|  |
| --- |
| Senac |
| Simulação de Canhão, Avião e Radar |
| PI-V |

|  |
| --- |
| CELSO VENANCIO LEITE  24/5/2016 |

Sumário

[Resumo 2](#_Toc451896447)

[Introdução 3](#_Toc451896448)

[Revisão 3](#_Toc451896449)

[Plano 3](#_Toc451896450)

[Equações Cinemática 3](#_Toc451896451)

[Trajetória do Avião 3](#_Toc451896452)

[Trajetória Balística 3](#_Toc451896453)

[Ângulo Azimute e Elevação 4](#_Toc451896454)

[Método 4](#_Toc451896455)

[Calculando a Interceptação 6](#_Toc451896456)

[Resultado 6](#_Toc451896457)

[Discussão 10](#_Toc451896458)

# Resumo

Este projeto tem como objetivo implementar um simulador de defesa antiaérea no qual um avião tenta atingir um alvo, enquanto, um canhão tenta abatê-lo. Uma aplicação cliente-servidor para simular um radar (servidor) que detecta e informa a posição exata do avião em espaço aéreo não autorizado. Outra aplicação simula o canhão (cliente) que recebe a posição do avião com algum atraso devido ao meio físico e calcula um ponto onde o avião será abatido, informando ao radar sobre o disparo para que este possa verificar se o avião é abatido ou não. O radar exibe toda a simulação em 3D utilizando OpenTK, uma implementação de OpenGL para .NET, e logs sobre as posições e distância entre os objetos são exibidos simultaneamente. O canhão consegue atingir o avião quando o avião é representado por uma esfera de cinco metros de raio simulado numa rede em que o ping entre os computadores é de dois milissegundos em média.

# Introdução

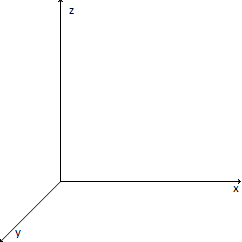
Este relatório apresenta o desenho, a implementação e simulação de um radar que detecta um avião inimigo tentando destruir um alvo. Enquanto um canhão tenta abatê-lo identificando sua trajetória e definindo o melhor momento para disparar seus quatro tiros disponíveis. O avião (uma esfera de raio igual a dois metros no espaço) é abatido quando o tiro (um ponto no espaço) o atinge. O avião destrói o alvo quando chega a menos de um quilometro do alvo. Dois aplicativos foram implementados uma para representar o radar que informa a posição do avião aproximadamente quatro vezes por segundo ao canhão. O canhão, a partir destas posições, tenta definir um ponto para tentar interceptar o avião com um tiro. Aguardando o melhor momento para informar o radar que o tiro foi disparado, considerando o tempo necessário para que a informação chegue ao radar. Como a precisão necessária para que o canhão atinja o avião é muito alta, considerar adequadamente o tempo gasto na comunicação para informar o disparo é um fator decisivo para que este acerte seu alvo.

# Revisão

Para compreensão do projeto é necessário a compreensão de alguns conceitos matemáticos. Esses conceitos são apresentados a seguir.

## Plano

Para aplicações que projetam objetos no espaço tridimensional é importante definir qual eixo representa altura, profundidade e largura. Todas as equações de agora em diante utilizaram os eixos como demonstrado na figura abaixo.



Podemos ver que o eixo Z é representa o eixo vertical. Isso é importante para as equações a seguir.

## Equações Cinemática

### Trajetória do Avião

Falaremos muito sobre a trajetória do avião que foi definido como uma reta no espaço. A seguinte equação define esta trajetória no instante t,

Onde é a posição inicial do avião, é o vetor velocidade do avião.

### Trajetória Balística

A trajetória dos tiros é definida pela equação,

Onde e equivalem a posição inicial do tiro e o vetor velocidade do disparo respectivamente assim como no caso do avião e o vetor constante , onde é a aceleração da gravidade.

### Ângulo Azimute e Elevação

A trajetória definida pela equação acima só é útil se o vetor velocidade estiver definido. Precisaremos defini-lo a partir de um ângulo azimute e elevação com magnitude definida em 1.175m por segundo. Dado os ângulos azimute e elevação , calcularemos o vetor velocidade com magnitude , como sendo

Podemos simplificar essa multiplicação de matrizes para apenas

Isso dará o vetor velocidade rotacionado em ângulos anti-horário, então podemos ignorar o sinal na última componente do vetor para funcionar no sentido correto já que queremos a elevação. Portanto o vetor final é dado por

Para desenvolver uma aplicação capaz de receber dados

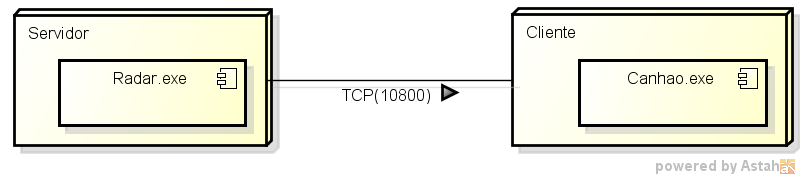
O objetivo do projeto é a experiência no desenvolvimento de aplicativos com comunicação entre computadores utilizando o protocolo TCP/IP. Além do desenvolvimento de interface gráfica em 3D para exibição da simulação

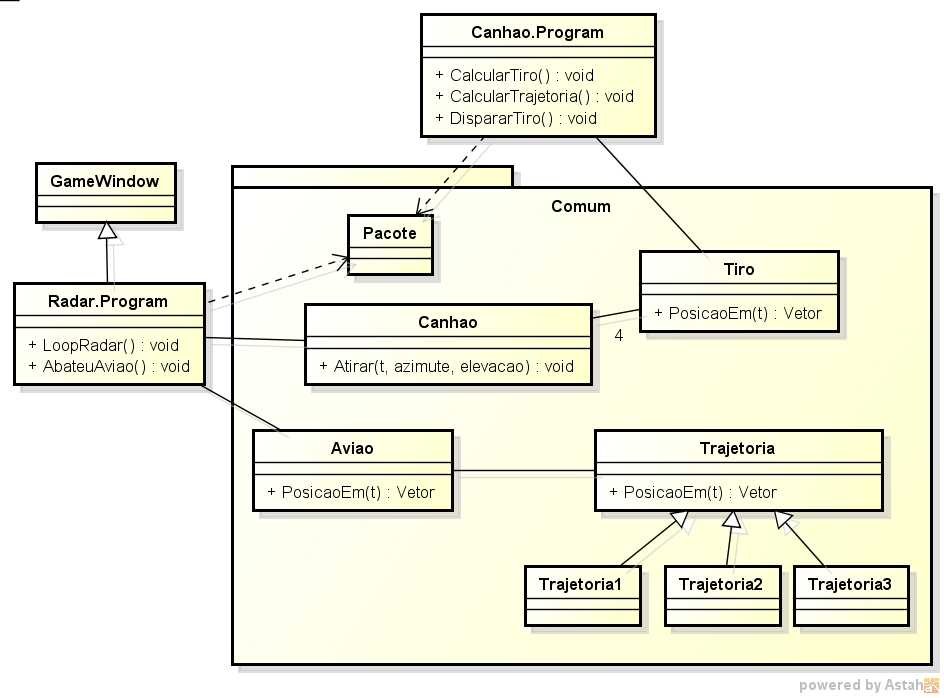
Como primeira entrega foi solicitado um aplicativo cliente (Radar) e outro servidor (Canhão), em que o servidor ao aceitar a conexão do cliente, envia um pacote contendo algum dado e espera outro de volta. O tempo de *throughput* é calculado. Este processo é repetido por 10 interações e o tempo médio é apresentado ao final.

# Método

Ambos aplicativo, cliente e servidor, foram desenvolvidos em *C#* utilizando o *.NET Framework v4* no *Visual Studio 2013*. Para a parte gráfica foi utilizado a biblioteca OpenTK que é um binding para .NET do OpenGL.

As duas aplicação devem ser executadas como mostra o diagrama abaixo:



O diagrama que demonstra a estrutura das duas aplicações e a biblioteca que é comum a ambas é demonstrada abaixo:

|  |  |
| --- | --- |
| **Pacote** | É a estrutura que representa os possíveis pacotes enviados por ambas as aplicações. Os tipo são:   * Posição – Informa ao canhão a posição do avião num determinado momento; * Tiro – Utilizado pelo canhão para informar ao radar que foi efetuado um disparo passando o ângulo de azimute e elevação; * AlvoDestruido – Informa ao canhão que o alvo foi destruído; * AviaoAbatido – Informa ao canhão que um dos disparos acertou o avião; * Ping – O canhão envia ao radar com o tempo; * Pong – O radar responde ao Ping com o Pong preenchendo com o tempo recebido no Ping. |
| **Canhao** | Contém os disparos realizados até o momento e sabe quantos ainda estão disponíveis. |
| **Tiro** | Calcula o vetor velocidade utilizando os ângulos azimute e elevação informados e o momento do disparo. Também calcula a posição da bala num momento t. |
| **Aviao** | Controla qual trajetória deverá ser utilizada seguindo as probabilidades do documento de especificação e a troca das mesmas. Informa também a posição do avião num momento t. |
| **Trajetoria** | Define o vetor velocidade, altura e etc. sobre uma determinada trajetória conforme o documento de especificação. A trajetória número 3 é a trajetória que representa o avião desistindo do ataque. |
| **GameWindow** | Classe base da biblioteca OpenTK para criação de telas em OpenGL. |
| **Radar.Program** | Implementação da lógica do programa do radar. Se comunica com o avião e canhão para saber as posições dos objetos no espaço e os desenham na tela através do OpenGL. Além de cuidar da comunicação com o canhão (TCP) enviando a posição e respondendo solicitações de ping. |
| **Canhao.Program** | Recebe os pacotes do radar para tentar identificar a trajetória do avião e assim poder calcular o melhor momento e direção para efetuar um disparo na tentativa de abater o avião protegendo o alvo. |

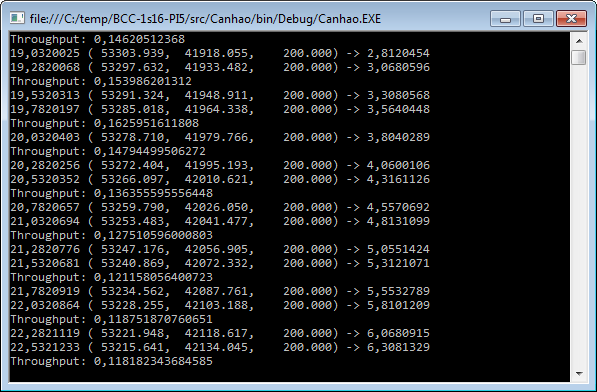
O loop principal da aplicação do radar é controlado pela classe GameWindow do OpenTK. Configurado para executar 60x por segundo. No entanto, somente a cada 4 execuções um pacote com a posição do avião é gerado e enviado a aplicação do canhão. A aplicação do canhão também executa seu loop principal 60x por segundo. Aguardando por pacotes de posição calculando trajetórias e testando o tempo médio do throughput.

## Calculando a Interceptação

# Resultado

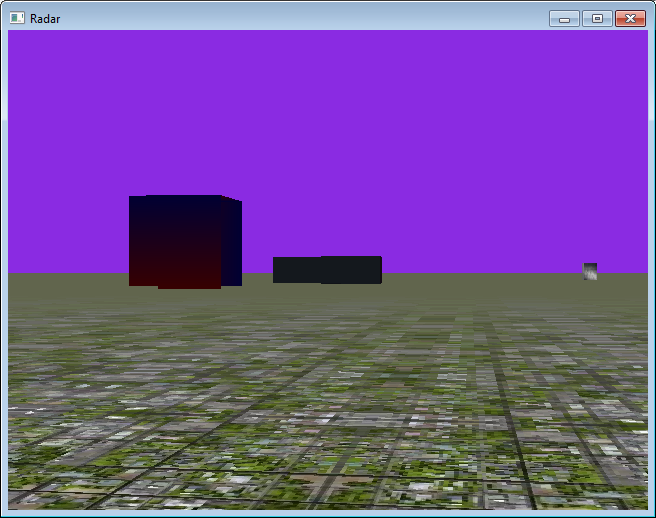
O objetivo de atingir o avião calculando sua trajetória e o momento certo do disparo foi alcançado para um avião de raio igual a 5m para uma comunicação com ping médio de até 2 milissegundos, mas não para 2m como foi especificado. No código há um método implementado para tentar interpolar entre um interação do loop e outra quando for detectado que a trajetória do avião e do tiro se intersectaram, mas sem sucesso. Utilizando este método a precisa é piorada. Provavelmente com mais alguns ajustes este método ficaria bem preciso. Portanto o método utilizado é simplesmente calcular a distância entre o avião e os tiros em cada interação do radar, o que ocorre 60x por segundo.

O print abaixo mostram a execução do aplicativo Canhão conectado ao aplicativo Radar



Os valores apresentados são o throughput (tempo médio de ida e volta dos pacotes Ping/Pong) e a posição do avião junto com o tempo em relação ao radar e o tempo interno do canhão.

Na imagem abaixo, o grande cubo degrade velho-azul representa o alvo, o retângulo cinza escuro o avião e ao fundo está o canhão.

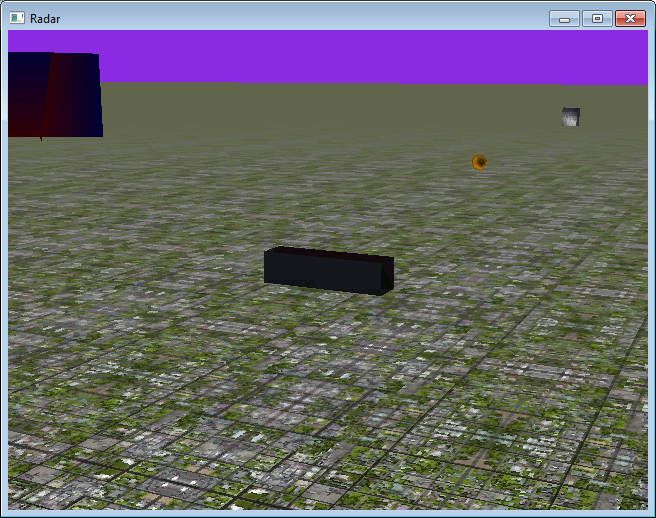


Canhão

Avião

Alvo

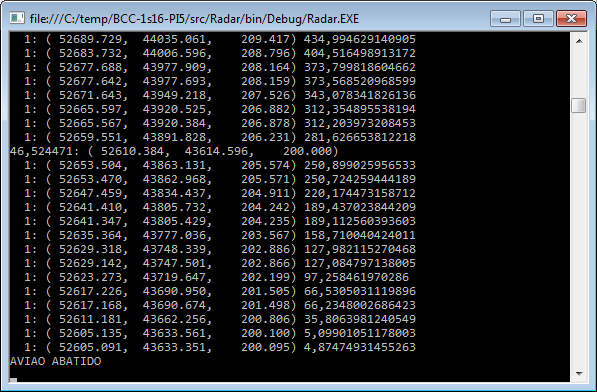
Os objetos foram projetados na tela com o tamanho bem superior ao tamanho no modelo matemático da simulação para facilitar a visualização de tudo o que está ocorrendo. Abaixo uma imagem da simulação no momento de um disparo.



Tiro

Durante a simulação a câmera acompanha o retângulo que representa o avião. O ângulo de visão pode ser alterado utilizando as teclas direcionais do teclado. A tecla control mais os direcionais para cima e para baixo controlam o zoom da cena.

Junto com a representa gráfica o radar disponibiliza várias informações quantitativas sobre o que está acontecedo em cada instante da simulação em outra tela como demonstra a imagem abaixo:



O ciclo de vida dos aplicativos é de apenas uma simulação. Após o término é necessário reiniciá-los para realização de uma nova simulação.

O código das duas aplicações pode ser obtido no *Github* no endereço <https://github.com/celsovlpss/BCC-1s16-PI5>.

# Discussão

A implementação em C# .Net permitiu uma atenção maior nos detalhes da simulação e não apenas no gerenciamento de memória e conexões que são muito mais complexos em C/C++. Mais opções podem ser trabalhadas, modelos para as equações podem ser testados permitindo um maior acerto e qualidade no trabalho final.