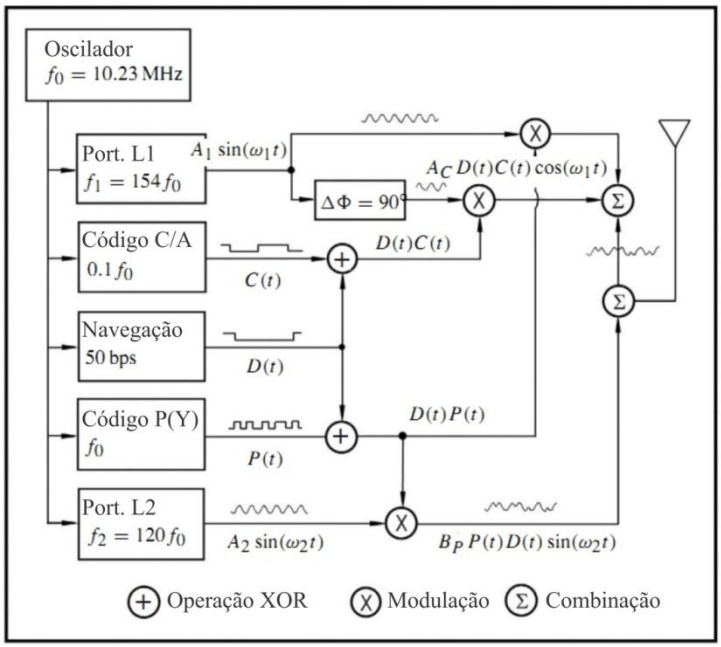


Figura 2.2 - Geração do sinal GPS. Traduzido de [4].

A construção dos sinais inicia-se de um oscilador de alta precisão de frequência fundamental f_0. A partir dele, são geradas as duas portadoras L_1 e L_2, os códigos C/A e P(Y) usados na modulação CDMA e as mensagens de navegação. É aplicada uma operação ou-exclusivo (XOR) entre os bits dos códigos com as mensagens de navegação e o resultado aplicado na modulação BPSK dos sinais das portadoras [8]. Como a portadora L_1 transmite os códigos C/A e P(Y) em quadratura de fase, seu sinal senoidal é duplicado e um deles atrasado 90º com relação ao outro antes da modulação. Por fim, é feita combinação dos 3 sinais em uma mesma linha de transmissão, que transmite o sinal resultante para a(s) antena(s).

2.2 – Segmentos do Sistema GPS

O conjunto de elementos que compõem o sistema GPS pode ser dividido em três Segmentos: Espacial, de Controle e de Usuário.

2.2.1 – Segmento Espacial

O segmento espacial é constituído pela constelação de satélites em órbita ao redor do planeta Terra e que emite constantemente as efemérides para a superfície terrestre. Essa constelação foi definida inicialmente para conter 21 satélites ativos (mais três satélites de reserva), sendo que, atualmente, é composta de 32 satélites [1], [10]. Percorrem órbitas elípticas de excentricidade menor que 0,02 e semi-eixo de, aproximadamente, 26600 km correspondendo a uma altitude média de 20200 km acima da superfície terrestre. Apresenta período fixo de 11h 57’ 58,3” e inclinação de 55º do plano orbital com relação ao Equador. Essa disposição foi definida para garantir a presença de, pelo menos, quatro satélites no horizonte visível de qualquer ponto na superfície terrestre a qualquer hora do dia, para que a disponibilidade de obtenção das coordenadas latitude, longitude, altitude e tempo sejam garantidas independente da hora ou local de observação.

Os satélites são dispostos em seis planos orbitais denominados A a F com ângulo de 60º entre eles (Fig. 2.3), movimentando-se a uma velocidade de 3,87 km/s (aproximadamente 13930 km/h) e utiliza sistemas de propulsão para auxiliar no controle de posicionamento da órbita definida. Cada satélite pode ser identificado por maneiras distintas: seu número sequencial de lançamento SVN (“*space vehicle number*”), seu código PRN (“*pseudo random noise*”), código definido pela NASA e código de identificação internacional. Geralmente, a identificação é feita pelos códigos PRN correspondentes aos satélites utilizados no momento da observação [1].

Os satélites utilizados no sistema GPS são divididos em famílias de satélites denominados blocos e são agrupados de acordo com características de projeto semelhantes entre eles. A Tabela 2.1 resume os diferentes blocos de satélites lançados ao longo dos anos para prover o funcionamento do sistema GPS.

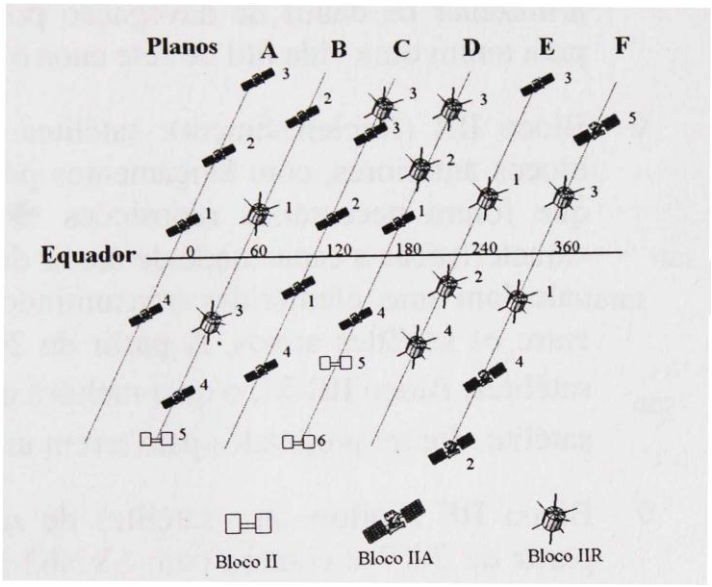


Figura 2.3 - Planos orbitais com satélites (configuração de setembro de 2005) [1].

Tabela 2.1 - Blocos de satélites e principais características [11]

Bloco	Lançamento	Quantidade	Peso (kg)	Fabricante
I	1978 - 1985	11	759	Rockwell International
II	1989 – 1990	9	1660	Rockwell International
IIA	1990 - 1997	19	1816	Rockwell International
IIR	1997 - 2004	13	2032	Lockheed Martin
IIR-M	2005 - 2008	8	2032	Lockheed Martin
IIF	2010 - 2014	12	1545	Boeing
III	2014 (planejado)	8	?	Lockheed Martin

Os primeiros satélites (bloco I) colocados em órbita serviram para contribuir com a pesquisa e desenvolvimento do sistema GPS, sem as degradações de sinais AS e SA, discutidos a seguir. Foram divididos em dois planos orbitais de 63 º com a reta do Equador. Foram muito importantes para as primeiras etapas de desenvolvimento do sistema GPS e, atualmente, nenhum satélite deste bloco continua em órbita.

O serviço começou a ser implantado de fato a partir de fevereiro de 1989 com os satélites dos blocos II e IIA, desenvolvidos pela *Rockwell International*, sendo o formato projetado da constelação completamente em órbita apenas em 1994. Os satélites do bloco II foram projetados para operarem 14 dias sem comunicação com o segmento de controle e iniciaram o envio de efeitos de degradação de sinais (AS e SA).

Nesses satélites foram implantados métodos de degradação do sinal que visavam limitar a precisão do serviço para usuários não autorizados devido a preocupações militares sobre a potencialidade do sistema. A técnica denominada *anti-spoofing* (AS) evitava que usuários não autorizados tivessem acesso ao código-P, disponível apenas para militares e agências governamentais americanos, além de outros usuários restritos. Além dela, a técnica SA era utilizada para diminuir a precisão do serviço, disponível para os usuários civis pelo código C/A, inserindo informações menos precisas nas efemérides. Essa última foi desativada em 2000 quando o governo americano permitiu que usuários civis tivessem acesso ao sistema GPS com maior precisão.

Os satélites do bloco IIA (*advanced*) entraram em funcionamento visando, além de outras melhorias, uma maior autonomia com relação ao segmento de controle, podendo armazenar informações de navegação e, assim, aumentar para 180 dias o intervalo entre as comunicações com as estações terrestres. Entretanto, essa medida reduziu a precisão do serviço oferecido, sendo solucionado com os satélites do bloco IIR (*replenishment*), desenvolvidos pela *Lockheed Martin* para substituir os satélites em operação. Esses satélites operam comunicando-se com as estações terrestres em intervalos de 4 dias ou de 180 dias no modo de navegação autônoma (AUTONAV), que utiliza a técnica "*UHF cross link ranges*" para calcular as distâncias entre os satélites e utilizar essas informações para manter a precisão das efemérides.

Com os satélites do bloco IIR-M, o sistema adicionou um segundo sinal civil denominado L_2C na frequência L_2 e um novo código militar M nas frequências L_1 e L_2 a fim de aperfeiçoar ainda mais a precisão para todos os usuários. Em maio de 2010, foi colocado em órbita o primeiro satélite do novo bloco IIF (*follow-on*) de satélites GPS. Esses satélites, produzidos pela Boeing, têm como principal característica, além da transmissão dos sinais antigos com precisão aperfeiçoada, a implantação de um novo sinal L_5 na frequência de 1379,91MHz desenvolvido para aplicações *safety-of-life* (SoL), como navegação para aviação civil. Esse sinal busca corrigir efeitos degradantes causados pela ionosfera e é transmitido com maior potência e maior largura de banda, além de outras vantagens, que deixam o sinal mais robusto para possíveis problemas de transmissão que podem ser detectados [4], [10].

O último satélite adicionado à constelação GPS corresponde ao satélite SVN63 de código PRN 01 lançado no dia 16 de julho de 2011 e corresponde ao segundo satélite da família IIF em órbita [12], [13]. Os projetos futuros referentes ao segmento espacial consistem no lançamento de mais 10 satélites do bloco IIF já em fabricação e

desenvolvimento de 30 satélites do bloco III, que apresentarão, além dos sinais e códigos atualmente em funcionamento, novas melhorias visando aperfeiçoar a disponibilidade e precisão do sistema GPS [10].

2.2.2 – Segmento de Controle

O segmento de controle, ou *operational control system* (OCS), corresponde a um conjunto de estações espalhadas ao redor do mundo. Sua finalidade é monitorar e controlar as condições operacionais da constelação de satélites GPS em órbita. Basicamente, é feita o monitoramento das posições de cada satélite periodicamente recebendo os sinais emitidos por eles e através deles, são calculadas possíveis manobras para correção da sua órbita, analisado o sincronismo do tempo GPS entre eles e o OCS, retransmitindo, posteriormente, sinais com as modificações necessárias para o correto funcionamento do sistema.

O OCS é composto de uma *master control station* (MCS), diversas *monitor stations* (MS) e uma rede de antenas terrestres para comunicação com os satélites denominada *ground control stations* (GCS). A MCS era localizada na Califórnia, EUA, mas atualmente está sob o controle do *U.S. Air Force Space Command, Second Space Operation Squadron* (2SOPS), localizado no *Consolidated Space Operation Center*, Colorado, EUA. Sua função é coordenar as diversas funções do OCS recebendo as informações capturadas pelas MS, processar as operações de controle e correções e transmiti-las para a GCS para que sejam enviadas para os satélites. Nesse processamento, são considerados atrasos na ionosfera para atenuar a degradação das mensagens de controle recebidas pelos satélites. Além disso, é responsabilidade da MCS controlar o tempo GPS e garantir a sincronização do sistema, atualizando os relógios dos satélites pelas mensagens de controle. Esse processo ocorre a cada oito horas ou pelo menos uma vez ao dia [1], [4].

As estações de monitoramento MS são compostas de seis estações espalhadas ao redor do mundo, além de mais 14 estações da *National Geospatial-Intelligence Agency* (NGA), adicionadas ao sistema a partir de 2005, que auxiliam o monitoramento dos satélites recebendo suas efemérides transmitidas na banda L. Dessa forma, toda a constelação de satélites é permanentemente visível às estações de monitoramento, além de cada satélite ser sempre visível a duas estações de

monitoramento, garantindo melhores cálculos de posição e efemérides. Cada estação contém oito canais independentes de monitoramento, possibilitando capturar, teoricamente, sinais de até oito satélites visíveis posicionados 15º acima do horizonte da MS já que, abaixo disso, atraso ionosférico e troposférico invalidam a leitura das informações [14].

Por fim, as GCS correspondem a quatro antenas terrestres dispostas ao redor do mundo que recebem as mensagens de controle da MCS e as retransmitem para a constelação de satélites. São antenas parabólicas com diâmetro aproximado de 10 metros que operam na banda S (faixa entre 2 e 4 GHz) na comunicação com os satélites visíveis [14]. A Fig. 2.4 [8] ilustra a localização das antenas terrestres e estações de monitoramento.

Figura 2.4 - Antenas terrestres (localizadas em Cape Canaveral, Ascension, Diego Garcia e Kwajalien) e estações de monitoramento (localizadas em Schriever, Hawaii, Cape Canaveral, Ascension, Diego Garcia e Kwajalien). Traduzido de [8].

Os softwares para processamento de dados das MS e GCS são hospedados em estações de trabalho *Sun* com sistema operacional UNIX e, além disso, a comunicação entre eles e a MCS é feita através de enlaces de dados baseados no protocolo TCP/IP [8].

2.2.3 – Segmento de Usuário

O segmento de usuário consiste dos equipamentos receptores das efemérides transmitidos continuamente pela constelação de satélites GPS. Correspondem a equipamentos espalhados na superfície terrestre e disponíveis para veículos, embarcações marítimas e aviões, além de equipamentos móveis projetados para diversas finalidades distintas.

Cada receptor é projetado para uma determinada aplicação que pode ter acesso a serviço disponível aos civis pelo Código C/A ou também para serviço militar através do Código P com maior precisão. Existe uma enorme diversidade nos tipos de receptores GPS existentes atualmente, tanto no âmbito das antenas utilizadas para a recepção das ondas eletromagnéticas emitidas pelas antenas dos satélites quanto do restante do *hardware* responsável pela amplificação desses sinais, decodificação das informações das efemérides e processamento das mesmas.

2.3 – Posicionamento GPS

2.3.1 – Elementos Geográficos

Para o correto funcionamento do sistema GPS, é necessário definir alguns conceitos geográficos para a padronização do processo de determinação de posicionamento de pontos na superfície terrestre (ou próxima a ela). O globo terrestre pode ser simplificadamente representado por uma esfera com centro localizado no seu centro de massa e raio aproximado de 6370 km (raio médio do círculo equatorial) [15].

A Terra apresenta uma rotação ao redor do seu eixo, que corresponde à reta que hipoteticamente atravessa a superfície terrestre no pólo norte e pólo sul. Todo plano que contém seu eixo é denominado de plano meridiano e sua intersecção com a superfície esférica aproximada da Terra é denominado de círculo meridiano (ou, apenas, meridiano). O principal meridiano, utilizado como referência para os outros, é o meridiano de Greenwich, que atravessa a cidade de Londres, Inglaterra. A partir deste, define-se como longitude o ângulo λ entre um meridiano qualquer com o

meridiano de Greenwich, variando de 0º a 180º (ou 0 h a 12 h) positivo ou negativo para o leste ou oeste, respectivamente [1].

Além disso, existem os paralelos, círculos resultantes da intersecção de planos perpendiculares ao eixo terrestre com a superfície esférica terrestre. O plano que gera o maior paralelo possível é denominado de plano equatorial e este paralelo denominado Equador, o qual divide a superfície terrestre em dois hemisférios: boreal (ou norte) e austral (ou sul). Com isso, é definida a latitude como o ângulo ϕ entre o vetor normal a um ponto terrestre com o plano equatorial. Apresenta valores de 0º a 90º positivos ou negativos para o norte ou sul, respectivamente, a partir do Equador [1].A Fig. 2.5 ilustra a Terra com seus principais paralelos e meridianos, além de uma posição P dada em latitude e longitude.

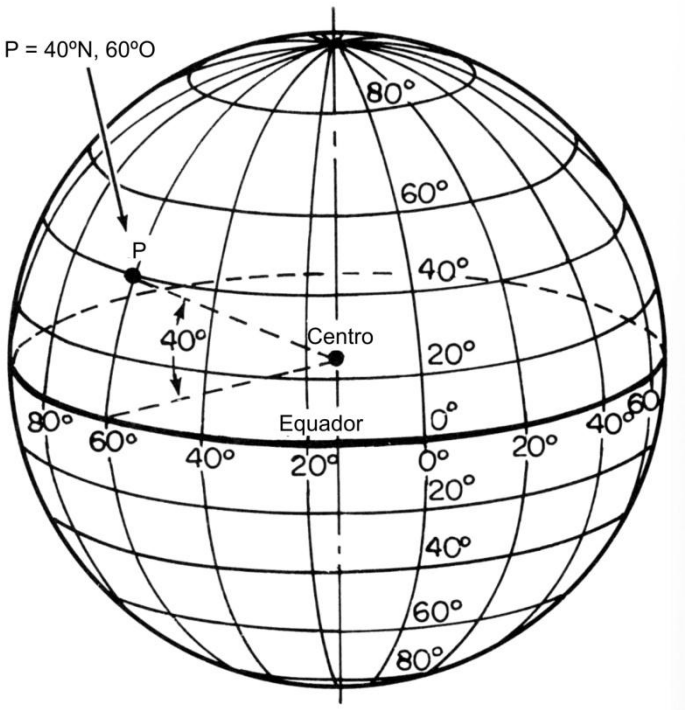


Figura 2.5 - Planeta Terra com alguns paralelos e meridianos.

A latitude ϕ e longitude λ medidas em graus, juntamente com a altitude *h* medida em metros (a partir do nível do mar), correspondem às grandezas de maior utilização na Geografia. Entretanto, o sistema de coordenadas cartesiano mapeado pelos eixos *x*, *y* e *z* é mais adotado matematicamente. Com isso, grande parte dos modelos e ferramentas utilizadas nas diversas áreas da engenharia utiliza este sistema de coordenadas como padrão. Dessa forma, o sistema de coordenadas ECEF ("*earth-centered earth-fixed coordinate system*") é utilizado para representar pontos referentes

à Terra através dos eixos do sistema cartesiano. O sistema ECEF contém sua origem no centro da esfera referente à Terra, com o plano *xy* localizado sobre o plano equatorial (eixos *x* e *y* nas direção de longitude 0º e +90º, respectivamente) e eixo z sobre o eixo de rotação terrestre com valores positivos direcionados para o norte, sendo, portanto, o sistema ECEF fixo com relação à rotação terrestre [8].

2.3.2 – Sistema de Referência WGS84

Embora o sistema de coordenadas ECEF seja de simples utilização, o planeta Terra não é bem representado por uma esfera. Através de observações do campo gravitacional em diversos pontos na superfície terrestre, concluiu-se que a Terra seria mais bem representada por uma elipsoide de revolução geocêntrica. Assim, foi definido o sistema de referência WGS ("*World Geodetic System*"), que apresenta os eixos *x*, *y* e *z* dispostos conforme o sistema ECEF, embora com semieixo maior *a* contido no plano equatorial e semieixo menor *b* na direção do eixo *z*, conforme mostra a Fig. 2.6 [8].

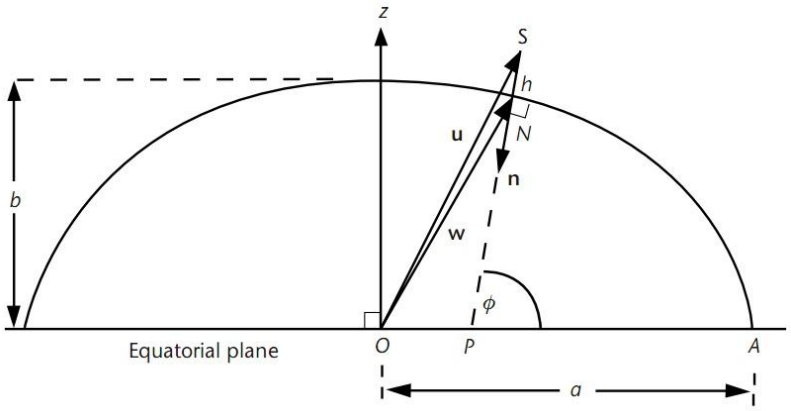


Figura 2.6 - Modelo elipsoidal terrestre [8].

A Figura 2.6 ilustra a representação da Terra por meio do modelo elipsoidal com semieixos maior *a* e menor *b*, vetor de posição ***u*** de um usuário *S* com altitude *h*, vetor de posição ***w*** do ponto *N* na superfície (ponto mais próximo do elipsoide de referência do usuário) e a latitude $\boldsymbol{\phi}$ referente ao usuário.

Após algumas correções, a versão WGS84 foi adotada como padrão para utilização no sistema GPS pelo DoD em 1987. O sistema WGS84 define parâmetros

do modelo terrestre que representam geometricamente e fisicamente o planeta Terra. Os principais parâmetros são listados na Tabela 2.2 [1], [8].

Tabela 2.2 - Principais parâmetros do WGS84 [1], [8]

Parâmetros	**Valores**
Semi-eixo maior	a = 6378137 m
Semi-eixo menor	b = 6356752,31 m
Excentricidade	e = 0,08181918994
Achatamento geométrico	f = 1/298,257223563
Velocidade angular da Terra	$\omega_E = 7929115*10^{-11}$ rad/s
Constante gravitacional terrestre	$\mu = 3989005*10^{8}$ m^3/s^2

A excentricidade *e,* a excentricidade segunda ***e'*** e o achatamento geométrico *f* são valores adimensionais comumente utilizados para descrever elipses e são calculados por meio de [4]:

$$e = \sqrt{1 - \frac{b^2}{a^2}} \tag{2.1}$$

$$e' = \sqrt{\frac{a^2}{b^2} - 1} = \frac{a}{b}e \tag{2.2}$$

$$f = 1 - \frac{b}{a} \tag{2.3}$$

Uma vez obtido o posicionamento do usuário pela determinação do vetor $\boldsymbol{u} = (x_u, y_u, z_u)$, é possível converter sua posição para latitude ϕ, longitude λ e altura h.

Primeiramente, calcula-se a longitude λ por [8]:

$$\begin{aligned} \lambda &= \tan^{-1}\frac{y_u}{x_u}, \qquad x_u \geq 0 \\ \lambda &= 180° + \tan^{-1}\frac{y_u}{x_u}, \qquad x_u < 0 \; e \; y_u \geq 0 \\ \lambda &= -180° + \tan^{-1}\frac{y_u}{x_u}, \qquad x_u < 0 \; e \; y_u < 0 \end{aligned} \tag{2.4}$$

Posteriormente, pode-se calcular a latitude ϕ e altura *h* de diferentes formas. Vários métodos de cálculo das coordenadas geodésicas a partir de coordenadas geocêntricas foram desenvolvidos. O método iterativo de Bowring, apresentado em [16], é demonstrado abaixo.

Inicialmente, calculam-se os valores de *p* e *tan* u_0*:*

$$p = \sqrt{x^2 + y^2}$$

$$\tan \mathrm{u}_0 = \left(\frac{z}{p}\right)\left(\frac{a}{b}\right)$$

Inicia-se um laço iterativo:

$$\cos^2 u = \frac{1}{1 + \tan^2 u_0}$$

$$\sin^2 u = 1 - \cos^2 u$$

$$\tan \phi = \frac{z + e'^2 b \sin^3 u}{p - e^2 a \cos^3 u}$$

$$\tan \mathrm{u} = \left(\frac{b}{a}\right) \tan \phi$$

Verifica-se se *tan u = tan* u_0. Se for falso, é feito *tan* u_0 *= tan u* e continua-se o laço. Se for verdadeiro, é terminada a conversão calculando o valor de *N*, por 2.5, e *h*, por 2.6 ou 2.7:

$$N = \frac{a}{\sqrt{1 - e^2 \sin^2 \phi}} \tag{2.5}$$

Para $\phi \neq \pm 90^o$:

$$h = \frac{p}{\cos \phi} - N \tag{2.6}$$

Ou para $\phi \neq 0^o$:

$$h = \frac{z}{\sin \phi} - N + e^2 N \tag{2.7}$$

Segundo [16], este método iterativo pode ser considerado exato por não exigir mais do que uma iteração e apresenta erros no cálculo de ϕ desconsideráveis para h ≥

-6300 km (ordem de 10^{-8}"), embora não seja indicado para valores de ϕ próximos de ±90º.

Inversamente, é possível calcular diretamente o vetor $\boldsymbol{u} = (x_u, y_u, z_u)$ de coordenadas cartesianas a partir de (ϕ, λ, h) por meio de 2.8 [8]:

$$x_u = \frac{a\cos\lambda}{\sqrt{1+(1-e^2)\tan^2\phi}} + h\cos\lambda\cos\phi$$

$$y_u = \frac{a\sin\lambda}{\sqrt{1+(1-e^2)\tan^2\phi}} + h\sin\lambda\cos\phi \qquad (2.8)$$

$$z_u = \frac{a\,(1-e^2)\sin\phi}{\sqrt{1-e^2\sin^2\phi}} + h\sin\phi$$

2.3.3 – Geometria do Posicionamento

O sistema GPS permite determinar o posicionamento em três dimensões de uma antena receptora que capta sinais de satélites que estão em órbita no planeta Terra. Recebendo esses sinais e determinando o tempo de propagação da antena do satélite até a antena receptora ($t_u - t_s = \Delta t$), é possível determinar a distância aproximada d desses dois pontos [1]. Dessa forma, a pseudodistância é calculada por 2.9:

$$d = c(t_u - t_s) = c\Delta t \qquad (2.9)$$

na qual velocidade da luz no vácuo, $c = 3.10^8$ m/s.

Considerando primeiramente um cenário em duas dimensões (x, y) e sendo conhecido o posicionamento de um satélite emissor s_1 (x_1, y_1), a região do espaço que apresenta uma mesma distância d_1 deste satélite corresponde a uma circunferência de centro na posição do satélite e raio d_1. No caso da determinação de um posicionamento na superfície terrestre (aproximada por uma circunferência neste cenário 2D), o posicionamento resulta em determinar qual dos dois pontos a_1 e a_2 onde se interceptam as duas circunferências corresponde à posição do receptor (salvo condições especiais onde as duas circunferências não se interceptam ou interceptam em apenas um ponto). Veja Fig. 2.7 (a). Essa questão é solucionada adotando um

segundo satélite s_2 de posição conhecida (x_2, y_2) e de distância d_2 calculada. Dessa forma, apenas um dos pontos a_1 e a_2 satisfaz esses distanciamentos, determinando o posicionamento 2D do receptor na superfície terrestre [1]. A situação é ilustrada na Fig. 2.7 (b):

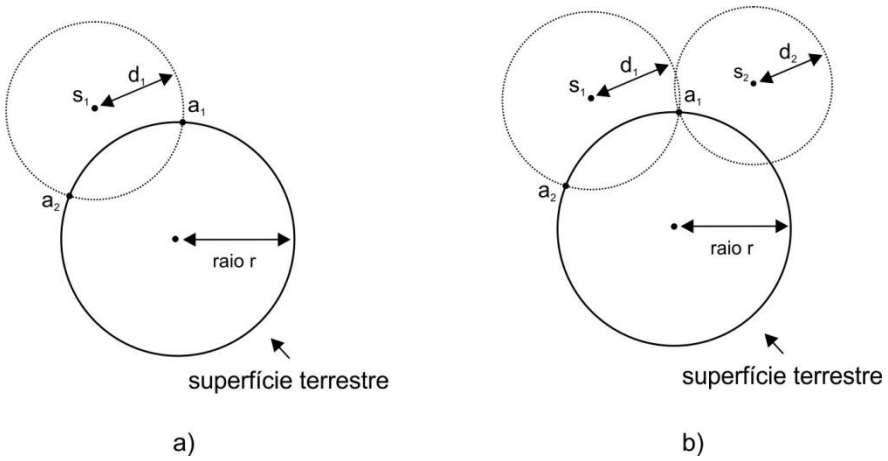


Figura 2.7 - Geometria 2D do posicionamento. (a) Apenas um satélite. (b) Dois satélites

Sendo a posição do usuário a ser determinada (x_u, y_u) e a esfera referente à Terra de raio r centrada na origem, é possível modelar essa situação através da solução do sistema abaixo de equações da distancia entre pontos distintos para o caso 2D (Eq. 2.10):

$$\begin{cases} (x_u - x_1)^2 + (y_u - y_1)^2 = {d_1}^2 \\ (x_u - x_2)^2 + (y_u - y_2)^2 = {d_2}^2 \\ {x_u}^2 + {y_u}^2 = r^2 \end{cases} \quad (2.10)$$

De forma análoga e considerando o caso real de 3 dimensões (x, y, z - ou latitude, longitude e altitude, dependendo do sistema de coordenadas adotado), essas circunferências são consideradas esferas com centro na posição de cada satélite s_i (x_i, y_i, z_i) e de raio d_i. A intersecção da esfera de um satélite s_1 com a Terra corresponde a uma circunferência (Fig. 2.8 a) e de duas esferas referentes a s_1 e s_2 com a Terra corresponde a dois pontos distintos a_1 e a_2 (salvo os casos onde não se interceptam ou interceptam em apenas um ponto) (Fig. 2.8 b). Assim, a posição do receptor é determinada geometricamente pela intersecção de três esferas de satélites com o

planeta Terra, determinando o único ponto possível a_1 que apresenta as distâncias d_i medidas de cada satélite (Fig. 2.8 c).

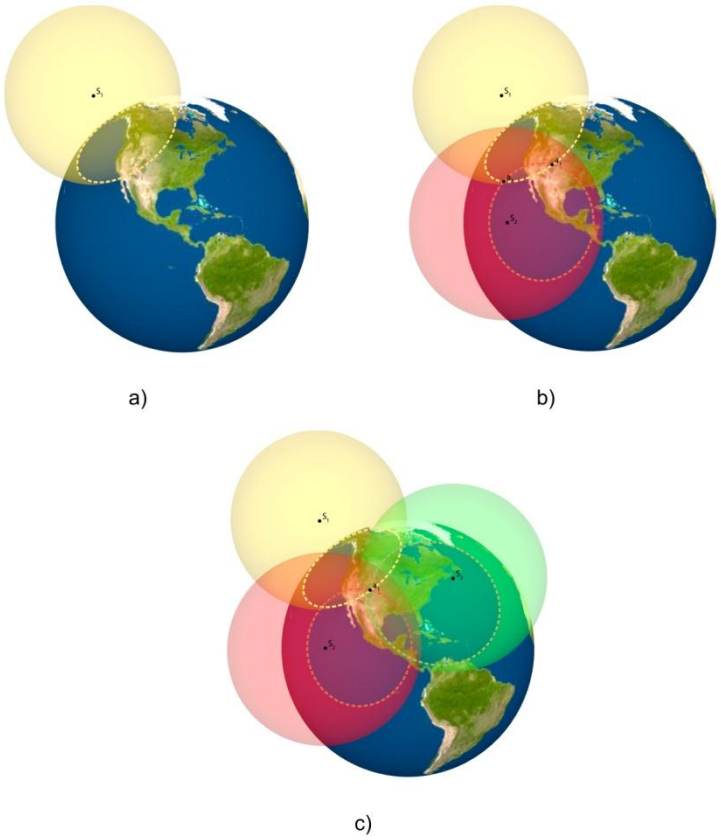


Figura 2.8 - Geometria 3D do posicionamento. (a) Apenas um satélite. (b) Dois satélites. (c) Três satélites

A determinação do posicionamento do receptor (x_u, y_u, z_u) consiste na resolução do sistema de três equações e três incógnitas:

$$\begin{cases} (x_u - x_1)^2 + (y_u - y_1)^2 + (z_u - z_1)^2 = {d_1}^2 \\ (x_u - x_2)^2 + (y_u - y_2)^2 + (z_u - z_2)^2 = {d_2}^2 \\ (x_u - x_3)^2 + (y_u - y_3)^2 + (z_u - z_3)^3 = {d_3}^2 \end{cases} \quad (2.11)$$

Essas equações são simplificadas porque, como visto na seção 2.3.2, a Terra é mais bem representada por uma elipsoide de revolução ao invés de uma esfera. Além disso, é considerado que os relógios dos satélites estão em perfeita sincronia com o relógio do receptor, o que não acontece na realidade devido à baixa precisão dos

relógios utilizados no segmento de usuários (comparado com os relógios atômicos de Cs ou Rb dos satélites) [1].

Incluindo o erro de precisão dos relógios utilizados nos receptores dos usuários e denominando-o de δt, a Equação 2.1 passa a ser escrita por 2.12 e, com isso, o conjunto de Equações 2.3 apresenta uma incógnita nova, o que exige a observação de um quarto satélite para resolver o sistema (Eq. 2.13) [8]:

$$d = c(t_u + \delta t - t_s) = c\Delta t + c\delta t \tag{2.12}$$

$$\begin{cases} d_1 = \sqrt{(x_u - x_1)^2 + (y_u - y_1)^2 + (z_u - z_1)^2} - c\delta t \\ d_2 = \sqrt{(x_u - x_2)^2 + (y_u - y_2)^2 + (z_u - z_2)^2} - c\delta t \\ d_3 = \sqrt{(x_u - x_3)^2 + (y_u - y_3)^2 + (z_u - z_3)^2} - c\delta t \\ d_4 = \sqrt{(x_u - x_4)^2 + (y_u - y_4)^2 + (z_u - z_4)^2} - c\delta t \end{cases} \tag{2.13}$$

Posteriormente, essas equações não-lineares podem ser resolvidas utilizando soluções fechadas, métodos interativos a partir da linearização ou filtro de Kalman [8].

2.3.4 – Linearização e Iteração das Equações das Distâncias

É utilizada a técnica de linearização do sistema de Equações 2.13 a fim de reescrevê-las de forma a possibilitar aplicação de métodos computacionais para sua resolução. Considerando a equação da pseudodistância de um satélite *i* ao usuário, dado em 2.14, e a posição e o erro do relógio sendo descritos por uma parcela aproximada (x_u', y_u', z_u', $\delta t'$) e pelos respectivos erros (Δx, Δy, Δz, Δt) em 2.15, tem-se:

$$\begin{aligned} d_i &= \sqrt{(x_i - x_u)^2 + (y_i - y_u)^2 + (z_i - z_u)^2} - c\delta t \\ &= f(x_u, y_u, z_u, \delta t) \end{aligned} \tag{2.14}$$

$$\begin{aligned} x_u &= x_u' + \Delta x \\ y_u &= y_u' + \Delta y \\ z_u &= z_u' + \Delta z \\ \delta t &= \delta t' + \Delta t \end{aligned} \tag{2.15}$$

Dessa forma, a equação de uma pseudodistância aproximada é dada por:

$$d_i' = \sqrt{(x_i - x_u')^2 + (y_i - y_u')^2 + (z_i - z_u')^2} - c\delta t' = f(x_u', y_u', z_u', \delta t') \tag{2.16}$$

Reescrevendo 2.14 com 2.15:

$$f(x_u, y_u, z_u, \delta t) = f(x_u' + \Delta x, y_u' + \Delta y, z_u' + \Delta z, \delta t' + \Delta t) \tag{2.17}$$

Expandindo 2.17 por séries de Taylor até as derivadas parciais de primeira ordem para eliminar os termos não-lineares, tem-se:

$$\begin{aligned} & f(x_u' + \Delta x, y_u' + \Delta y, z_u' + \Delta z, \delta t' + \Delta t) \\ &= f(x_u', y_u', z_u', \delta t') + \frac{\partial f(x_u', y_u', z_u', \delta t')}{\partial x_u'}\Delta x + \frac{\partial f(x_u', y_u', z_u', \delta t')}{\partial y_u'}\Delta y \\ &+ \frac{\partial f(x_u', y_u', z_u', \delta t')}{\partial z_u'}\Delta z + \frac{\partial f(x_u', y_u', z_u', \delta t')}{\partial \delta t'}\Delta t \end{aligned} \tag{2.18}$$

Calculando as derivadas parciais:

$$\begin{aligned} \frac{\partial f(x_u', y_u', z_u', \delta t')}{\partial x_u'} &= -\frac{x_i - x_u'}{r_i} \\ \frac{\partial f(x_u', y_u', z_u', \delta t')}{\partial y_u'} &= -\frac{y_i - y_u'}{r_i} \\ \frac{\partial f(x_u', y_u', z_u', \delta t')}{\partial z_u'} &= -\frac{z_i - z_u'}{r_i} \\ \frac{\partial f(x_u', y_u', z_u', \delta t')}{\partial \delta t'} &= c \end{aligned} \tag{2.19}$$

onde

$$r_i = \sqrt{(x_i - x_u')^2 + (y_i - y_u')^2 + (z_i - z_u')^2}$$

Substituindo 2.13 e 2.16 em 2.15:

$$d_i = d_i{}' - \frac{x_i - x_u'}{r_i{}'}\Delta x - \frac{y_i - y_u'}{r_i{}'}\Delta y - \frac{z_i - z_u'}{r_i{}'}\Delta z + c\Delta t \tag{2.20}$$

Assim, 2.20 corresponde à Equação 2.14 linearizada. Renomeando os termos da Equação 2.14:

$$\Delta d = d_i' - d_i$$
$$a_{xi} = \frac{x_i - x_u'}{r_i}$$
$$a_{yi} = \frac{y_i - y_u'}{r_i}$$
$$a_{zi} = \frac{z_i - z_u'}{r_i}$$

Substituindo os termos renomeados em 2.20 e escrevendo separadamente para cada satélite, tem-se o sistema (2.21) de incógnitas Δx, Δy, $\Delta \boldsymbol{z}$, Δt:

$$\begin{cases} \Delta d_1 = a_{x1}\Delta x + a_{y1}\Delta y + a_{z1}\Delta z + c\Delta t \\ \Delta d_2 = a_{x2}\Delta x + a_{y2}\Delta y + a_{z2}\Delta z + c\Delta t \\ \Delta d_3 = a_{x3}\Delta x + a_{y3}\Delta y + a_{z3}\Delta z + c\Delta t \\ \Delta d_4 = a_{x4}\Delta x + a_{y4}\Delta y + a_{z4}\Delta z + c\Delta t \end{cases} \quad (2.21)$$

Rearranjando essas equações para o formato matricial, tem-se:

$$\Delta \boldsymbol{d} = \begin{bmatrix} \Delta d_1 \\ \Delta d_2 \\ \Delta d_3 \\ \Delta d_4 \end{bmatrix} \quad \boldsymbol{H} = \begin{bmatrix} a_{x1} & a_{y1} & a_{y1} & 1 \\ a_{x2} & a_{y2} & a_{z2} & 1 \\ a_{x3} & a_{y3} & a_{z3} & 1 \\ a_{x4} & a_{y4} & a_{z4} & 1 \end{bmatrix} \quad \Delta \boldsymbol{x} = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta t \end{bmatrix} \quad (2.22)$$

$$\Delta \boldsymbol{d} = \boldsymbol{H}\Delta \boldsymbol{x}$$

O sistema 2.22 apresenta como solução a equação 2.23 abaixo:

$$\Delta \boldsymbol{x} = \boldsymbol{H^{-1}}\Delta \boldsymbol{d} \quad (2.23)$$

Uma vez calculados os valores de $\Delta \boldsymbol{x}$, é obtido o posicionamento do receptor através de 2.15. Caso os valores de $\Delta \boldsymbol{x}$ ultrapasse os requisitos de precisão do posicionamento em que este método é utilizado, os cálculos são refeitos de forma iterativa atribuindo os valores de (x_u, y_u, z_u) calculados para determinar d_i' a ser utilizado na próxima iteração.

2.3.5 – Tipos de posicionamento

A determinação das pseudodistâncias referentes a cada satélite utilizado no posicionamento é a principal função dos receptores no sistema GPS. Essas pseudodistâncias e as posições determinadas de cada satélite a partir das informações contidas nas efemérides são fatores que influenciam diretamente a precisão do posicionamento do receptor.

Existem dois tipos de posicionamento que estão presentes nos receptores GPS denominados posicionamento por código e pela fase da portadora. Esses tipos de posicionamento diferem-se na forma como determinam as pseudodistâncias. Existem tanto receptores que apresentam apenas um dos tipos de posicionamento quanto receptores que utilizam os dois métodos de forma complementar, a fim de reduzir efeitos degradantes do meio e, consequentemente, aumentar a precisão da observação [1], [6].

O posicionamento por código utiliza o código C/A (ou código P(Y), no caso de usuários autorizados) para determinar as pseudodistâncias através da análise do sinal recebido com o sinal gerado internamente no receptor, determinando o atraso sofrido pela propagação do sinal transmitido. A Figura 2.9 [8] demonstra essa situação.

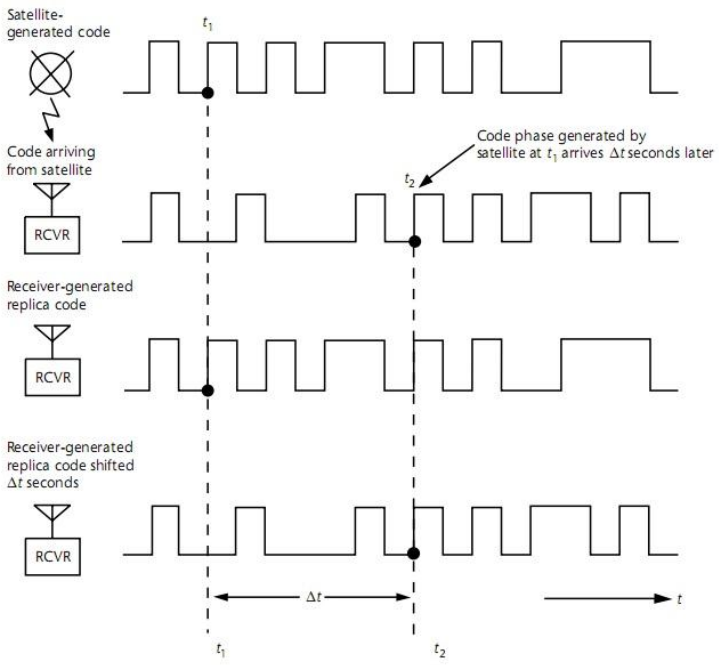


Figura 2.9 - Posicionamento por código [8].

Na Fig. 2.9, o sinal modulado referente ao instante t_1 é recebido pelo receptor apenas no instante t_2 ($t_1 < t_2$). Considerando que os relógios do satélite e receptor estão sincronizados, o sinal gerado internamente é deslocado no tempo até que a máxima correlação entre os sinais seja alcançada. Com isso, calcula-se o deslocamento Δt efetuado determinando a pseudodistância ao satélite.

Enquanto isso, o posicionamento pela fase da portadora corresponde à determinação das pseudodistâncias medindo o atraso de fase sofrido durante a propagação do sinal desde a antena do satélite até a antena do receptor. O posicionamento pela fase da portadora apresenta melhores precisões do que o posicionamento por código, embora as fases das portadoras sejam mais sensíveis a perturbações atmosféricas, exigindo aplicação de modelos que descrevem esses fenômenos para que seja possível minimizá-los [1].

Para determinar esse atraso de fase, é comparada a fase da portadora do sinal recebido com a fase da portadora local e é descrito por duas parcelas, a primeira denominada *ambiguidade de ciclos* sendo um número inteiro *N* de ciclos e a segunda referente à fração ϕ de um ciclo. Considerando um ciclo como sendo o comprimento de onda da portadora λ (λ_1 = 19,05 cm e λ_2 = 24,45 cm), tem-se a pseudodistância *d* entre o receptor e o satélite [1], [6].

$$d = \lambda N + \lambda\phi \tag{2.24}$$

Entretanto, para que seja viável a utilização do posicionamento pela fase da portadora, deve-se resolver a ambiguidade do número inteiro de ciclos com uma boa precisão. O que é considerado um problema, já que a distância entre o satélite e receptor é bem grande (maior que 10 km) e há possibilidade de perda de ciclos e multicaminhamento dos sinais até o receptor [1], [6].

3 - Simulação

A simulação desenvolvida neste trabalho consiste na aplicação dos principais conceitos utilizados pelo sistema GPS real em um cenário simplificado. Foi implementada uma aplicação em linguagem ANSI C para sistemas operacionais Microsoft Windows através do ambiente de desenvolvimento de código aberto *Code::Blocks* versão 10.05 [17]. Além de bibliotecas padrões da linguagem C, foi utilizado também o OpenGL, API ("*application programming interface*") multiplataforma para implementação de aplicações com gráficos 2D e 3D interativos largamente utilizado na Computação Gráfica devido ao seu alto desempenho, estabilidade e escalabilidade [18].

A aplicação denominada *GPS Constellation* ilustra visualmente o planeta Terra e o sistema GPS através dos segmentos espacial e de usuário (constelação de satélites e receptor do usuário) e demonstra seu funcionamento através de informações dos satélites utilizados nos cálculos e da posição do usuário em tempo-real. Resume-se em duas telas principais denominadas "*Space*" e "*Info*" com as seguintes características:

- Tela ***Space***: tela de gráficos 3D que apresenta o planeta Terra e 24 satélites dispostos em 6 órbitas conforme especificados no início da implantação do sistema GPS real. Os satélites apresentam movimento orbital e, simultaneamente, as órbitas movem-se horizontalmente. É apresentado também um usuário do sistema GPS através de um ponto branco movimentando-se próximo à superfície da Terra.
- Tela ***Info***: tela de gráficos 2D que disponibiliza as informações atuais do sistema (posição dos satélites utilizados no posicionamento, posição real e calculada do usuário, entre outras informações) e as opções de interatividade com o usuário da simulação.

A Fig. 3.1 apresenta a janela do simulador *GPS Constellation* com suas duas telas.

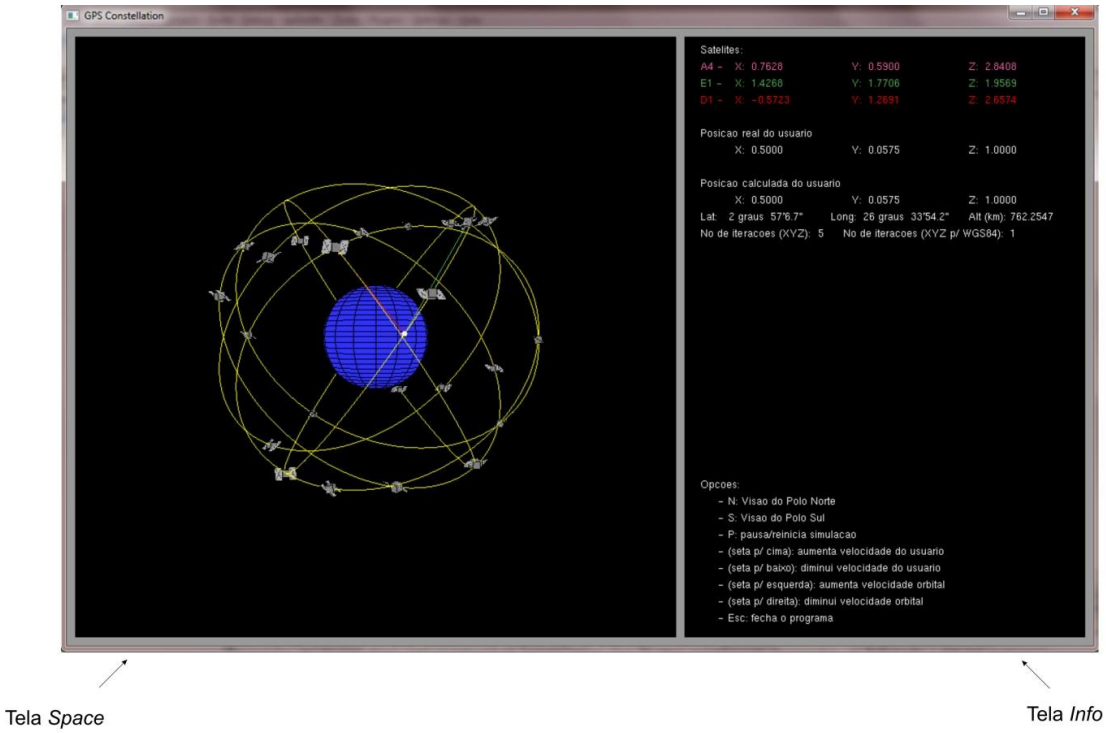


Figura 3.1 - GPS Constellation com suas duas telas.

É feito, também, o registro dos cálculos efetuados durante as iterações para o cálculo do usuário GPS e dos eventos realizados pelo usuário da simulação com a criação de um arquivo *log.txt* no diretório do simulador. Caso seja feito diversas simulações, *log.txt* é reescrito, armazenando sempre informações da última simulação efetuada.

É importante ressaltar que, para a execução do aplicativo, é necessário o arquivo *glut.dll* no diretório do sistema operacional. Seu download e instruções de como instalá-lo corretamente estão disponíveis em [19].

3.1 – Tela Space do GPS Constellation

No ambiente 3D implementado na tela *Space*, a Terra é representada por uma esfera azul, com meridianos e paralelos em preto, fixa para o usuário da simulação enquanto as órbitas apresentam rotação em relação ao eixo da Terra, contrastando com o caso real, onde a Terra apresenta rotação e as órbitas são fixas. Foi escolhido desenvolver a simulação de tal forma para que fosse possível visualizar o usuário em toda a simulação na região terrestre determinada.

O cenário da simulação contém características que procuram manter a fidelidade visual com o caso real. A Terra e as 6 órbitas estão em escala com a situação real, com exceção dos 24 satélites que, por motivos visuais, apresentam tamanho com várias ordens de grandeza a mais do que as dimensões reais. O movimento horizontal das órbitas representa a rotação da Terra e está sincronizado com o movimento orbital dos satélites, preservando o período orbital aproximado dos satélites de 12 horas. Por último, os satélites estão dispostos nas órbitas de forma a garantir uma cobertura o mais homogênea possível em qualquer instante da simulação, com ângulos orbitais iniciais múltiplos de 15º (distando 90º entre satélites adjacentes de uma mesma órbita) conforme mostrado na Fig. 3.2, e sendo eles incrementados uniformemente ao longo da simulação.

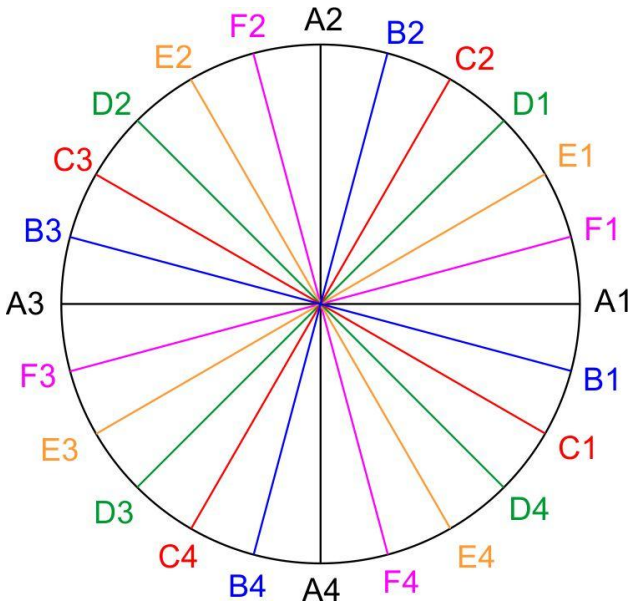


Figura 3.2 - Representação dos ângulos orbitais iniciais dos satélites.

A Fig. 3.2 apresenta as 6 órbitas (de A a F) sobrepostas com seus 4 satélites cada com cores distintas dispostas ao longo de 360º. Ao longo do tempo, os satélites movem-se nas suas órbitas cobrindo a superfície terrestre de forma uniforme, garantindo, assim, a presença de pelo menos 4 satélites visíveis acima do horizonte.

O ambiente 3D da tela *Space* apresenta como sistema de coordenadas o sistema cartesiano, padrão da API OpenGL, com os eixos *x* e *y* na horizontal e vertical, respectivamente, e eixo *z* com direção positiva normal à tela do computador.

A Fig. 3.3 apresenta o ambiente 3D da tela *Space* com representação do sistema de coordenadas adotado para sua programação.

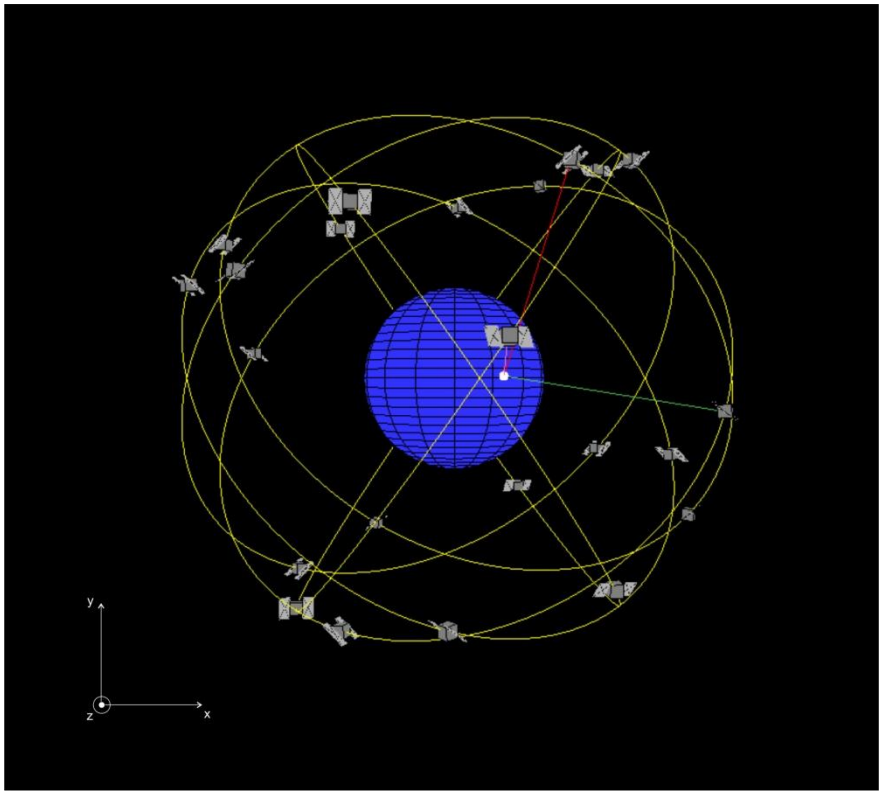


Figura 3.3 - Ambiente 3D com o sistema de coordenadas adotado na programação.

Conforme o usuário e os satélites movem-se, é feito o cálculo do posicionamento através das posições conhecidas dos 3 satélites mais próximos ao usuário no momento do cálculo e estes satélites utilizados são indicados em ambas as janelas. Na janela *Space*, essa indicação é feita através de linhas de visão entre estes satélites e o usuário.

São utilizados apenas 3 satélites para a determinação da posição do usuário pois os cálculos consideram que os relógios dos satélites e do usuário GPS estão perfeitamente sincronizados, possibilitando o cálculo das distâncias com precisão.

3.2 – Tela Info do GPS Constellation

A tela *Info* tem como objetivo informar o usuário da simulação sobre sua situação atual e, na parte inferior, apresenta algumas opções de interface com o usuário. As informações da simulação são:

- Posição dos 3 satélites utilizados no cálculo do posicionamento do usuário: apresenta a posição *(x, y, z)* atual dos satélites, conforme sistema de coordenadas definido na Figura 3.3. Essa posição está normalizada de acordo com o raio médio da Terra adotado na implementação do sistema, ou seja, o raio da esfera referente ao planeta Terra é de 1 unidade (valor real adotado é 6370 km) e o raio das órbitas é de 3,171, aproximadamente (valor real é 20200 km). Cada posição do satélite é acompanhado do seu nome, que corresponde à letra referente a sua órbita (de A a F) e de um número de identificação interna a uma mesma órbita (de 1 a 4). Além disso, as linhas são escritas na mesma cor que as linhas desenhadas entre o usuário e o satélite para auxiliar sua identificação.
- Posição real do usuário: apresenta a posição real (x_u, y_u, z_u) do usuário no momento da simulação. O usuário do sistema GPS apresenta um movimento fixo e quadrático sobre o plano z = 1 no sistema de coordenadas adotado. O usuário move-se ao longo das arestas de um quadrado de lado 1 com centro em (0, 0, 1), conforme Figura 3.4.
- Posição calculada do usuário: apresenta a posição calculada do usuário segundo os conceitos do sistema GPS, resolvendo o sistema de Equações 2.11 através da linearização e método iterativo apresentados em 2.3.4 para $\delta t = 0$. Os valores calculados de ri e das matrizes $\Delta \boldsymbol{d}$, $\boldsymbol{H}$ e $\Delta \boldsymbol{x}$ (Eq. 2.22) são registrados a cada iteração no arquivo *log.txt* para análise posterior. A posição é dada nas coordenadas cartesianas *(x, y, z)* utilizadas na simulação e no formato de latitude, longitude e altitude (ϕ, λ, h) através do método de conversão de coordenadas apresentado em 2.3.2.
- Número de iterações realizadas: disponibiliza ao usuário a quantidade de iterações realizadas para o cálculo dos métodos iterativos (para a determinação do posicionamento do usuário e conversão de coordenadas).

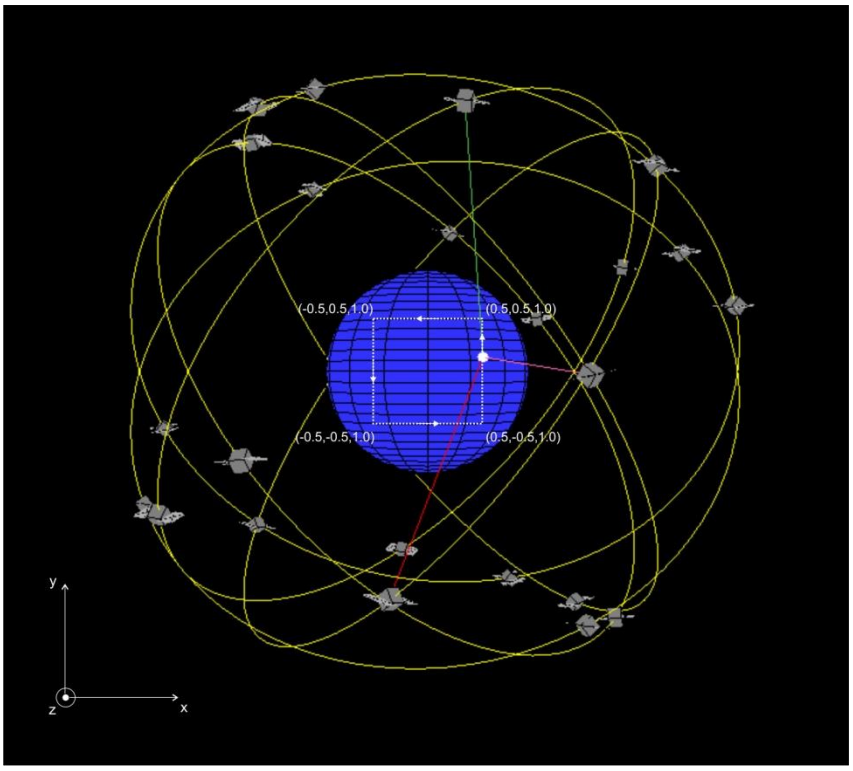


Figura 3.4 - Traçado do movimento do usuário.

Além dessas informações, a tela *Info* informa as opções de interface com o usuário do aplicativo através do teclado:

- Tecla E: habilita visão na linha do Equador (visão padrão);
- Tecla N: habilita visão superior do ambiente 3D, direcionado para o Pólo Norte (Fig. 3.5 a);
- Tecla S: habilita visão inferior do ambiente 3D, direcionado para o Pólo Sul (Fig. 3.5 b);
- Tecla P: pausa/reinicia simulação;
- Tecla ↑: aumenta a velocidade do usuário do sistema GPS;
- Tecla ↓: diminui a velocidade do usuário do sistema GPS;
- Tecla →: aumenta a velocidade de rotação das órbitas e dos satélites nas órbitas, para manter o período orbital constante com a referência de tempo na simulação (1 rotação completa das órbitas = 24 h);
- Tecla ←: diminui a velocidade de rotação das órbitas e dos satélites;
- Tecla Esc: finaliza a simulação e fecha o aplicativo.

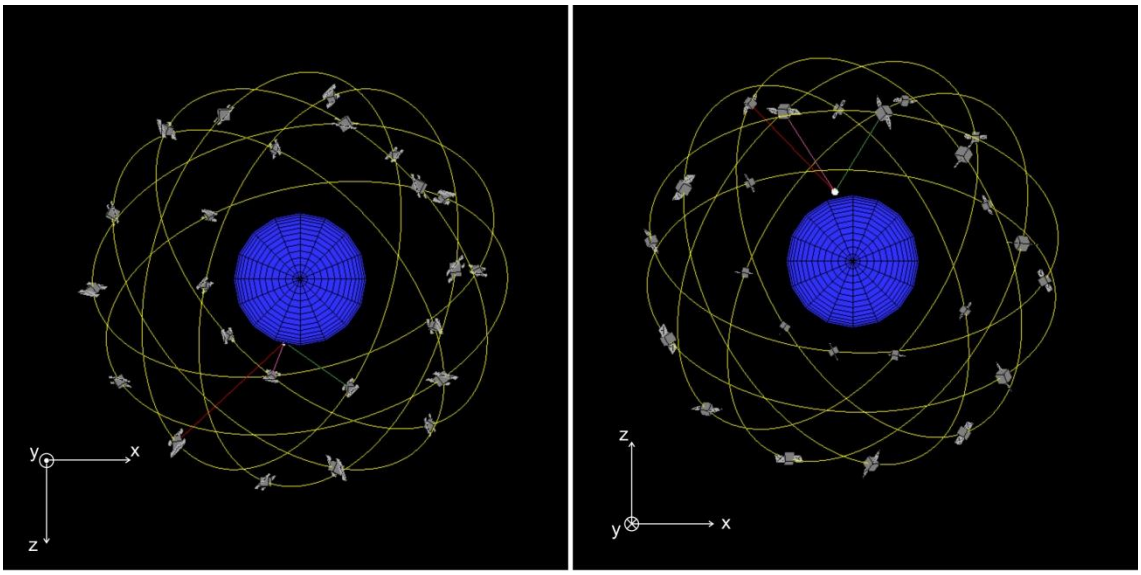


Figura 3.5 - Opções de visão com as coordenadas. a) Visão superior do Pólo Norte. b) Visão inferior do Pólo Sul.

As opções de aumento/diminuição da velocidade das órbitas e satélites podem ser compreendidas também como a escala do tempo real com o tempo da simulação, sendo que, quanto menor a velocidade escolhida pelo usuário, mais próximo o movimento da constelação do simulador ficará da constelação real.

3.3 – Implementação

O desenvolvimento da simulação é dividido nos arquivos *main.cpp* e *calculos.cpp* que contêm os códigos implementados, além de um arquivo de cabeçalho das funções *calculos.h*.

O arquivo *calculos.cpp* (pseudocódigo disponível no Apêndice A) apresenta a implementação dos cálculos efetuados durante toda a simulação. A função *calcClosestSatellites()* define as posições dos 3 satélites mais próximos do usuário no momento do posicionamento, as quais são utilizadas na função *calcUserGpsPosition()* que contém o método iterativo de 2.3.4 implementado com precisão de 10^{-5} para valores de $\Delta\mathbf{x}$. A Equação 2.22 com δt = 0 é resolvida utilizando o método da Eliminação de Gauss (ou método do Escalonamento) para sistemas lineares de ordem 3 implementado na função *gaussElim()* que é chamada a cada iteração no cálculo do

posicionamento. Nesta função, é feito o armazenamento das variáveis utilizadas no posicionamento do usuário no arquivo *log.txt*, possibilitando análises posteriores do método iterativo utilizado. Posteriormente, é feita a conversão para as coordenadas *(ϕ, λ, h)* pela função *convertXyz2Wgs()*, que implementa o método de 2.3.2. Latitude e longitude são coordenadas geralmente dadas em graus, minutos e segundos, sendo, portanto, convertidas para esse formato pela função *convertReal2DegMinSec()*.

O arquivo *main.cpp* (pseudocódigo no Apêndice B) concentra o desenvolvimento da interface gráfica das janelas *Space* e *Info* e seus conteúdos, definindo o formato dos objetos do ambiente 3D, na função space_display(), e a forma de apresentação das informações da simulação, em info_display(). Os cálculos da simulação do sistema GPS são chamados através da função *idle()* a, aproximadamente, cada 100 ms de execução do aplicativo. A interface com o usuário é definida pelas funções *keyboard()* e *specialKeys()*, que tratam os eventos acionados a partir do teclado.

4 - Resultados

Para a implementação do simulador, foi definido que os cálculos do posicionamento pelo sistema GPS fossem executados em um intervalo aproximado de 100 ms a fim de limitar a carga de processamento para que o aplicativo seja executado corretamente na maioria dos computadores atuais. Para execuções em computadores com pouco poder de processamento, foi verificado, principalmente, uma lentidão nos movimentos dos objetos presentes na tela *Space*.

Foi verificado que, para esse intervalo entre cálculos consecutivos de posicionamento, os resultados apresentam uma sincronia com os movimentos do ambiente 3D bastante tolerável.

Os cálculos da posição *(x, y, z)* do usuário GPS determinam a posição correta do usuário na maior parte do tempo em que a simulação é executada. Entretanto, é observado através da análise dos valores registrados em *log.txt* que, para algumas posições dos satélites utilizados no posicionamento, o método 2.3.4 não converge, ou seja, $\Delta\boldsymbol{x}$ calculado a cada iteração apresenta valores absolutos cada vez maiores até que as variáveis utilizadas não possam mais representá-los. Veja Apêndice C. Apesar destes momentos peculiares, o método utilizado calcula o valor correto da posição do usuário GPS em poucas iterações (em até 8 iterações, na maioria das situações analisadas), o que garante sua eficiência.

Por outro lado, a conversão da posição calculada para as coordenadas (ϕ, λ, h), segundo o método demonstrado em 2.3.2, apresenta valores corretos para o padrão de movimento adotado para o usuário GPS na simulação em todos os casos de testes realizados, já que corresponde à faixa de valores de altitude que apresentam erros desprezíveis e latitudes distantes de ±90º.

São apresentados e discutidos alguns casos a fim de demonstrar o correto funcionamento do simulador.

4.1 – Movimentos com *x* e *z* constantes

Os intervalos em que o usuário está nas arestas verticais (x_u = ± 0,5 e z_u = 1) do quadrado referente ao seu movimento (Fig.3.4) correspondem a localizações de longitude λ constante e pode ser calculada por:

$$\lambda = \tan^{-1}\frac{x_u}{z_u} = \tan^{-1}\frac{\pm 0{,}5}{1} = \pm 0{,}463648\, rads = \pm 26{,}5650° = \pm 26° 33' 54{,}2"$$

No caso especial da Fig. 4.1 onde o usuário está na posição inicial em *(0,5; 0,0; 1,0)*, são utilizados os satélites F3, D1 e A3 no momento as simulação e o seu posicionamento *(x, y, z)* calculado pelos satélites estão iguais aos valores reais, sendo necessários 5 iterações para o cálculo com a precisão de 10^{-5}.

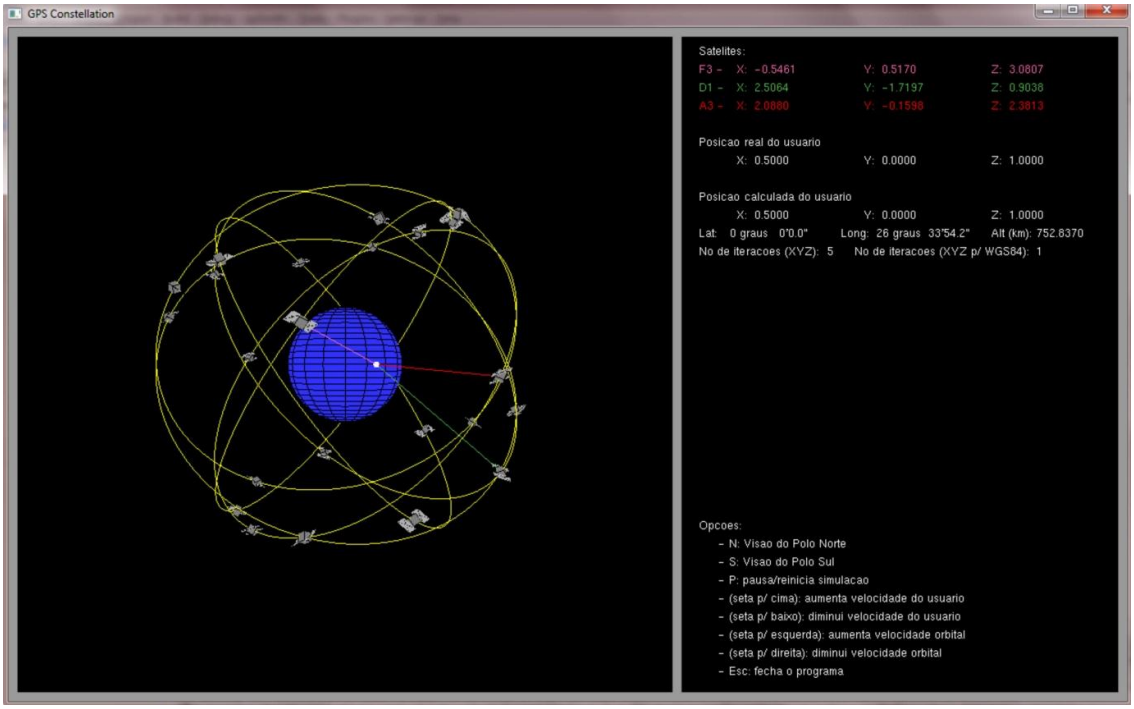


Figura 4.1 - Simulação na visão do Equador com usuário em (0,5; 0,0; 1.0).

Pode-se confirmar que a latitude ϕ = 0°0'0" está no plano do Equador. Para a altitude *h*, pode-se confirmar o valor calculado no simulador tanto pelas Equações 2.5 e 2.6 quanto utilizar a visão pelo Pólo Norte para calculá-la a partir de conceitos geométricos fundamentais, já que, no plano do Equador, a Terra corresponde a uma circunferência de raio *a* (Fig. 4.2).

$$a + h = \sqrt{(z_u * a)^2 + (x_u * a)^2} \;\leftrightarrow\; h = a\sqrt{1{,}25} - a = 752{,}837\, km$$

A altitude calculada corresponde a valores elevados se comparado com as altitudes na maioria das aplicações do sistema GPS. Entretanto, é esperada essa divergência, pois os valores (x, y, z) do simulador estão normalizados em relação ao raio médio da Terra adotado (6370 km) e o movimento do usuário é mantido sobre o plano z = 1.

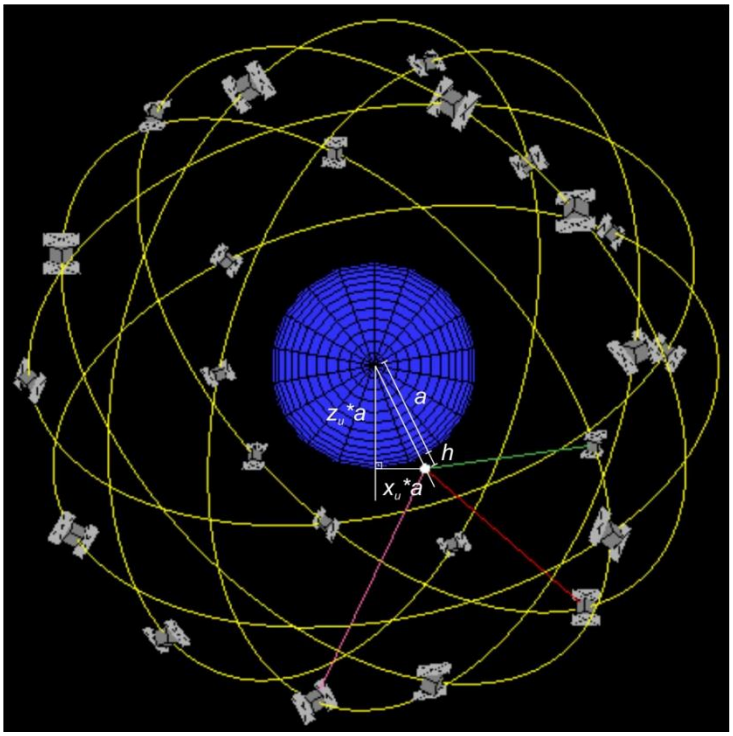


Figura 4.2 - Visão do Pólo Norte para cálculo de h com usuário em (0,5; 0,0; 1,0).

Os pontos médios destas arestas (±0,5; 0,0; 1,0) apresentam as menores altitudes alcançadas pelo usuário nestes intervalos e apresentam crescimento constante até alcançarem seus valores máximos nos vértices do quadrado, em (±0,5; ±0,5; 1).

No caso da Fig. 4.3 onde o usuário está no vértice (0,5; 0,5; 1,0), são usados os satélites D1, F4 e E4 no momento da simulação. A posição *(x, y, z)* calculada do usuário é igual à sua posição real. A longitude λ permanece constante (λ = 26°33’54,2’’) enquanto que a latitude ϕ e a altitude *h* apresentam valores máximos dentro desses intervalos.

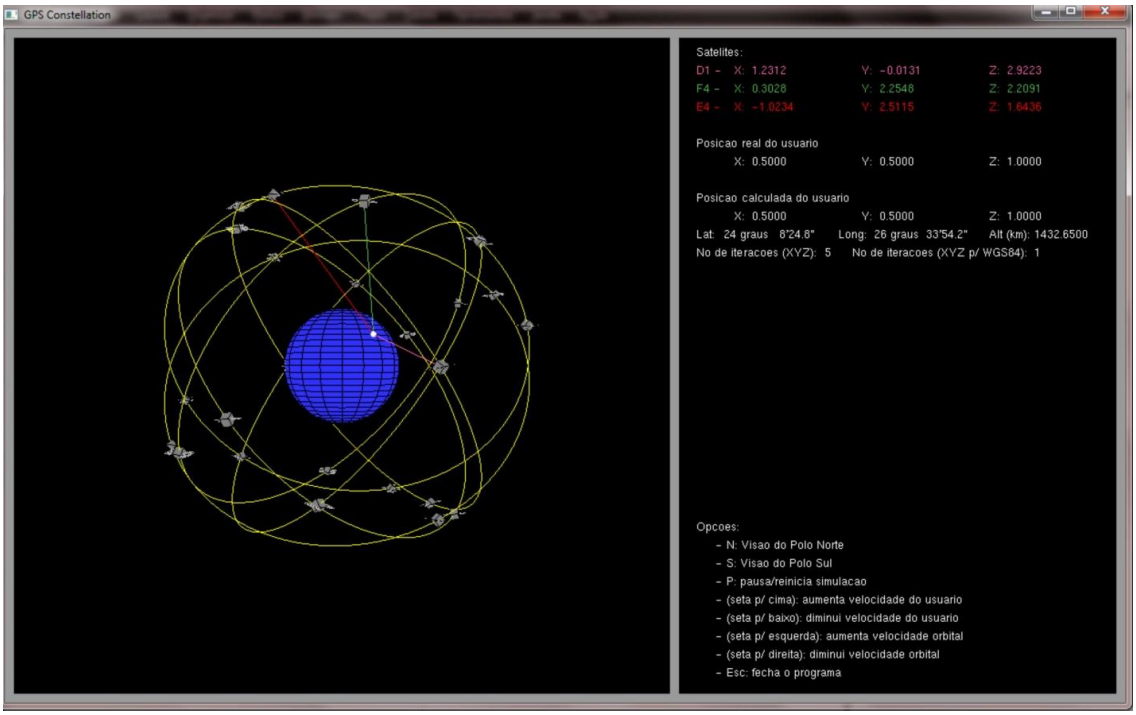


Figura 4.3 - Simulação na visão do Equador com usuário em (0,5; 0,5; 1,0).

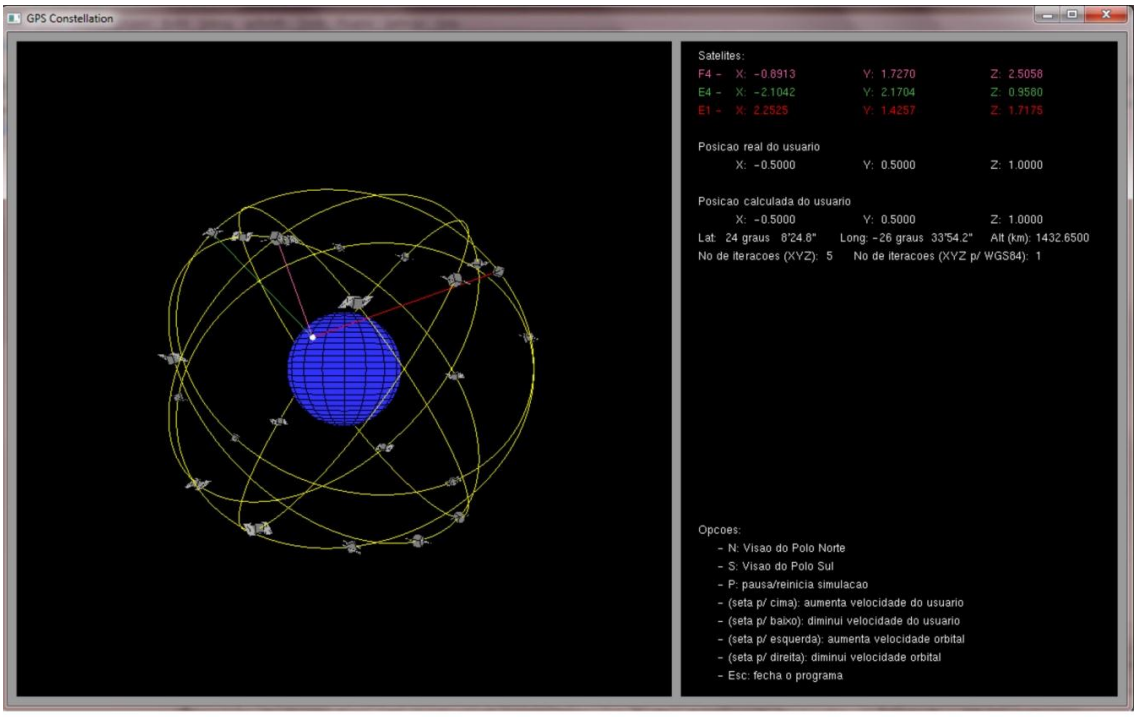
Para o cálculo de *h* utilizando 2.5 e 2.6, tem-se:

$$N = \frac{a}{\sqrt{1 - e^2 \sin^2 \phi}} = 6381710{,}753$$

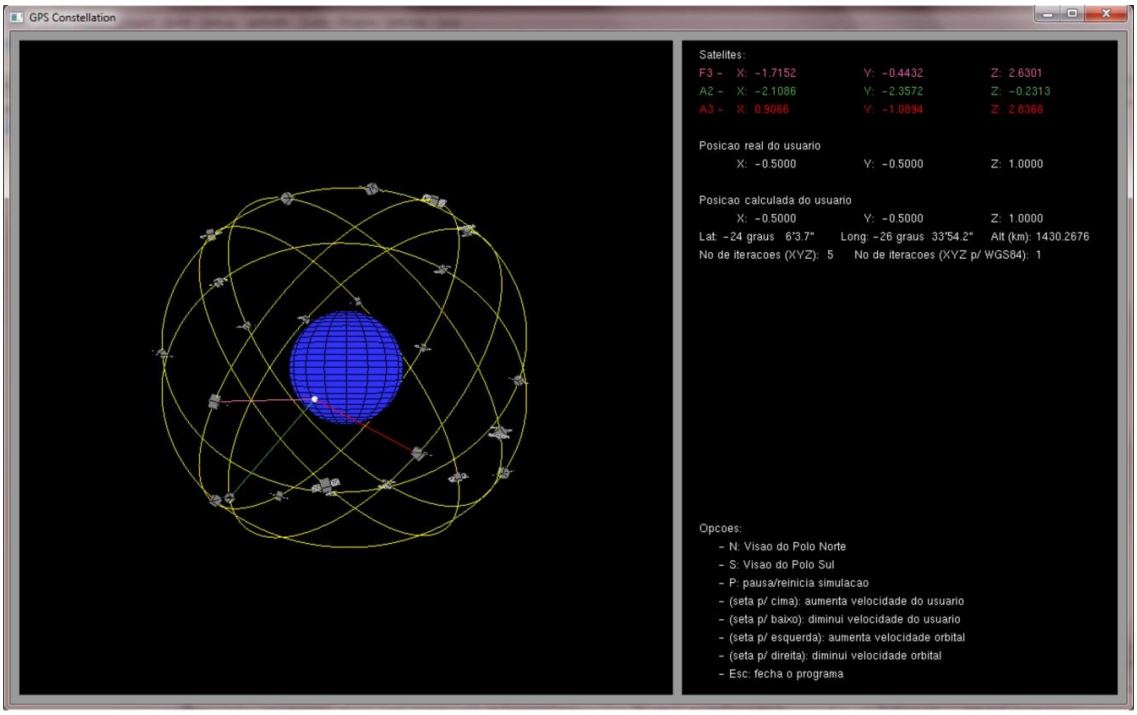
$$h = \frac{p}{\cos \phi} - N = \frac{7130973{,}951}{0{,}9125} - 6381710{,}753 = 1432{,}650\ km$$

Novamente, a altitude calculada é alta para situações reais, mas é justificada devido à normalização utilizada na implementação do simulador.

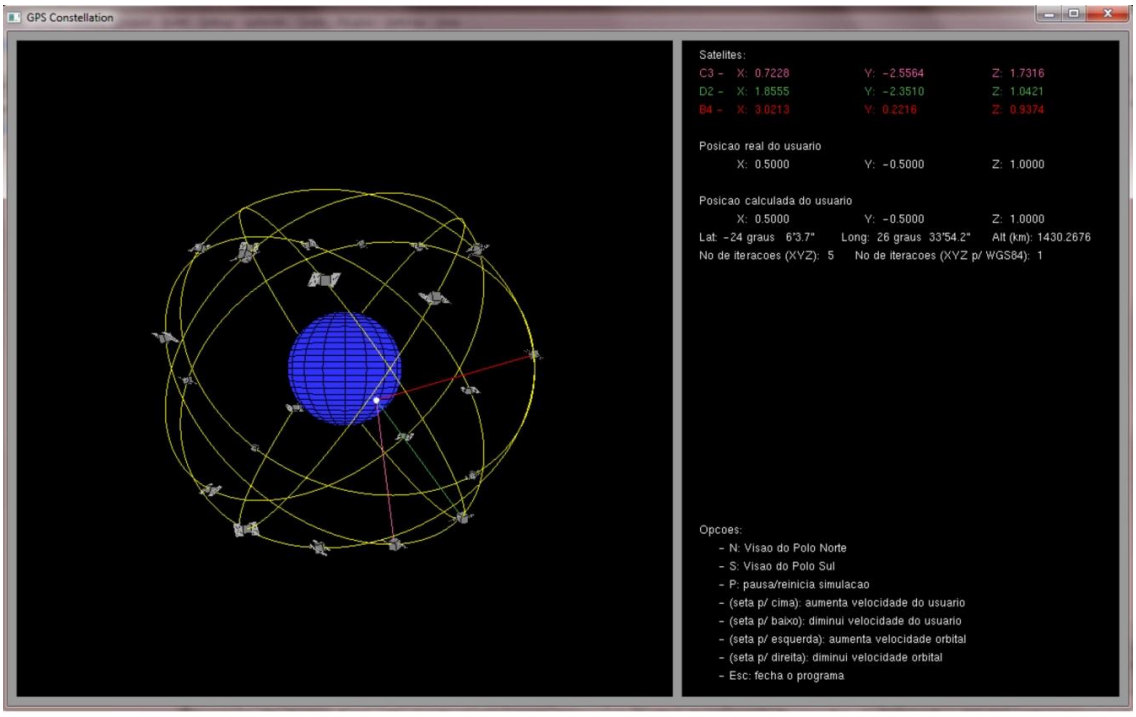
As latitudes e altitudes repetem-se, com valores aproximados, nas arestas restantes (-0,5; 0,5; 1,0), (-0,5; -0,5; 1,0) e (0,5; -0,5; 1,0), como se pode ver nas Figuras 4.4, apresentando valores negativos para as latitudes nos casos onde y = -0,5 (Fig. 4.4-b e 4.4-c).


GPS Constellation
Satelites:
F4 - X: -0.8913 Y: 1.7270 Z: 2.5058
E4 - X: -2.1042 Y: 2.1704 Z: 0.9580
E1 - X: 2.2525 Y: 1.4257 Z: 1.7175
Posicao real do usuario
X: -0.5000 Y: 0.5000 Z: 1.0000
Posicao calculada do usuario
X: -0.5000 Y: 0.5000 Z: 1.0000
Lat: 24 graus 8'24.8" Long: -26 graus 33'54.2" Alt (km): 1432.6500
No de iteracoes (XYZ): 5 No de iteracoes (XYZ p/ WGS84): 1
Opcoes:
- N: Visao do Polo Norte
- S: Visao do Polo Sul
- P: pausa/reinicia simulacao
- (seta p/ cima): aumenta velocidade do usuario
- (seta p/ baixo): diminui velocidade do usuario
- (seta p/ esquerda): aumenta velocidade orbital
- (seta p/ direita): diminui velocidade orbital
- Esc: fecha o programa

a)


GPS Constellation
Satelites:
F3 - X: -1.7152 Y: -0.4432 Z: 2.6301
A2 - X: -2.1086 Y: -2.3572 Z: -0.2313
A3 - X: 0.9066 Y: +1.0894 Z: 2.8368
Posicao real do usuario
X: -0.5000 Y: -0.5000 Z: 1.0000
Posicao calculada do usuario
X: -0.5000 Y: -0.5000 Z: 1.0000
Lat: -24 graus 6'3.7" Long: -26 graus 33'54.2" Alt (km): 1430.2676
No de iteracoes (XYZ): 5 No de iteracoes (XYZ p/ WGS84): 1
Opcoes:
- N: Visao do Polo Norte
- S: Visao do Polo Sul
- P: pausa/reinicia simulacao
- (seta p/ cima): aumenta velocidade do usuario
- (seta p/ baixo): diminui velocidade do usuario
- (seta p/ esquerda): aumenta velocidade orbital
- (seta p/ direita): diminui velocidade orbital
- Esc: fecha o programa

b)



c)

Figura 4.4 - Simulações para as arestas restantes do quadrado referente ao movimento do usuário. a) Posição com longitude negativa e latitude positiva. b) Posição com longitude e latitude negativas. c) Posição com longitude positiva e latitude negativa.

4.2 - Movimentos com *y* e *z* constantes

Os intervalos em que o usuário está nas arestas horizontais (y_u = ± 0,5 e z_u = 1) do quadrado referente ao seu movimento (Fig.3.4) correspondem, teoricamente, a localizações de latitude ϕ constante. Entretanto, devido às grandes variações de altitude entre os máximos, que corresponde aos vértices em (0,5; ±0,5; 1,0), e mínimos, pontos médios das arestas em (0,0; ±0,5; 1,0), os valores das latitudes apresentaram uma pequena variação como se pode ver a seguir.

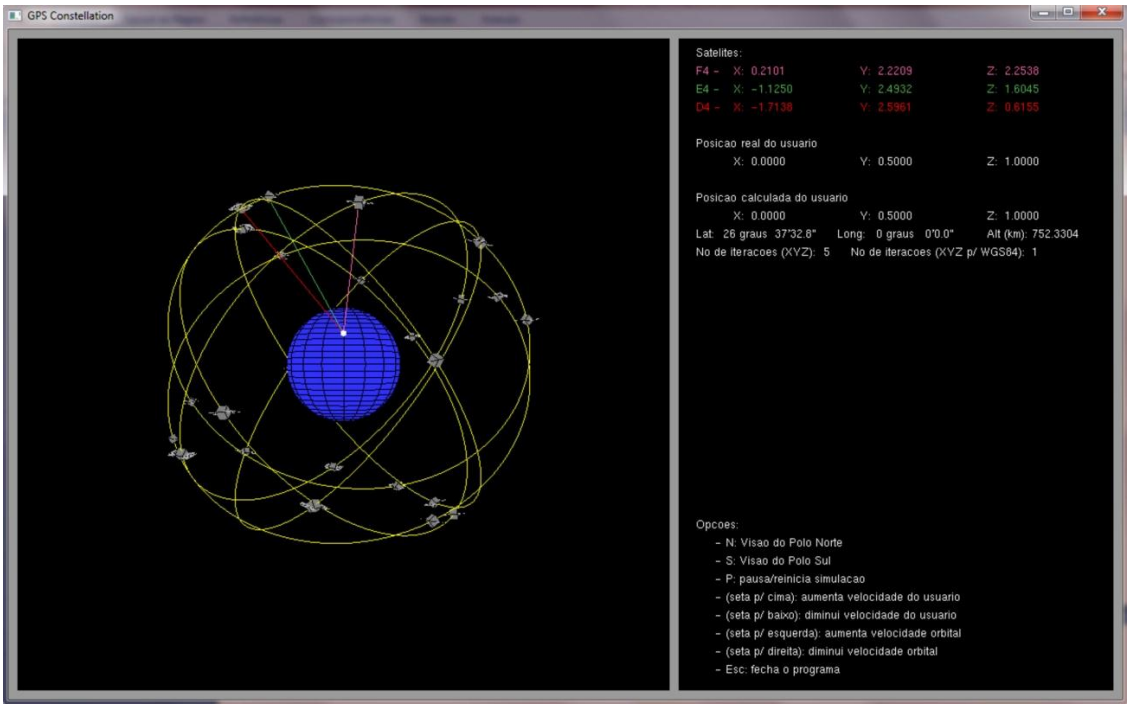


Figura 4.5 - Simulação na visão do Equador com usuário em (0,0; 0,5; 1,0).

No ponto médio da aresta superior (0,0; 0,5; 1,0), Fig. 4.5, o usuário está utilizando os satélites F4, E4 e D4 e está na linha do meridiano de Greenwich, portanto, com longitude $\lambda = 0$. A latitude ϕ é de 26°37'32,8'' (com 1 iteração do método proposto por [16]), sendo essa variação de, aproximadamente, 2°29' quando comparado com os valores dos vértices devido à variação da altitude que, neste caso, é de h = 752,33 km e nos casos anteriores de h = 1432,65 km (Fig. 4.3 e 4.4-a).

No caso da aresta inferior, a situação repete-se com sinal invertido para os valores da latitude ϕ, que corresponde a ϕ = -26°34'13,7'' para altitude de h = 748,90 km, em (0,0; -0,5; 1,0) (Fig. 4.6), e ϕ = -24°6'3,7'' para altitude de h = 1430,27 km (Fig. 4.4-b e 4.4-c).

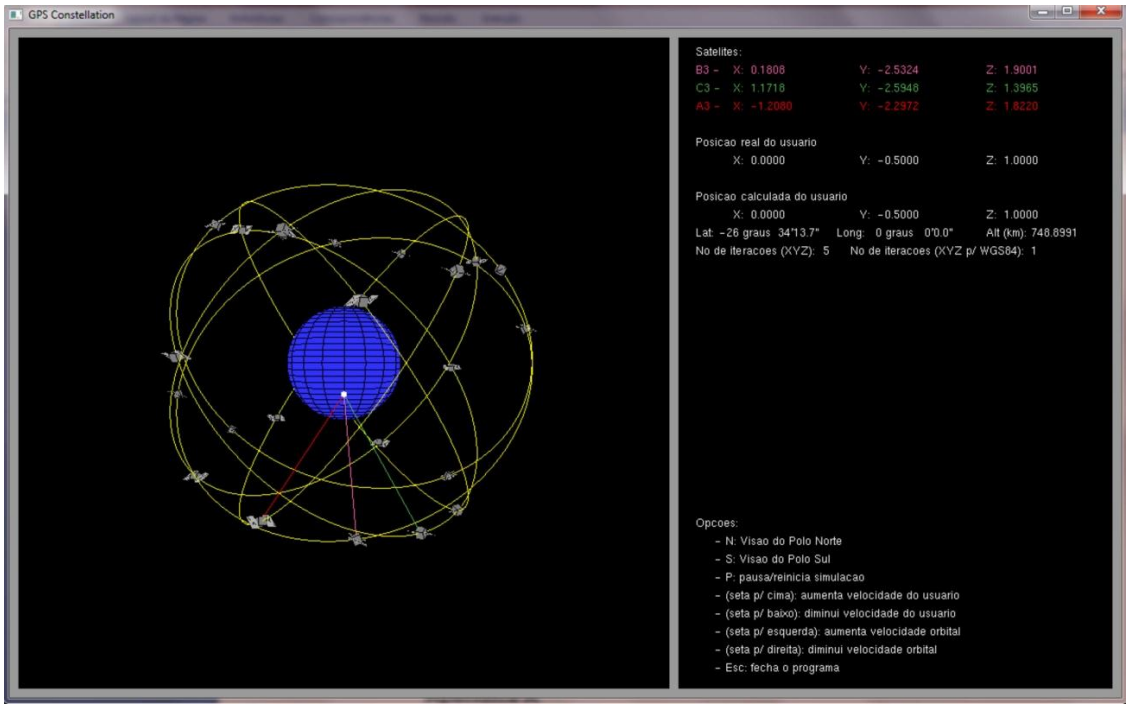


Figura 4.6 - Simulação na visão do Equador com usuário em (0,0; -0,5; 1,0).

A Tabela 4.1 apresenta os valores de algumas posições no movimento do usuário GPS nas coordenadas *(x, y, z)* do simulador e WGS84 calculadas.

Tabela 4.1 - Posições no sistema cartesiano e WGS84

Posição (x, y, z)	**Latitude ϕ**	**Longitude λ**	**Altitude h (km)**
(0,5; 0,0; 1,0)	0° 0’ 0,0’’	26°33’54,2’’	752,84
(0,5; 0,25; 1,0)	12°38’ 5,8’’	26°33’54,2’’	928,79
(0,5; 0,5; 1,0)	24° 8’24,8’’	26°33’54,2’’	1432,65
(0,25; 0,5; 1,0)	25°55’59,0’’	14° 2’10,5’’	928,34
(0,0; 0,5; 1,0)	26°37’32,8’’	0° 0’ 0,0’’	752,33
(-0,25; 0,5; 1,0)	25°55’59,0’’	-14° 2’10,5’’	928,34
(-0,5; 0,5; 1,0)	24° 8’24,8’’	-26°33’54,2’’	1432,65
(-0,5; 0,25; 1,0)	12°38’ 5,8’’	-26°33’54,2’’	928,79
(-0,5; 0,0; 1,0)	0° 0’ 0,0’’	-26°33’54,2’’	752,84
(-0,5; -0,25; 1,0)	-12°37’41,2’’	-26°33’54,2’’	928,59
(-0,5; -0,5; 1,0)	-24° 6’ 3,7’’	-26°33’54,2’’	1430,27
(-0,25; -0,5; 1,0)	-25º52’57,3’’	-14° 2’10,5’’	925,23
(0,0; -0,5; 1,0)	-26°34’13,7’’	0° 0’ 0,0’’	748,90

(0,25; -0,5; 1,0)	-25°52’57,3’’	14° 2’10,5’’	925,23
(0,5; -0,5; 1,0)	-24° 6’ 3,7’’	26°33’54,2’’	1430,27
(0,5; -0,25; 1,0)	-12°37’41,2’’	26°33’54,2’’	928,59

Por fim, podem-se perceber, também, pequenas variações nos valores entre posições correspondentes no hemisfério norte e hemisfério sul. Essas variações devem-se à utilização de valores normalizados com relação ao raio da Terra adotado, o que reduz todos os valores utilizados na simulação e ocasiona maiores erros de representação nas variáveis e aproximações feitas nos cálculos.

5 - Conclusão

O posicionamento geográfico através do sistema GPS consiste, atualmente, em um serviço disponível a todos com a crescente popularização e variedade de dispositivos preparados para utilizá-lo. Com sua aplicação em diferentes finalidades e constantes aprimoramentos efetuados nos diferentes segmentos do sistema, a utilização do sistema GPS difundiu-se na sociedade moderna, cada vez mais interessada em novas tecnologias e possibilidades que são providas por elas.

Com fins inicialmente militares, o sistema GPS está presente tanto no mais sofisticado sistema de navegação aérea quanto nos celulares pessoais utilizados diariamente por milhões de pessoas no mundo. Seu desenvolvimento garantiu que fosse utilizado em qualquer região terrestre e a qualquer momento ininterruptamente. Além disso, com as recentes inovações, sua disponibilidade, confiabilidade e eficiência têm sido aprimoradas permitindo que seja utilizado mesmo em diferentes condições climáticas e com tempo de resposta e precisão cada vez menores.

Por meio de métodos matemáticos apresentados e implementados na simulação desenvolvida, podem-se compreender, de forma simplificada, como os receptores GPS são projetados para determinar seu posicionamento. Desenvolvendo tais métodos em nível de hardware, em contraste com a simulação feita neste trabalho, o desempenho dos cálculos e precisões dos valores manipulados pode ser aprimorado, embora o baixo consumo de energia deva ser levado em consideração no projeto, já que é um requisito importante para aplicações embarcadas e móveis.

O ambiente gráfico desenvolvido na simulação permite demonstrar visualmente o funcionamento do sistema GPS através dos satélites em órbita no planeta Terra. Além disso, corresponde a uma aplicação que permite futuros aprimoramentos a fim de torna-la mais utilizável tanto para fins didáticos quanto comerciais.

Com foco na constelação de satélites atuais, uma sugestão de aprimoramento futuro seria desenvolver uma comunicação com o segmento de controle do sistema GPS de forma a atualizar em tempo real o estado e posições dos satélites em funcionamento no sistema, aumentando, assim, o realismo do simulador e provendo informações úteis sobre o funcionamento do sistema GPS à comunidade de usuários e desenvolvedores de softwares que utilizam o serviço de posicionamento. Tais informações juntamente com o ambiente gráfico podem ser disponibilizadas na

internet, permitindo que qualquer usuário com acesso à rede obtenha informações atualizadas sobre o serviço.

Por outro lado, visando aprimorar a experiência de utilização para posicionamentos por parte do usuário final, poderia ser aplicada uma API gráfica, como a do *Google Maps*, para prover mapas de cidades e estradas, além da opção de visualização por fotos tiradas de satélites, da região onde o usuário se encontra no momento. Para isso, seria necessária a utilização de uma antena receptora GPS conectada com o computador para captar os sinais, processa-los e disponibiliza-los para a aplicação.

Apêndice

Apêndice A

É apresentado neste apêndice o pseudocódigo referente ao arquivo *calculos.cpp*:

```
ESTRUTURA DE DADOS Satellite
  início
    name: cadeia de caracteres;
    orbitalAngle, initialAngle: real;
    orbit: caracter;
  fim

ESTRUTURA DE DADOS ClosestSatellite
  início
    name: cadeia de caracteres;
    x, y, z, distance: real;
    index: inteiro;
  fim

ESTRUTURA DE DADOS User
  início
    x, y, z: real;
    latDegree, latMin, latSec: real;
    lonDegree, lonMin, lonSec: real;
    altitude: real;
  fim

CONSTANTES
  EARTHR = 6370;
  ORBITR = 20200;
  DEGREE2RADIAN = PI/180;
  SINCOSQTT = 10000;

VARIÁVEIS GLOBAIS
  sat: vetor tipo Satellite de tamanho 24;
  closestSat: vetor tipo ClosestSatellite de tamanho 3;
  user, userGps: elementos tipo User;
  sint, cost: vetores reais de tamanho SINCOSQTT;
  delta = 360*DEGREE2RADIAN/SINCOSQTT: real;


//Função que inicia os valores das variáveis tipo User
Função initUser
  início
    user.x = 0.5;
    user.y = 0.0;
    user.z = 1.0;
    user.latDegree = 0;
    user.latMin = 0;
    user.latSec = 0.0;
    user.lonDegree = 0;
    user.lonMin = 0;
    user.lonSec = 0.0;
```

```
    user.altitude = 0.0;

    userGps.x = 0;
    userGps.y = 0;
    userGps.z = 0;
    userGps.latDegree = 0;
    userGps.latMin = 0;
    userGps.latSec = 0.0;
    userGps.lonDegree = 0;
    userGps.lonMin = 0;
    userGps.lonSec = 0.0;
    userGps.altitude = 0.0;
  fim


//Função que inicia os valores dos 24 satélites
Função initSatellites
  variáveis
    i: inteiro;

início
  para i de 1 até 24 faça
  início
    caso i/4
      início
        igual 0:
          sat[i].name[0] = A;
          sat[i].name[1] = '0'+(i%4)+1;
          sat[i].orbit = A;
          sat[i].initialAngle = (PI/2)*(i%4) + (PI/12)*(i/4);
          sat[i].orbitalAngle = (PI/2)*(i%4) + (PI/12)*(i/4);
        igual 1:
          sat[i].name[0] = B;
          sat[i].name[1] = '0'+(i%4)+1;
          sat[i].orbit = B;
          sat[i].initialAngle = (PI/2)*(i%4) + (PI/12)*(i/4) - (PI/6);
          sat[i].orbitalAngle = (PI/2)*(i%4) + (PI/12)*(i/4) - (PI/6);
        igual 2:
          sat[i].name[0] = C;
          sat[i].name[1] = '0'+(i%4)+1;
          sat[i].orbit = C;
          sat[i].initialAngle = (PI/2)*(i%4) + (PI/12)*(i/4) - (PI/3);
          sat[i].orbitalAngle = (PI/2)*(i%4) + (PI/12)*(i/4) - (PI/3);
        igual 3:
          sat[i].name[0] = D;
          sat[i].name[1] = '0'+(i%4)+1;
          sat[i].orbit = D;
          sat[i].initialAngle = (PI/2)*(i%4) + (PI/12)*(i/4);
          sat[i].orbitalAngle = (PI/2)*(i%4) + (PI/12)*(i/4);
        igual 4:
          sat[i].name[0] = E;
          sat[i].name[1] = '0'+(i%4)+1;
          sat[i].orbit = E;
          sat[i].initialAngle = (PI/2)*(i%4) + (PI/12)*(i/4) - (PI/6);
          sat[i].orbitalAngle = (PI/2)*(i%4) + (PI/12)*(i/4) - (PI/6);
        igual 5:
          sat[i].name[0] = F;
          sat[i].name[1] = '0'+(i%4)+1;
          sat[i].orbit = F;
```

```
          sat[i].initialAngle = (PI/2)*(i%4) + (PI/12)*(i/4) - (PI/3);
          sat[i].orbitalAngle = (PI/2)*(i%4) + (PI/12)*(i/4) - (PI/3);
    fim-para
fim


//Função que inicia sint e cost com valores truncados de seno e cosseno
Função initSenCos
  variáveis
    i: inteiro;
início
  para I de 0 até SINCOSQTT
  início
    sint[i] = sin(i*delta);
    cost[i] = cos(i*delta);
  fim-para
fim


//Função responsável pelo movimento quadrático do usuário. Incrementa suas
//posições X e Y, mantendo-o em um plano Z constante
Função updateUserPosition
  parâmetros
    increment: real;
início
  se user.x maior-igual a 0.5 e user.y maior-igual a 0.5 faça
  início
    user.x = user.x - increment;
  fim-se
  senão se user.x menor-igual a -0.5 e user.y maior-igual a 0.5 faça
  início
    user.y = user.y - increment;
  fim-se
  senão se user.x menor-igual a -0.5 e user.y menor-igual a -0.5 faça
  início
    user.x = user.x + increment;
  fim-se
  senão se user.x maior-igual a 0.5 e user.y menor-igual a -0.5 faça
  início
    user.y = user.y + increment;
  fim-se
  senão se user.x maior-igual a 0.5 e user.y maior que -0.5 e user.y menor que
0.5 faça
  início
    user.y = user.y + increment;
    se user.y maior que 0.5 faça user.y = 0.5;
  fim-se
  senão se user.y maior-igual a 0.5 e user.x maior que -0.5 e user.x menor que
0.5 faça
  início
    user.x = user.x - increment;
    se user.x menor que -0.5 faça user.x = -0.5;
  fim-se
  senão se user.x menor-igual a -0.5 e user.y maior que -0.5 e user.y menor
que 0.5 faça
  início
    user.y = user.y - increment;
    se user.y  menor que -0.5 faça user.y = -0.5;
  fim-se
```

```
  senão se user.y menor-igual a -0.5 e user.x maior que -0.5 e user.x menor
que 0.5 faça
  início
    user.x = user.x + increment;
    se user.x maior que 0.5 faça user.x = 0.5;
  fim-se
fim


//Função que retorna o valor X real do usuário
Função getUserX
início
  retorna user.x;
fim


//Função que retorna o valor Y real do usuário
Função getUserY
início
  retorna user.y;
fim


//Função que retorna o valor Z real do usuário
Função getUserZ
início
  retorna user.z;
fim


//Função que retorna o valor X do usuário calculado pelo GPS
Função getUserGpsX
início
  retorna userGps.x;
fim


//Função que retorna o valor Y do usuário calculado pelo GPS
Função getUserGpsY
início
  retorna userGps.y;
fim


//Função que retorna o valor Z do usuário calculado pelo GPS
Função getUserGpsZ
início
  retorna userGps.z;
fim


//Função que retorna os graus da longitude do usuário calculado pelo GPS
Função getUserGpsLonDegree
início
  retorna userGps.lonDegree;
fim


//Função que retorna os minutos da longitude do usuário calculado pelo GPS
```

```
Função getUserGpsLonMin
início
  retorna userGps.lonMin;
fim


//Função que retorna os segundos da longitude do usuário calculado pelo GPS
Função getUserGpsLonSec
início
  retorna userGps.lonSec;
fim


//Função que retorna os graus da latitude do usuário calculado pelo GPS
Função getUserGpsLatDegree
início
  retorna userGps.latDegree;
fim


//Função que retorna os minutos da latitude do usuário calculado pelo GPS
Função getUserGpsLatMin
início
  retorna userGps.latMin;
fim


//Função que retorna os segundos da latitude do usuário calculado pelo GPS
Função getUserGpsLatSec
início
  retorna userGps.latSec;
fim


//Função que retorna a altitude do usuário calculado pelo GPS
Função getUserGpsAltitude
início
  retorna userGps.altitude;
fim


//Função que retorna o valor do seno referente ao parâmetro index
Função getSint
  parâmetros
    index: real;
início
  retorna sint[index];
fim


//Função que retorna o valor do cosseno referente ao parâmetro index
Função getSint
  parâmetros
    index: real;
início
  retorna cost[index];
fim
```

```
//Função que retorna o valor do delta, intervalo entre dois valores
//consecutivos em que são calculados seus senos e cossenos
Função getDelta
início
  retorna delta;
fim


//Função que incrementa os valores do 'orbitalAngle' dos satélites, ângulo
//responsável pelo movimento dos satélites nas suas respectivas órbitas
Função updateOrbitalAngles
  parâmetros
    increment: real;
  variáveis
    i: inteiro;
início
  para i de 1 até 24 faça
    sat[i].orbitalAngle = sat[i].initialAngle + increment;
  fim-para
fim


//Função que recebe um ponto (x,y,z) através do vetor 'pos' e o rotaciona em
//relação ao eixo Y 'angle' radianos
Função roty
  parâmetros
    pos: vetor de tamanho 3 do tipo real;
    angle: real;
  variáveis
    x1, y1, z1: reais;
    index: inteiro;
início
  enquanto angle maior que 2*PI faça
    angle -= 2*M_PI;
  fim-enquanto
  index = angle/delta;
  x1 =  pos[0]*cost[index] + 0 + pos[2]*sint[index] + 0;
  y1 = 0 + pos[1] + 0 + 0;
  z1 = -pos[0]*sint[index] + 0 + pos[2]*cost[index] + 0;
  pos[0] = x1;
  pos[1] = y1;
  pos[2] = z1;
fim


//Função que recebe um ponto (x,y,z) através do vetor 'pos' e o rotaciona em
//relação ao eixo Z 'angle' radianos
Função rotz
  parâmetros
    pos: vetor de tamanho 3 do tipo real;
    angle: real;
  variáveis
    x1, y1, z1: reais;
    index: inteiro;
início
  enquanto angle maior que 2*PI faça
    angle -= 2*PI;
  fim-enquanto
  index = angle/delta;
```

```
  x1 = pos[0]*cost[index] - pos[1]*sint[index] + 0;
  y1 = pos[0]*sint[index] + pos[1]*cost[index] + 0;
  z1 = 0 + 0 + pos[2];
  pos[0] = x1;
  pos[1] = y1;
  pos[2] = z1;
fim


//Função que calcula a posição do satélite 's' no momento 'epoch' da simulação
//e armazena no vetor posXYZ recebido como parâmetro
Função calcSatPosition(double *posXYZ, int s, float epoch)
  parâmetros
    posXYZ: vetor de tamanho 3 do tipo real;
    s: inteiro;
    epoch: real;
  variáveis
    angle: real;
    index: inteiro;
início
  angle = sat[s].orbitalAngle;
  se angle menor que 0 faça
    angle = angle + 2*PI;
  fim-se
  enquanto angle maior que 2*PI faça
    angle = angle - 2*PI;
  fim-enquanto
  index = angle/delta;
  posXYZ[0] =  (ORBITR/EARTHR)*cost[index];
  posXYZ[1] = 0;
  posXYZ[2] = -(ORBITR/EARTHR)*sint[index];

  faça rotz(posXYZ, 55*(PI/180));
  faça roty(posXYZ, epoch*(PI)/180 + (s/4)*(PI)/3);
fim


//Função que calcula a distância entre os valores reais do usuário e um
//ponto (x,y,z) do vetor 'sat' referente a algum satélite. É usada para
//determinar os 3 satélites mais próximos ao usuário
Função calcDistanceUserSat
  parâmetros
    sat: vetor de tamanho 3 do tipo real;
início
  retorna raiz quadrada de ((user.x - sat[0])^2 + (user.y - sat[1])^2 +
(user.z - sat[2])^2);
fim


//Função que calcula a distância entre os valores aproximados do usuário na
//iteração ('x', 'y' e 'z') e o 'sat', um dos 3 satélites mais próximos
Função calcDistanceSat
  parâmetros
    sat: element do tipo ClosestSatellite;
    x, y, z: reais;
início
  retorna raiz quadrada de ((sat.x – x)^2 + (sat.y – y)^2 + (sat.z – z)^2);
fim
```

```
//Função que calcula os 3 satélites mais próximos ao usuário no momento
//'epoch' da simulação
Função calcClosestSatellites
  parâmetros
    epoch: real;
    flaglines: inteiro;
  variáveis
    i: inteiro;
    auxSat: vetor de tamanho 3 do tipo real;
    newDist: real;
início
  flagLines = 1;
  para i de 1 até 24 faça
  início
    faça calcSatPosition(auxSat, i, epoch);
    se i menor que 3 faça
    início
      closestSat[i].x = auxSat[0];
      closestSat[i].y = auxSat[1];
      closestSat[i].z = auxSat[2];
      closestSat[i].distance = calcDistanceUserSat(auxSat);
      closestSat[i].index = i;
      closestSat[i].name = sat[i].name;
    fim-se
    senão faça
    início
      newDist = calcDistanceUserSat(auxSat);
      se newDist menor que closestSat[0].distance faça
      início
        closestSat[0].x = auxSat[0];
        closestSat[0].y = auxSat[1];
        closestSat[0].z = auxSat[2];
        closestSat[0].distance = newDist;
        closestSat[0].index = i;
        closestSat[0].name = sat[i].name;
      fim-se
      senão se newDist menor que closestSat[1].distance faça
      início
        closestSat[1].x = auxSat[0];
        closestSat[1].y = auxSat[1];
        closestSat[1].z = auxSat[2];
        closestSat[1].distance = newDist;
        closestSat[1].index = i;
        closestSat[1].name = sat[i].name;
      fim-se
      senão se newDist menor que closestSat[2].distance faça
      início
        closestSat[2].x = auxSat[0];
        closestSat[2].y = auxSat[1];
        closestSat[2].z = auxSat[2];
        closestSat[2].distance = newDist;
        closestSat[2].index = i;
        closestSat[2].name = sat[i].name;
      fim-se
    fim-senão
  fim-para
fim
```

```
//Função que retorna o valor de uma coordenada 'coord' de 'sat', um dos 3
//satélites mais próximos ao usuário
Função getValueOfClosestSats
  parâmetros
    sat, coord: inteiro;
início
  se coord igual a 0 faça
  início
    retorna closestSat[sat].x;
  fim-se
  senão se coord igual a 1 faça
  início
    retorna closestSat[sat].y;
  fim-se
  senão
  início
    retorna closestSat[sat].z;
  fim-senão
fim


//Função que retorna o nome de 'sat', um dos 3 satélites mais próximos ao
//usuário
Função getNameOfClosestSats
  parâmetros
    sat: inteiro;
início
  retorna closestSat[sat].name;
fim


//Função que resolve o sistema linear Ax = B de ordem 3 pelo método da
//Eliminação de Gauss
Função gaussElim(double A[][3], double B[], double X[])
  parâmetros
    A: matriz de 3 linhas e 3 colunas do tipo real
    B, X: vetores de tamanho 3 do tipo real
  variáveis
    r: real;
início
  //elimina a21
  r = A[1][0]/A[0][0];
  A[1][0] = A[1][0] - A[0][0]*r;
  A[1][1] = A[1][1] - A[0][1]*r;
  A[1][2] = A[1][2] - A[0][2]*r;
  B[1] = B[1] - B[0]*r;

  //elimina a31
  r = A[2][0]/A[0][0];
  A[2][0] = A[2][0] - A[0][0]*r;
  A[2][1] = A[2][1] - A[0][1]*r;
  A[2][2] = A[2][2] - A[0][2]*r;
  B[2] = B[2] - B[0]*r;

  //elimina a32
  r = A[2][1]/A[1][1];
  A[2][1] = A[2][1] - A[1][1]*r;
  A[2][2] = A[2][2] - A[1][2]*r;
```

```
  B[2] = B[2] - B[1]*r;
  X[2] = B[2]/A[2][2];
  X[1] = (B[1] - X[2]*A[1][2])/A[1][1];
  X[0] = (B[0] - X[1]*A[0][1] - X[2]*A[0][2])/A[0][0];
fim


//Função que converte um valor real 'value' para graus, minutos e segundos,
//armazenando nas variáveis da estrutura User
Função convertReal2DegMinSec
  parâmetros
    value: real;
    latitude: inteiro;
  variáveis
    aux: real;
início
  se latitude igual a 0 faça
  início
    userGps.lonDegree = parte inteira de (value);
    aux = (parte fracionária de (value))*60;
    se value menor que 0 faça aux = aux*(-1);
    userGps.lonMin = parte inteira de (aux);
    userGps.lonSec = (parte fracionária de (aux))*60;
  fim-se
  senão
  início
    userGps.latDegree = parte inteira de (value);
    aux = (parte fracionária de (value))*60;
    se value menor que 0 faça aux = aux*(-1);
    userGps.latMin = parte inteira de (aux);
    userGps.latSec = (parte fracionária de (aux))*60;
  fim-se
fim


//Função que converte um ponto no sistema ECEF para WGS84 segundo método de
//Bowring e retorna o valor de iterações utilizadas na conversão
Função convertXyz2Wgs
  variáveis
    count=0: inteiro;
    p, a, b, e, e2, N: reais;
    x, y, z: reais;
    sinU, cosU, tanU0, tanU, fi, tanFi: reais;
    convRad2Deg = 180/M_PI: real;
início
  a = 6378137.0;
  b = 6356752.31;
  x = userGps.z*a;
  y = userGps.x*a;
  z = userGps.y*b;
  e = raiz quadrada(1-(b^2)/(a^2));
  e2 = (a/b)*e;

  se x maior-igual a 0
  início
    faça convertReal2DegMinSec(arctan(y/x)*convRad2Deg, 0);
  fim-se
  senão se x menor que 0 e y menor-igual a 0
  início
```

```
    convertReal2DegMinSec(180.0 + arctan(y/x)*convRad2Deg, 0);
  fim-se
  senão se x menor que 0 e y menor que 0
  início
    convertReal2DegMinSec(-180.0 + arctan(y/x)*convRad2Deg, 0);
  fim-se
  p = raiz quadrada((x)^2 + (y)^2);
  tanU = (z/p)*(a/b);

  faça
  início
    count = count + 1;
    tanU0 = tanU;
    cosU = raiz quadrada(1/(1 + (tanU0)^2));
    sinU = raiz quadrada(1 – cosU^2);
    tanFi = (z + (e2^2)*b*(sinU^3))/(p - (e^2)*a*(cosU^3));
    tanU = (b/a)*tanFi;
  fim-faça-enquanto
  enquanto modulo(tanU0 - tanU) maior que 1e-6;
  fi = arctan(tanFi);
  faça convertReal2DegMinSec(fi*convRad2Deg, 1);

  N = a/(raiz quadrada(1 - (e^2)*(sin(fi))^2));

  se fi diferente de ±90 faça
  início
    userGps.altitude = (p/cos(fi)) - N;
    userGps.altitude = userGps.altitude/1E3;
  fim-se
  senão se fi diferente de 0 faça
  início
    userGps.altitude = (z/sin(fi)) - N + (e^2)*N;
    userGps.altitude = userGps.altitude/1E3;
  fim-se
  retorna count;
fim


//Função que calcula a posição do usuário através do sistema GPS. Utiliza o
método iterativo para o sistema de equações das distâncias dos 3 satélites
mais próximos do usuário. Retorna o valor de iterações utilizadas no cálculo
Função calcUserGpsPosition
  parâmetro
    fp: ponteiro para arquivo;
  variáveis
    i, count=0: inteiros;
    r, d: vetores de tamanho 3 do tipo real;
    A: matriz de 3 linhas e 3 colunas do tipo real
    B, X: vetores de tamanho 3 do tipo real
    xt, yt, zt: reais;

início
  xt = 0;
  yt = 0;
  zt = 0;
  faça
  início
    count = count+1;
    salva count em "log.txt";
```

```
    para i de 1 a 3 faça
    início
      r[i] = calcDistanceSat(closestSat[i], xt, yt, zt);
      d[i] = closestSat[i].distance;
      B[i] = r[i] - d[i];
    fim-para

    salva valores de r[] em "log.txt";
    salva valores de B[] em "log.txt";
    para i de 1 a 3 faça
    início
      A[i][0] = (closestSat[i].x - xt)/r[i];
      A[i][1] = (closestSat[i].y - yt)/r[i];
      A[i][2] = (closestSat[i].z - zt)/r[i];
    fim-para

    salva valores de A[][] em "log.txt";

    faça gaussElim(A, B, X);
    userGps.x = xt + X[0];
    userGps.y = yt + X[1];
    userGps.z = zt + X[2];
    salva valores de X[] em "log.txt";
    xt = userGps.x;
    yt = userGps.y;
    zt = userGps.z;
  fim-faça-enquanto
  enquanto (modulo(X[0]) maior que 0.00001) ou (modulo(X[1]) maior que
0.00001) ou (modulo(X[2]) maior que 0.00001);

  salva posição calculada do usuario em "log.txt";

  retorna count;
fim
```

Apêndice B

É apresentado neste apêndice o pseudocódigo referente ao arquivo *main.cpp*:

```
CONSTANTES
  CIRCLEPOINTS = 1000

VARIÁVEIS OPENGL
  window, info, space: variáveis tipo GLuint;
  spaceWidth = 720, spaceHeight = 720: variáveis tipo GLuint;
  infoWidth = 480, infoHeight = 720: variáveis tipo GLuint;
  sub_width = 720, sub_height = 720: variáveis tipo GLuint;
  GAP = 10: variável tipo GLuint;
  font_style: variável tipo GLvoid;

VARIÁVEIS GLOBAIS
  epoch = 0.0: real;
  speed = 1.0: real;
  flagLines = 0, flagStop = 0: inteiros;
  flagEquaVision = 1, flagNorthVision = 0, flagSouthVision = 0: inteiros;
  count1 = 0, count2 = 0: inteiros;
  last_time = 0: inteiro;
  userIncr = 0.0005: real;
  fp: ponteiro para arquivo;


//Função que escreve a cadeia de caracteres 'format' na janela 'info' na
//posição 'x', 'y'
Função drawstr
  parâmetros
    format: cadeia de caracteres;
    x, y: inteiros;
início
  imprime na tela (format) na posição (x, y) com fonte (font_style);
fim


//Função que deseja uma esfera azul representando o planeta Terra na tela
//'space'
Função drawEarth
início
  define cor (azul);
  desenha esfera em (0,0,0) de raio 1;
  define cor (preta);
  desenha conjunto de círculos na esfera;
fim


//Função que deseja uma esfera branca representando o usuário na tela
//'space'
Função drawUser
início
  define cor(branca);
  desenha esfera em (getUserX(), getUserY(), getUserZ()) de raio 0.05;
fim


//Função que desenha um satélite referente ao índice 'satellite' na tela
//'space'
```

```
Função drawSatellite
  parâmetros
    satellite: inteiro;
início
  desenha satélite em (sat[satellite].x, sat[satellite].y, sat[satellite].z);
fim


//Função que desenha as linhas ligando o usuário aos 3 satélites mais próximos
//dele
Função drawClosestSatLines
  variáveis
    i: inteiro;
início
  para i de 1 a 3 faça
    início
      se i igual a 1 define (cor do satélite 1);
      senão se i igual a 2 define (cor do satélite 2);
      senão se i igual a 3 define (cor do satélite 3);

      desenha linha de (getUserX(), getUserY, getUserZ) até
(getValueOfClosestSats(i, 0), getValueOfClosestSats(i, 1),
getValueOfClosestSats(i, 2));
  fim-para
fim


//Função que define parâmetros do OpenGl para a janela principal
Função main_display
início
  define cor (cinza);
  define outros parâmetros necessários;
fim


//Função que define parâmetros do OpenGl para redimensionamento da janela
//principal
Função main_reshape (int w, int h)
  parâmetros
    w, h: inteiros;
início
  define novos parâmetros e posição para janela principal com nova largura w e
altura h;
  sub_width = w - 3*GAP - infoWidth;
  sub_height = h - 2*GAP;
  define novos parâmetros e posição para janela 'space' com nova largura
sub_width e altura sub_weight;
  define novos parâmetros e posição para janela 'info' com largura infoWidth e
altura sub_weight;
fim


//Função que desenha a janela 'space', com o planeta Terra, 6 órbitas, 24
//satélites, usuário e as linhas até os satélites mais próximos. É chamada
//continuamente para desenhar a posição corrente dos objetos.
Função space_display
  variáveis
    i, orbit, index: inteiros;
    angle: real;
início
  limpa os buffers do OpenGl;
  faça drawEarth();
  faça drawUser();
  para orbit de 1 a 6 faça
  início
    define cor (amarelo)
    desenha círculo com CIRCLEPOINTS pontos e raio (ORBITR/EARTHR);
    para i de 1 a 4 faça
```

```
    início
      faça drawSatellite(orbit*i);
    fim-para
    se flaglines igual a 1 faça
    início
      faça drawClosestSatLines();
    fim
  fim-para
fim


//Função que desenha a janela 'info', com o nome e a posição (x,y,z) dos 3
//satélites mais próximos do usuário, a posição real (x,y,z) do usuário, a
//posição segundo o sistema GPS em (x,y,z) e (latitude, longitude, altitude),
//o número de iterações necessárias em cada método iterativo e, por fim, as
//opções de interface para aumentar e diminuir as velocidades do usuário e dos
//satélites
Função info_display
início
  limpa os buffers do OpenGl;
  define fonte (font_style);
  define cor (branco);
  drawstr(20, 20, "Satelites:");
  define cor (cor do satélite 1);
  drawstr(20, 40, getNameOfClosestSats(0));
  drawstr(60, 40, "X:");
  drawstr(80, 40, getValueOfClosestSats(0, 0));
  drawstr(200, 40, "Y:");
  drawstr(220, 40, getValueOfClosestSats(0, 1));
  drawstr(340, 40, "Z:");
  drawstr(360, 40, getValueOfClosestSats(0, 2));
  define cor (cor do satélite 2);
  drawstr(20, 60, getNameOfClosestSats(1));
  drawstr(60, 60, "X:");
  drawstr(80, 60, getValueOfClosestSats(1, 0));
  drawstr(200, 60, "Y:");
  drawstr(220, 60, getValueOfClosestSats(1, 1));
  drawstr(340, 60, "Z:");
  drawstr(360, 60, getValueOfClosestSats(1, 2));
  define cor (cor do satélite 3);
  drawstr(20, 80, getNameOfClosestSats(2));
  drawstr(60, 80, "X:");
  drawstr(80, 80, getValueOfClosestSats(2, 0));
  drawstr(200, 80, "Y:");
  drawstr(220, 80, getValueOfClosestSats(2, 1));
  drawstr(340, 80, "Z:");
  drawstr(360, 80, getValueOfClosestSats(2, 2));
  define cor (branco);
  drawstr(20, 120, "Posicao real do usuario");
  drawstr(60, 140, "X:");
  drawstr(80, 140, getUserX());
  drawstr(200, 140, "Y:");
  drawstr(220, 140, getUserY());
  drawstr(340, 140, "Z:");
  drawstr(360, 140, getUserZ());
  define cor (branco);
  drawstr(20, 180, "Posicao calculada do usuario");
  drawstr(60, 200, "X:");
  drawstr(80, 200, getUserGpsX());
  drawstr(200, 200, "Y:");
  drawstr(220, 200, getUserGpsY());
  drawstr(340, 200, "Z:");
  drawstr(360, 200, getUserGpsZ());
  drawstr(20, 220, "Lat:");
  drawstr(35,       220,       getUserGpsLatDegree(),       getUserGpsLatMin(),
getUserGpsLatSec());
  drawstr(165, 220, "Long:");
  drawstr(200,       220,       getUserGpsLonDegree(),       getUserGpsLonMin(),
```

```
getUserGpsLonSec());
  drawstr(330, 220, "Alt (km):");
  drawstr(380, 220, getUserGpsAltitude());
  drawstr(20, 240, "No de iteracoes (XYZ): ");
  drawstr(160, 240, count1);
  drawstr(190, 240, "No de iteracoes (XYZ p/ WGS84): ");
  drawstr(390, 240, count2);

  //interface com o usuario
  define cor (branco);
  drawstr(20, 560, "Opcoes:");
  se visão definida é do Equador faça
    drawstr(40, 560, "- N: Visao do Polo Norte");
    drawstr(40, 580, "- S: Visao do Polo Sul");
  fim-se
  se visão definida é do Polo Norte faça
    drawstr(40, 560, "- E: Visao do Equador");
    drawstr(40, 580, "- S: Visao do Polo Sul");
  fim-se
  se visão definida é do Polo Sul faça
    drawstr(40, 560, "- N: Visao do Polo Norte");
    drawstr(40, 580, "- E: Visao do Equador”);
  fim-se
  drawstr(40, 600, "- P: pausa/reinicia simulacao");
  drawstr(40, 620, "- (seta p/ cima): aumenta velocidade do usuario");
  drawstr(40, 640, "- (seta p/ baixo): diminui velocidade do usuario");
  drawstr(40, 660, "- (seta p/ esquerda): aumenta velocidade orbital");
  drawstr(40, 680, "- (seta p/ direita): diminui velocidade orbital");
  drawstr(40, 700, "- Esc: fecha o programa");
fim


//Função que define a interface com o usuário através do teclado. Quando uma
//tecla comum é pressionada, a função é chamada e ‘key’ armazena o valor ASCII
//da tecla
Função keyboard
  parâmetros
    key: inteiro;
início
  caso key
  início
    //define visão do Equador
    igual e ou E:
      flagEquaVision = 1;
      flagNorthPoleVision = 0;
      flagSouthPoleVision = 0;
    //define visão do Polo Norte
    igual n ou N:
      flagEquaVision = 0;
      flagNorthPoleVision = 1;
      flagSouthPoleVision = 0;
    //define visão do Polo Sul
    igual s ou S:
      flagEquaVision = 0;
      flagNorthPoleVision = 0;
      flagSouthPoleVision = 1;
    //pausa simulacao
    igual p ou P:
      se flagStop igual a 1 faça flagStop = 0;
      senão faça flagStop = 1;
      registra evento em “log.txt”;
    //tecla “Esc”
    igual 27:
      finaliza “log.txt”;
      termina programa;
    fim-caso
fim
```

```
//Função que define a interface com o usuário através de teclas especiais do
//teclado. 'key'armazena o valor da tecla especial. Utilizada para receber
//comandos das setas do teclado
Função specialKeys
  parâmetros
    key: inteiro;
início
  caso key
  início
    igual SETA_DIREITA:
          speed = speed + 0.1;
          registra evento em "log.txt";
    igual SETA_ESQUERDA:
          se (speed - 0.1) maior que 0 faça
          inicio
            speed = speed - 0.1;
            registra evento em "log.txt";
          fim-se
    igual SETA_CIMA:
          userIncr = userIncr + 0.00025;
          registra evento em "log.txt";
    igual SETA_BAIXO:
          se (userIncr - 0.00025) maior que 0 faça
          inicio
            userIncr =  userIncr - 0.00025;
            registra evento em "log.txt";
          fim-se
  fim-caso
fim


//Função que é processada continuamente durante o laço principal do OpenGl
Função idle
início
  se (tempo_em_ms – last_time maior que 100) faça
  início
    last_time = tempo_em_ms;
    se flagStop igual a 0 faça
    inicio
      faça updateUserPosition(userIncr);
      faça updateOrbitalAngles(2*epoch*PI/180);
      faça calcClosestSatellites(epoch, flagLines);
      count1 = calcUserGpsPosition();
      count2 = convertXyz2Wgs();
    fim-se
  fim-se
fim


//Função chamada a cada 'value' milissegundos. É utilizada para redefinir a
//velocidade dos satélites nas órbitas, caso seja pressionadas as teclas
//SETA_CIMA ou SETA_BAIXO.
Função timer
  parâmetros
    value: inteiro;
início
  se flagStop igual a 0 faça
    epoch = epoch + (1.0/20.0)*speed;
  fim-se
fim


//Função principal do programa. Define características das janelas e
//parâmetros OpenGl. Inicia os valores do usuário e dos satélites e define as
//funções a serem chamadas ao longo da simulação.
Função main
início
```

```
  escreve na janela secundária(Iniciando simulacao...);
  faça initSenCos();
  inicia arquivo log.txt;
  escreve na janela secundária(Pronto!);

  define parâmetros do OpenGl;
  define tamanho da janela principal;
  define posição da janela principal;
  window = cria janela com titulo("GPS Constellation");
  define funções para OpenGl;

  space = cria janela de tamanho(window, GAP, GAP, spaceWidth, spaceHeight);
  define funções para janela 'space';

  info = cria janela de tamanho(window, GAP+spaceWidth+GAP, GAP, infoWidth,
infoHeight);
  define funções para janela 'info';

  faça initUser();
  faça initSatellites();

  faça timer(50);

  laço_principal_OpenGl();
fim
```

Apêndice C

É apresentado neste apêndice trechos de um arquivo *log.txt*, que registra os valores adotados no cálculo do método apresentado em 2.3.4, além de outros eventos.

```
Iniciando simulacao...

Iteracao: 1
r1: 3.171115        r2: 3.171115        r3: 3.171115
Δd1: 0.943170       Δd2: 0.933004       Δd3: 0.818216
H[1][1]: -0.000000  H[1][2]: -0.000000  H[1][3]: 1.000000
H[2][1]: 0.761877   H[2][2]: -0.212105  H[2][3]: 0.612009
H[3][1]: 0.405863   H[3][2]: -0.579030  H[3][3]: 0.707107
Δx: 0.271476        Δy: -0.190193       Δz: 0.943170

Iteracao: 2
r1: 2.252468        r2: 2.413891        r3: 2.329889
Δd1: 0.024523       Δd2: 0.175780       Δd3: -0.023010
H[1][1]: -0.120524  H[1][2]: 0.084438   H[1][3]: 0.989113
H[2][1]: 0.888409   H[2][2]: -0.199850  H[2][3]: 0.413267
H[3][1]: 0.435884   H[3][2]: -0.706461  H[3][3]: 0.557600
Δx: 0.226465        Δy: 0.200161        Δz: 0.035301

Iteracao: 3
r1: 2.248496        r2: 2.251852        r3: 2.372383
Δd1: 0.020551       Δd2: 0.013741       Δd3: 0.019484
H[1][1]: -0.221455  H[1][2]: -0.004433  H[1][3]: 0.975160
H[2][1]: 0.851769   H[2][2]: -0.303118  H[2][3]: 0.427328
H[3][1]: 0.332617   H[3][2]: -0.778178  H[3][3]: 0.532733
Δx: 0.001980        Δy: -0.009486       Δz: 0.021481

Iteracao: 4
r1: 2.227975        r2: 2.238193        r3: 2.352936
Δd1: 0.000030       Δd2: 0.000082       Δd3: 0.000037
H[1][1]: -0.224384  H[1][2]: -0.000216  H[1][3]: 0.974501
H[2][1]: 0.856082   H[2][2]: -0.300730  H[2][3]: 0.420339
H[3][1]: 0.334525   H[3][2]: -0.780578  H[3][3]: 0.528006
Δx: 0.000078        Δy: 0.000019        Δz: 0.000049

Iteracao: 5
r1: 2.227945        r2: 2.238111        r3: 2.352899
Δd1: 0.000000       Δd2: 0.000000       Δd3: 0.000000
H[1][1]: -0.224422  H[1][2]: -0.000224  H[1][3]: 0.974492
H[2][1]: 0.856079   H[2][2]: -0.300749  H[2][3]: 0.420332
H[3][1]: 0.334497   H[3][2]: -0.780599  H[3][3]: 0.527994
Δx: -0.000000       Δy: -0.000000       Δz: 0.000000
Posicao calculada do usuario: 0.500000, 0.000500, 1.000000

(...)

Iteracao: 1
r1: 3.171115        r2: 3.171115        r3: 3.171115
Δd1: 1.023855       Δd2: 0.966034       Δd3: 0.834235
H[1][1]: 0.472249   H[1][2]: 0.381810   H[1][3]: 0.794483
H[2][1]: 0.064574   H[2][2]: 0.242256   H[2][3]: 0.968061
H[3][1]: 0.727701   H[3][2]: 0.436599   H[3][3]: 0.528991
Δx: 16.954118       Δy: -37.581217      Δz: 9.271654

Iteracao: 2
r1: 42.300307       r2: 42.304641       r3: 42.314520
Δd1: 40.153047            Δd2: 40.099561            Δd3: 39.977640

H[1][1]: -0.365401  H[1][2]: 0.917061   H[1][3]: -0.159627
H[2][1]: -0.395922  H[2][2]: 0.906507   H[2][3]: -0.146599
```

```
H[3][1]: -0.346134  H[3][2]: 0.920859   H[3][3]: -0.179469
Δx: -9.259552       Δy: 41.449713       Δz: 7.782730

Iteracao: 3
r1: 16.022872       r2: 16.164048       r3: 16.481449
Δd1: 13.875611            Δd2: 13.958968            Δd3: 14.144570

H[1][1]: -0.386760  H[1][2]: -0.165871  H[1][3]: -0.907140
H[2][1]: -0.463361  H[2][2]: -0.191801  H[2][3]: -0.865164
H[3][1]: -0.326849  H[3][2]: -0.150714  H[3][3]: -0.932982
Δx: 20.164892       Δy: -78.291307            Δz: -9.577699

Iteracao: 4
r1: 80.249375       r2: 80.236485       r3: 80.207706
Δd1: 78.102115            Δd2: 78.031405            Δd3: 77.870826

H[1][1]: -0.328500  H[1][2]: 0.942482   H[1][3]: -0.061774
H[2][1]: -0.344665  H[2][2]: 0.937118   H[2][3]: -0.054923
H[3][1]: -0.318571  H[3][2]: 0.945138   H[3][3]: -0.072302
Δx: -8.065848       Δy: 82.360170       Δz: 35.135986

Iteracao: 5
r1: 44.580977       r2: 44.707441       r3: 44.993205
Δd1: 42.433717            Δd2: 42.502361            Δd3: 42.656325

H[1][1]: -0.410401  H[1][2]: -0.150885  H[1][3]: -0.899336
H[2][1]: -0.438156  H[2][2]: -0.160357  H[2][3]: -0.884480
H[3][1]: -0.388636  H[3][2]: -0.145641  H[3][3]: -0.909808
Δx: 28.229536       Δy: -145.810091           Δz: -35.602480

Iteracao: 6
r1: 146.727694            r2: 146.708683            r3: 146.666031
Δd1: 144.580434           Δd2: 144.503602           Δd3: 144.329152

H[1][1]: -0.317088  H[1][2]: 0.947902   H[1][3]: -0.030606
H[2][1]: -0.325941  H[2][2]: 0.945009   H[2][3]: -0.026858
H[3][1]: -0.311698  H[3][2]: 0.949485   H[3][3]: -0.036359
Δx: -8.081798       Δy: 152.342652            Δz: 78.027391

Iteracao: 7
r1: 91.994466       r2: 92.116471       r3: 92.392803
Δd1: 89.847206            Δd2: 89.911391            Δd3: 90.055923

H[1][1]: -0.417892  H[1][2]: -0.144130  H[1][3]: -0.896991
H[2][1]: -0.431373  H[2][2]: -0.148743  H[2][3]: -0.889827
H[3][1]: -0.407323  H[3][2]: -0.141628  H[3][3]: -0.902236
Δx: 42.076005       Δy: -258.376289           Δz: -78.251293

Iteracao: 8
r1: 258.038858            r2: 258.016988            r3: 257.967738
Δd1: 255.891598           Δd2: 255.811908           Δd3: 255.630858

H[1][1]: -0.312045  H[1][2]: 0.949923   H[1][3]: -0.016536
H[2][1]: -0.317082  H[2][2]: 0.948289   H[2][3]: -0.014404
H[3][1]: -0.308991  H[3][2]: 0.950859   H[3][3]: -0.019804
Δx: -8.530389       Δy: 269.170256            Δz: 148.850732

Iteracao: 9
r1: 170.897734            r2: 171.017732            r3: 171.289988
Δd1: 168.750474           Δd2: 168.812652           Δd3: 168.953108

H[1][1]: -0.421243  H[1][2]: -0.140746  H[1][3]: -0.895961
H[2][1]: -0.428506  H[2][2]: -0.143235  H[2][3]: -0.892113
H[3][1]: -0.415549  H[3][2]: -0.139409  H[3][3]: -0.898824
Δx: 65.385342       Δy: -446.130050           Δz: -149.005078

Iteracao: 10
r1: 443.889250            r2: 443.865890            r3: 443.813120
```

```
Δd1: 441.741990          Δd2: 441.660810          Δd3: 441.476241

H[1][1]: -0.309480  H[1][2]: 0.950861   H[1][3]: -0.009265
H[2][1]: -0.312409  H[2][2]: 0.949914   H[2][3]: -0.008025
H[3][1]: -0.307708  H[3][2]: 0.951415   H[3][3]: -0.011163
Δx: -9.450314       Δy: 464.093539           Δz: 266.717337

Iteracao: 11
r1: 302.454765           r2: 302.573711           r3: 302.843936
Δd1: 300.307504          Δd2: 300.368631          Δd3: 300.507056

H[1][1]: -0.422954  H[1][2]: -0.138919  H[1][3]: -0.895439
H[2][1]: -0.427060  H[2][2]: -0.140327  H[2][3]: -0.893268
H[3][1]: -0.419736  H[3][2]: -0.138166  H[3][3]: -0.897069
Δx: 104.399690           Δy: -759.426400          Δz: -266.869297

Iteracao: 12
r1: 754.100118           r2: 754.075939           r3: 754.021183
Δd1: 751.952858          Δd2: 751.870859          Δd3: 751.684303

H[1][1]: -0.308081  H[1][2]: 0.951346   H[1][3]: -0.005252
H[2][1]: -0.309806  H[2][2]: 0.950789   H[2][3]: -0.004522
H[3][1]: -0.307039  H[3][2]: 0.951676   H[3][3]: -0.006369
Δx: -11.074550           Δy: 789.381035           Δz: 463.271030

(...)

Iteracao: 124
r1: 62464893310155072.000000          r2: 62464893310155072.000000
      r3: 62464893310155072.000000
Δd1: 62464893310155072.000000          Δd2: 62464893310155072.000000
      Δd3: 62464893310155072.000000
H[1][1]: 0.307219   H[1][2]: -0.951638  H[1][3]: 0.001235
H[2][1]: 0.307219   H[2][2]: -0.951638  H[2][3]: 0.001235
H[3][1]: 0.307219   H[3][2]: -0.951638  H[3][3]: 0.001235
Δx: -11417406040259.072000      Δy: -65643026277240152.000000          Δz:  -
0.000000

Iteracao: 125
r1: 20177812887985316.000000          r2: 20177812887985316.000000
      r3: 20177812887985316.000000
Δd1: 20177812887985312.000000          Δd2: 20177812887985312.000000
      Δd3: 20177812887985312.000000
H[1][1]: 0.951630   H[1][2]: 0.307221   H[1][3]: 0.003823
H[2][1]: 0.951630   H[2][2]: 0.307221   H[2][3]: 0.003823
H[3][1]: 0.951630   H[3][2]: 0.307221   H[3][3]: 0.003823
Δx: 11742972589064326.000000          Δy: 28570015763944524.000000
      Δz: 58983258351369336.000000

Iteracao: 126
r1: 63450956169165712.000000          r2: 63450956169165712.000000
      r3: 63450956169165712.000000
Δd1: 63450956169165712.000000          Δd2: 63450956169165712.000000
      Δd3: 63450956169165712.000000
H[1][1]: 0.117553   H[1][2]: -0.352571  H[1][3]: -0.928372
H[2][1]: 0.117553   H[2][2]: -0.352571  H[2][3]: -0.928372
H[3][1]: 0.117553   H[3][2]: -0.352571  H[3][3]: -0.928372
Δx: 26053315113958.332000       Δy: 9780670658887726.000000           Δz:  -
72057594037927936.000000

Iteracao: 127
r1: 35523727522126604.000000          r2: 35523727522126604.000000
      r3: 35523727522126604.000000
Δd1: 35523727522126600.000000          Δd2: 35523727522126600.000000
      Δd3: 35523727522126600.000000
H[1][1]: 0.209235   H[1][2]: -0.905075  H[1][3]: 0.370217
H[2][1]: 0.209235   H[2][2]: -0.905075  H[2][3]: 0.370217
H[3][1]: 0.209235   H[3][2]: -0.905075  H[3][3]: 0.370217
```

```
Δx: -1.#IND00       Δy: -1.#IND00       Δz: -1.#IND00
Posicao calculada do usuario: -1.#IND00, -1.#IND00, -1.#IND00

(...)

Iteracao: 1
r1: 3.171115        r2: 3.171115        r3: 3.171115
Δd1: 1.059800       Δd2: 0.776377       Δd3: 0.633263
H[1][1]: 0.198597   H[1][2]: 0.143828   H[1][3]: 0.969470
H[2][1]: 0.830662   H[2][2]: 0.347727   H[2][3]: 0.434841
H[3][1]: -0.112219  H[3][2]: 0.626336   H[3][3]: 0.771434
Δx: 0.468221        Δy: -0.163147       Δz: 1.021462

Iteracao: 2
r1: 2.150281        r2: 2.534017        r3: 2.707195
Δd1: 0.038966       Δd2: 0.139279       Δd3: 0.169343
H[1][1]: 0.075131   H[1][2]: 0.287982   H[1][3]: 0.954684
H[2][1]: 0.854731   H[2][2]: 0.499535   H[2][3]: 0.141068
H[3][1]: -0.304404  H[3][2]: 0.793932   H[3][3]: 0.526317
Δx: 0.024844        Δy: 0.246317        Δz: -0.035441

Iteracao: 3
r1: 2.125718        r2: 2.403729        r3: 2.544516
Δd1: 0.014403       Δd2: 0.008992       Δd3: 0.006664
H[1][1]: 0.064312   H[1][2]: 0.175435   H[1][3]: 0.982388
H[2][1]: 0.890723   H[2][2]: 0.424138   H[2][3]: 0.163459
H[3][1]: -0.333630  H[3][2]: 0.747888   H[3][3]: 0.573895
Δx: 0.006924        Δy: 0.001271        Δz: 0.013981

Iteracao: 4
r1: 2.111324        r2: 2.394772        r3: 2.537891
Δd1: 0.000009       Δd2: 0.000034       Δd3: 0.000040
H[1][1]: 0.061471   H[1][2]: 0.176029   H[1][3]: 0.982464
H[2][1]: 0.891164   H[2][2]: 0.425194   H[2][3]: 0.158232
H[3][1]: -0.337229  H[3][2]: 0.749339   H[3][3]: 0.569884
Δx: 0.000011        Δy: 0.000059        Δz: -0.000002

Iteracao: 5
r1: 2.111315        r2: 2.394738        r3: 2.537852
Δd1: 0.000000       Δd2: 0.000000       Δd3: 0.000000
H[1][1]: 0.061466   H[1][2]: 0.176002   H[1][3]: 0.982469
H[2][1]: 0.891172   H[2][2]: 0.425175   H[2][3]: 0.158235
H[3][1]: -0.337238  H[3][2]: 0.749327   H[3][3]: 0.569894
Δx: 0.000000        Δy: 0.000000        Δz: 0.000000
Posicao calculada do usuario: 0.500000, 0.084500, 1.000000


Diminuiu velocidade do usuario.
Tempo de execucao: 19.213000 seg

Iteracao: 1
r1: 3.171115        r2: 3.171115        r3: 3.171115
Δd1: 1.061913       Δd2: 0.773869       Δd3: 0.634717
H[1][1]: 0.202653   H[1][2]: 0.140788   H[1][3]: 0.969077
H[2][1]: 0.833460   H[2][2]: 0.344930   H[2][3]: 0.431704
H[3][1]: -0.108770  H[3][2]: 0.627990   H[3][3]: 0.770583
Δx: 0.466418        Δy: -0.162378       Δz: 1.021852

Iteracao: 2
r1: 2.146896        r2: 2.536929        r3: 2.705280
Δd1: 0.037695       Δd2: 0.139683       Δd3: 0.168882
H[1][1]: 0.082081   H[1][2]: 0.283587   H[1][3]: 0.955427
H[2][1]: 0.857958   H[2][2]: 0.495162   H[2][3]: 0.136831
H[3][1]: -0.299910  H[3][2]: 0.796149   H[3][3]: 0.525548
Δx: 0.026659        Δy: 0.245794        Δz: -0.035793

Iteracao: 3
r1: 2.123609        r2: 2.406176        r3: 2.543068
```

```
Δd1: 0.014408       Δd2: 0.008930       Δd3: 0.006671
H[1][1]: 0.070427   H[1][2]: 0.170954   H[1][3]: 0.982759
H[2][1]: 0.893500   H[2][2]: 0.419918   H[2][3]: 0.159142
H[3][1]: -0.329523  H[3][2]: 0.750279   H[3][3]: 0.573145
Δx: 0.006912        Δy: 0.001275        Δz: 0.013943

Iteracao: 4
r1: 2.109210        r2: 2.397280        r3: 2.536437
Δd1: 0.000009       Δd2: 0.000034       Δd3: 0.000039
H[1][1]: 0.067631   H[1][2]: 0.171516   H[1][3]: 0.982857
H[2][1]: 0.893933   H[2][2]: 0.420944   H[2][3]: 0.153916
H[3][1]: -0.333109  H[3][2]: 0.751738   H[3][3]: 0.569147
Δx: 0.000011        Δy: 0.000059        Δz: -0.000002

Iteracao: 5
r1: 2.109201        r2: 2.397246        r3: 2.536397
Δd1: 0.000000       Δd2: 0.000000       Δd3: 0.000000
H[1][1]: 0.067626   H[1][2]: 0.171489   H[1][3]: 0.982862
H[2][1]: 0.893941   H[2][2]: 0.420926   H[2][3]: 0.153920
H[3][1]: -0.333119  H[3][2]: 0.751727   H[3][3]: 0.569156
Δx: 0.000000        Δy: 0.000000        Δz: 0.000000
Posicao calculada do usuario: 0.500000, 0.084750, 1.000000

(...)

Finalizando simulacao.
Tempo de execucao: 29.867000 seg
```

Referências Bibliográficas

[1] Segantine, P. C. L. “GPS: Sistema de Posicionamento Global”. Departamento de Engenharia de Transportes. EESC-USP. São Carlos (SP), 2005

[2] “The Decca Navigator - Principles and Performance of the System”. The Decca Navigator Company Limited, July 1976

[3] "LORAN-C General Information". United States Coast Guard. Disponível em <http://www.navcen.uscg.gov/?pageName=loranMain>. Acessado em 14 de julho de 2011

[4] Seeber, G. “Satellite Geodesy”. 2ª Ed. 2003

[5] BERNARDI, J. V. E., LANDIM, P. M. B. “Aplicação do Sistema de Posicionamento Global (GPS) na coleta de dados”. DGA, IGCE, UNESP/Rio Claro, Lab. Geomatemática, Texto Didático 10, 31 pp. 2002. Disponível em <http://www.rc.unesp.br/igce/aplicada/textodi.html>. Acesso em 14 de julho de 2011

[6] Huerta, E., Mangiaterra, A., Noguera, G. “GPS: posicionamiento satelital”. 1ª ed. UNR Editora. Universidad Nacional de Rosario, 2005.

[7] "Commission awards major contracts to make Galileo operational early 2014”. 07 de janeiro de 2010. Disponível em <http://europa.eu/rapid/pressReleasesAction.do?reference=IP/10/7&language=en>

[8] Kaplan, E., Hegarty, C. “Understanding GPS: principles and applications”. 2ª ed. Artech House, 2006.

[9] Dantas, A. “GPS: aplicativos e serviços de mapas localizam ruas e até parentes pela tela do celular”. O Globo Online. Publicado em 07/01/2008. Disponível em <http://oglobo.globo.com/tecnologia/mat/2008/01/07/327902920.asp>

[10] “Block II Satellite Information”. United States Naval Observatory (USNO). Disponível em <ftp://tycho.usno.navy.mil/pub/gps/gpsb2.txt>. Acessado em 06 de setembro de 2011

[11] Krebs, G. D., “GPS Navigation System”. Disponível em <http://space.skyrocket.de/doc_sat/gps.htm>. Acessado em 06 de setembro de 2011.

[12] “GPS Constellation Status”. United States Naval Observatory (USNO). Disponível em <ftp://tycho.usno.navy.mil/pub/gps/gpstd.txt>. Acessado em 06 de setembro de 2011.

[13] “Second GPS Block II-F Satellite Successfully Launched”. GPS World. 18/07/2011. Disponível em <http://www.gpsworld.com/gnss-system/gps-modernization/news/second-gps-block-ii-f-satellite-successfully-launched-11891>

[14] Jorgensen, P. S. “Various Uses of the GPS Operational Control System (OCS) Tracking Data”. Disponível em <http://tycho.usno.navy.mil/ptti/1986/Vol%2018_21.pdf>

[15] Coffey, J., “Radius of the Earth”. 06 de março de 2009. Disponível em <http://www.universetoday.com/26629/radius-of-the-earth/>

[16] Bowring, B. R., “Transformation from Spatial to Geographical Coordinates”, Survey Review, Vol. XXIII, No. 181, pp. 323-327, 1976.

[17] “The open source, cross platform, free C++ IDE”. Disponível em <http://www.codeblocks.org/>. Acessado em 20 de outubro de 2011.

[18] OpenGL Overview. “OpenGL - The Industry's Foundation for High Performance Graphics”. Disponível em <http://www.opengl.org/about/overview/>. Acessado em 20 de outubro de 2011.

[19] DLL-files.com. “glut32.dll free download”. Disponível em <http://www.dll-files.com/dllindex/dll-files.shtml?glut32>. Acessado em 20 de outubro de 2011.