ACADEMIA DA FORÇA AÉREA
DIVISÃO DE ENSINO
CURSO DE FORMAÇÃO DE OFICIAIS AVIADORES

GUSTAVO FERREIRA **MAZONAVE**, Cad Av

Desenvolvimento de um simulador de RWR (*Radar Warning Receiver*) para o Clube de Guerra Eletrônica e Cibernética

Pirassununga
2020

ACADEMIA DA FORÇA AÉREA
DIVISÃO DE ENSINO
CURSO DE FORMAÇÃO DE OFICIAIS AVIADORES

GUSTAVO FERREIRA **MAZONAVE**, Cad Av

Desenvolvimento de um simulador de RWR (*Radar Warning Receiver*) para o Clube de Guerra Eletrônica e Cibernética

Trabalho de Conclusão de Curso apresentado no curso de Formação de Oficiais da Academia da Força Aérea – AFA
Orientadora: 1º Tenente Renata Batista Rivero Garcia

Pirassununga
2020

ACADEMIA DA FORÇA AÉREA
DIVISÃO DE ENSINO
CURSO DE FORMAÇÃO DE OFICIAIS AVIADORES

GUSTAVO FERREIRA **MAZONAVE**, Cad Av

Desenvolvimento de um simulador de RWR (*Radar Warning Receiver*) para o Clube de Guerra Eletrônica e Cibernética

Trabalho de Conclusão de Curso apresentado no Curso de Formação de Oficiais Aviadores da Academia da Força Aérea.

Data de aprovação: 06/10/2020

MEMBROS COMPONENTES DA BANCA EXAMINADORA

_____
Orientador: 1º Ten QOCon MFS Renata Batista Rivero Garcia

_____
Membro Titular: 1º Ten QOAv Carlos Guilherme dos Santos Paixão

_____
Membro Titular: Cap QOAv Vinicius Andre Lima Neri

Pirassununga
2020

Ao meu grande irmão Bernardo.

AGRADECIMENTOS

Aos meus pais, Pedro e Sheila, por tudo que fizeram por mim. Ao meu irmão Bernardo pelo apoio, e por estarmos lado a lado sempre nessa carreira.

A todos os professores, os quais tive o prazer de seguir os ensinamentos.

À Tenente Renata, orientadora deste trabalho, por toda a disposição e ajuda fornecida para o desenvolvimento do mesmo.

À Academia da Força Aérea, pela oportunidade de realização do curso, e pelos recursos disponibilizados para a realização do trabalho.

RESUMO

Este trabalho abordará a criação de um jogo de computador que simula, de maneira simplificada, a utilização de *Radar Warning Receivers* (RWRs), instrumentos presentes em diversas aeronaves militares operadas pela Força Aérea Brasileira. Para isso, este trabalho abordará a importância da Guerra Eletrônica no cenário atual da guerra aérea, sua história, e focará na importância do instrumento em questão, o RWR. Será apresentado em seguida os motivos que levaram à decisão de se desenvolver um jogo, ao invés de utilizar *softwares* já existentes ou os equipamentos de simulação de transmissão e recepção de sinais de rádio presentes no mercado. Dessa forma, será abordada a criação do jogo, desenvolvido no *software* de motor de jogos Unity 3d versão 2018, demonstrando a escolha da plataforma de desenvolvimento, as fontes utilizadas para a simulação do instrumento, e registrando o processo de criação e a utilização do jogo. No fim serão registradas as conclusões finais. O trabalho será feito com o objetivo final de deixar o jogo para o Clube de Guerra Eletrônica e Cibernética da Academia da Força Aérea para introdução e estudos da área.

Palavras-chave: *Radar Warning Receiver.* Guerra Eletrônica. Simulação.

ABSTRACT

This paper will adress the creation of a computer game that simulates, in a simplified way, the use of Radar Warning Receivers (RWRs), instruments that are present in many different military aircrafts in use by the Brazillian Air Force. For this, this paper will adress the Electronic Warfare's importance in the current air war scenario, its history, and will focus in the instrument's (RWR) importance. It will be presented next the reasons that led to the decision of developing a game, instead of using existing software or the radio signals transmission and reception simulation equipments available in the market. And so, it will be adressed the game's creation, developed in the game engine software Unity 3d, 2018 version, demonstrating the choice for this development platform, the sources used for the instrument's simulation, and registering the creation process and the gameplay. In the end will be registered the final conclusions. This paper will be made with the final objective of leaving the game to the Electronic Warfare and Cybernetics Club from the Air Force Academy (AFA) for introduction and studies in the area.

Keywords: Radar Warning Receiver. Electronic Warfare. Simulation.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

AAA – Artilharia Anti-Aérea
AFA – Academia da Força Aérea
CGEC – Clube de Guerra Eletrônica e Cibernética
COMGAR – Comando-Geral de Operações Aéreas
DCS – *Digital Combat Simulator*
GE – Guerra Eletrônica
HSI – Indicador de Situação Horizontal
ITA – Instituto Tecnológico de Aeronáutica
LABGE – Laboratório de Guerra Eletrônica
MAWS – Sistema de Alerta de Aproximação de Míssil
RWR – Receptor de Alerta de Radar
SAM – Míssil Ar-Superfície
TWS – Rastreio Enquanto Escaneia
USAF – Força Aérea dos Estados Unidos
USN – Marinha dos Estados Unidos

SUMÁRIO

1 INTRODUÇÃO

Com a crescente importância da área de Guerra Eletrônica (GE) não apenas no âmbito da Força Aérea Brasileira, mas como em praticamente todas as Forças Armadas do mundo, cresce igualmente a necessidade de conhecimento na área pelos militares que operarão as aeronaves equipadas com essas tecnologias, e, portanto, cresce a necessidade de estudo sobre o assunto. O acesso a informações desse campo é, em geral, restrito, por conta de sua própria natureza e do grau de importância dessas tecnologias para a segurança nacional.

Este trabalho tem como objetivo desenvolver um jogo para o Clube de Guerra Eletrônica e Cibernética (CGEC) da Academia da Força Aérea (AFA) que simule a utilização de um instrumento de Guerra Eletrônica presente em diversas aeronaves operadas pela Força Aérea: o *Radar Warning Receiver* (RWR – Receptor de Alerta de Radar, em português), um equipamento composto por diversas antenas ao redor de uma aeronave, que detecta sinais de radar e mostra para o piloto, através de um *display* na nacele, as informações importantes como a natureza do transmissor do sinal (se é amigo ou inimigo), a intensidade e o tipo de sinal de radar detectado.

> O RWR, identificando radares de ameaças, determina cada uma das ameaças presentes. (…) Analisando os parâmetros do radar, o RWR determina também a localização da ameaça relativa à aeronave protegida e o modo de operação da ameaça. Modos de operação da ameaça são geralmente: busca, rastreamento e lançamento.
> (ADAMY, 2006, p. 36, tradução própria)

A criação desse jogo visa apresentar os cadetes pertencentes ao CGEC ao instrumento (RWR), visto que o mesmo pertence ao campo da Guerra Eletrônica, e é nesse campo tido como um dos mais importantes. O jogo, porém, não tem como objetivo o treinamento, muito menos simula com a exatidão necessária para tal o comportamento dos equipamentos presentes, pois para isso seria necessário abordar um equipamento de modelo específico, um programa de simulação complexo (o qual exigiria um desenvolvimento igualmente complexo) e acesso a informações confidenciais presentes nos manuais das aeronaves que operam com esses equipamentos. O público que terá acesso ao jogo, os membros do Clube, é formado por cadetes de diferentes quadros que, no caso dos aviadores, operarão diferentes aeronaves, que têm como objetivo comum entre si o estudo adiantado do campo da Guerra Eletrônica. Portanto o jogo visa apenas a apresentação visual, de maneira a representar genericamente a

simbologia e visualização dos instrumentos, e a introdução ao conceito de utilização dos mesmos.

No mercado, atualmente, diversos simuladores de sinal e recepção de rádio existem, inclusive específicos para simular o comportamento do RWR, mas são equipamentos de *hardware* com preços elevados e utilização complexa. A principal desvantagem desse tipo de material para nosso objetivo está no fato de, por muitas vezes, não oferecer ao usuário uma simulação dos instrumentos, e sim dos sinais de radiofrequência e sua recepção, o que, apesar de ser importante para o aprendizado na área eletrônica, não oferece uma boa apresentação ou introdução ao seu uso operacional. A utilização de jogos (*software*) no lugar de simuladores de sinais para a introdução dessa tecnologia é a opção mais lógica, porém outro obstáculo presente é a falta de *softwares* focados especificamente nesse equipamento embarcado. Existem no mercado atualmente alguns simuladores de combate aéreo virtual, tais como o *Digital Combat Simulator* (*Eagle Dinamycs*), *Falcon BMS*, *VRS Superbug* (*Flight Simulator X*), entre outros, que simulam aeronaves de combate dotadas de equipamentos RWR, e simulam o uso desse equipamento e táticas de combate envolvidas. Porém, além dessas opções serem pagas, o uso de um simulador completo de combate aéreo vai além dos objetivos principais do Clube, pois é necessário, além de um computador com performance muito elevada, aprender a voar as complexas aeronaves simuladas. Dessa forma, a solução encontrada foi desenvolver um jogo específico para o instrumento, de simples utilização e poucos requisitos de performance.

Por esse motivo, esse trabalho utilizará o motor de jogo *Unity* 3d 2018 em sua versão livre, pois é uma plataforma de desenvolvimento simples, amplamente utilizada por desenvolvedores amadores e iniciantes, e que possibilita a criação do jogo de maneira simples, necessitando de poucos requisitos de performance de *hardware* para funcionar.

Será revisada nesse trabalho a história do RWR, instrumento que se busca simular, o qual foi introduzido nos anos 60 pela Força Aérea dos Estados Unidos (USAF) e que teve importante contribuição desde a Guerra do Vietnã.

> A próxima importante fase em desenvolvimento de EW (guerra eletrônica) se deu durante o início dos anos 60 quando a USAF (Força Aérea dos Estados Unidos) e a USN (Marinha dos Estados Unidos) equiparam suas frotas de aeronaves táticas com a primeira geração de sistemas defensivos; *radar warning receivers* e *defensive jammers* (interferidores defensivos).
> (KOPP, 2005, tradução própria)

Em seguida será demonstrada a importância desse equipamento na área da Guerra Eletrônica, e como a sua utilização assim como de outros equipamentos de guerra eletrônica veio a definir a guerra aérea moderna, em que o espectro eletromagnético domina o foco dos combates. Será, então, apresentado o processo de desenvolvimento do jogo no motor *Unity*, a literatura utilizada para referência na criação do instrumento simulado, e a utilização do jogo completo, para, no fim, serem apresentadas as conclusões finais do trabalho.

2 REVISÃO BIBLIOGRÁFICA

Segundo o Manual de Guerra Eletrônica do Comando-Geral de Operações Aéreas (COMGAR, 2013), na guerra moderna, com o crescente uso de sistemas eletrônicos para comunicações e sensoriamento, ganhou importância a Guerra da Informação, definida como sendo as ações para obtenção de superioridade de informação impedindo ou reduzindo seu uso pelo inimigo. Quando a Guerra de Informações é travada no domínio do espectro eletromagnético, é denominada Guerra Eletrônica (GE). Definida pelo mesmo Manual, a Guerra Eletrônica é o conjunto de ações que: utilizam a energia eletromagnética para destruir, neutralizar ou reduzir a capacidade de combate do inimigo; Tiram proveito do uso do espectro eletromagnético pelo inimigo; E visam assegurar o emprego próprio das emissões eletromagnéticas.

2.1 Histórico da Guerra Eletrônica (GE)

O Manual de Guerra Eletrônica do COMGAR (2013), também destaca o desenvolvimento, conforme os anos, da GE, e seu papel na guerra moderna, na qual, vê-se um grande destaque de seu uso na guerra aérea.

2.1.1 Batalha Naval de Tsushima (1905)

O primeiro episódio relatado de emprego de GE ocorreu na Guerra Russo-Japonesa, na Batalha Naval de Tsushima, em 1905. Em janeiro de 1904, o cruzador inglês *HMS Diana*, estacionado no Canal de Suez, interceptou as mensagens transmitidas por telégrafo pela frota russa saindo dos portos, sendo o primeiro relato de interceptação de sinais da história. Durante o conflito, 27 e 28 de maio de 1905, o Japão utilizou dessa habilidade para seu benefício na batalha. A frota russa chegou a interceptar as mensagens iniciais da frota japonesa, e cogitou transmitir na mesma frequência para causar interferência nas mensagens, porém o procedimento foi negado. O uso extensivo de transmissões de telégrafo e a interceptação desses sinais tanto pelos japoneses quanto pelos russos, marcou o uso da GE na batalha. (COMGAR, 2013)

2.1.2 A GE na Primeira e Segunda Guerra Mundial

Na Primeira Guerra Mundial (1914 - 1918) o uso de equipamentos de radiocomunicações de forma extensiva também levou à frequente tática de interferência

dessas transmissões, causando erros de informações ou perda de comunicações. Além disso houve o uso de aparelhos radiogoniométricos para localização de transmissores, como navios, conforme ocorreu em 1916, tendo a Inglaterra localizado uma frota alemã, interceptado os navios e mantido assim a superioridade marítima no conflito.

Na Segunda Guerra Mundial (1939 – 1945) o radar, sistema que usa ondas eletromagnéticas para a detecção de aeronaves, teve seu uso introduzido, primeiramente pelos britânicos, em seguida pelas outras potências na guerra. Com isso, as campanhas de bombardeio aliadas e alemãs foram um jogo de medidas e contramedidas na penetração e defesa dos espaços aéreos. As operações com o espectro eletromagnético foram batizadas de "Batalha dos Feixes de Energia", e Winston Churchill chamou-as de "Guerra Mágica" ou "Batalha dos Mágicos" (Wizard War). (COMGAR, 2013)

Durante a Batalha da Inglaterra, diversos sistemas de auxílio à navegação por rádio feitos pelos alemães para guiar os aviões à ilha britânica sofreram interferência por parte dos ingleses e suas respectivas contramedidas. (COMGAR, 2013)

Em 1943 surgiu um laboratório conjunto, inglês e americano, em Malvern, Inglaterra, para desenvolver projetos de contramedidas eletrônicas, e então foi produzido o primeiro interferidor (*Jammer*) de radar americano, instalado em uma aeronave B-17, tendo sido a primeira aeronave destinada à GE. (COMGAR, 2013)

Um interferidor de radar, ou *Jammer*, envia sinais de ondas eletromagnéticas para o receptor do radar que está interferindo, dessa forma causando confusão na leitura dos sinais de retorno da(s) aeronave(s) que está protegendo. (ADAMY, 2006)

Os *chaffs* também foram criados na Segunda Guerra Mundial, inicialmente pelos ingleses para interferir nos radares alemães, porém os alemães também acabaram criando sua própria versão. Os *chaffs* são aparas metálicas, semelhantes a lâminas, geralmente de alumínio, que são lançadas das aeronaves e espalhadas na atmosfera para interferir e confundir os sinais dos radares inimigos. Seu primeiro uso foi em Hamburgo, 24 de julho de 1943, quando 791 bombardeiros lançaram 20 toneladas de *chaffs*, sobre a cidade alemã. (COMGAR, 2013)

Na Guerra do Pacífico as aeronaves B-29 realizavam missões de reconhecimento eletrônico sobre as ilhas japonesas, além de agirem como interferidores. (COMGAR, 2013)

2.1.3 Guerra do Vietnã (1955 – 1975)

A importância dada à guerra eletrônica aérea atualmente teve como uma grande influência a batalha aérea travada na Guerra do Vietnã, a partir de 1965, com os ataques aéreos americanos na região, quando os mesmos tiveram que enfrentar os sistemas de defesa antiaéreos do Vietnã do Norte, que consistiam de mísseis antiaéreos (SAM), baterias de artilharia antiaéreas (AAA) e seus inúmeros radares de busca de longo alcance e radares diretores de tiro.

Essa densa camada de defesas forçou os americanos a investirem mais em GE, principalmente após um número alto de perdas de aeronaves para essas defesas, sendo produzidas aeronaves interferidoras como o EB-6C *Prowler*; aeronaves especializadas em supressão às defesas antiaéreas, chamadas *Wild Weasels*; além de ter sido aperfeiçoado o uso primordial do *Radar Warning Receiver*, ou alerta de radar, que indicava ao operador ou piloto a direção e natureza de radares inimigos, bem como o lançamento de mísseis guiados por radar em sua direção. (COMGAR, 2013)

Durante os anos a GE teve papel importante e foi chave para o resultado de diversas outras guerras, as quais incluem: Guerra dos Seis Dias, Guerra do Yom Kippur, Guerra das Falklands, Guerra do Golfo Pérsico, entre outras. Com o desenvolvimento de novos avanços tecnológicos esse campo de atuação na guerra torna-se cada vez mais vital para as Forças Armadas, e principalmente para as Forças Aéreas de todo mundo. (COMGAR, 2013)

2.2 O *Radar Warning Receiver* (RWR)

Apesar do conceito de *Radar Warning Receiver* (RWR) existir desde a Segunda Guerra Mundial, este equipamento sofreu modificações e modernizações de seu design e funcionamento com o passar das décadas, e teve uso extensivo a partir da Guerra do Vietnã, onde teve um papel importante nas táticas americanas na batalha contra os SAMs.

2.2.1 História na Guerra do Vietnã

De acordo com o livro de Bernard Nalty (2013), no dia 24 de julho de 1965 o voo *Leopard*, de 4 aeronaves McDonnell Douglas F-4C Phantom, foi atacado por dois mísseis antiaéreos *Guideline*, partes do sistema SAM SA-02 russo, operado pelos norte-vietnamitas. Mesmo recebendo o alerta da aproximação dos mísseis por uma aeronave de reconhecimento

eletrônica RB-66C, a aeronave *Leopard* 02 foi atingida e abatida, enquanto as outras executaram manobras evasivas para escapar do segundo míssil lançado. Esse foi o primeiro evento de uma aeronave americana abatida por um míssil antiaéreo na Guerra do Vietnã.

As defesas aéreas norte-vietnamitas estavam passando por modernizações e atualizações. Seus operadores de radar estavam adquirindo habilidades importantes, conseguindo guiar seus caças MIGs do solo para a interceptação, sendo mais eficientes na detecção de alvos e, apesar do RB-66C ter sido uma aeronave de interferência eletrônica bastante avançada para a época, os norte-vietnamitas estavam conseguindo se adaptar e circundar seu uso pelos americanos, tornando a plataforma de interferência pouco efetiva. Dessa forma, nos meses seguintes um grande número de aeronaves americanas foi abatido, e um número igualmente alto de pilotos morreram ou foram capturados. A maneira para contra-atacar essas defesas foi a criação dos *Wild Weasels* (Doninhas selvagens em português), aeronaves especializadas com tripulações treinadas para localizar e destruir SAMs, utilizando os RWRs para tal.

> Protótipos *Wild Weasels* eram equipados com um escaneador panorâmico receptor e um sistema de guia de vetor e de alerta para detectar e localizar radares inimigos. Ambos os equipamentos obteriam um rumo ao transmissor comparando a intensidade dos sinais de radar adquiridos pelas antenas instaladas em várias partes da aeronave. Esses sinais apareceriam como linhas de luz em visores montados na nacele traseira, e o oficial de guerra eletrônica interpretava suas características para determinar a direção do transmissor e sua frequência de repetição de pulso. Um visor localizado no painel de instrumentos do piloto duplicava a informação disponível ao oficial de guerra eletrônica.
>
> A imagem do visor de rumo, mais uma luz piscante e um barulho alto nos headsets, diziam à tripulação que um radar hostil estava travando e seguindo a aeronave. Outro receptor radar obteria o sinal de guiamento do SAM e usava luzes (mais tarde suplementado por um tom no headset) para sinalizar que um lançamento era iminente. O piloto podia então procurar pelo míssil em aproximação e manobrar para evadi-lo.
> (NALTY, 2013, tradução própria)

Assim nasceram os *Wild Weasels* e suas táticas que, mesmo sendo constantemente atualizadas conforme as tecnologias avançam e os operadores de SAMs se aperfeiçoam, mantém-se em uso na USAF até os dias de hoje, tendo em vista que as defesas antiaéreas são ainda uma grande ameaça para as operações aéreas militares atuais. (NALTY, 2013)

Dessa forma, o *Radar Warning* Receiver (RWR), também chamado de Alerta Radar, que atualmente é visto como um dos equipamentos básicos de guerra eletrônica em aeronaves, pôde demonstrar sua importância na batalha contra os SAMs na Guerra do Vietnã.

2.2.2 O equipamento

Os equipamentos RWR são, segundo o Manual de Guerra Eletrônica do COMGAR (2013):

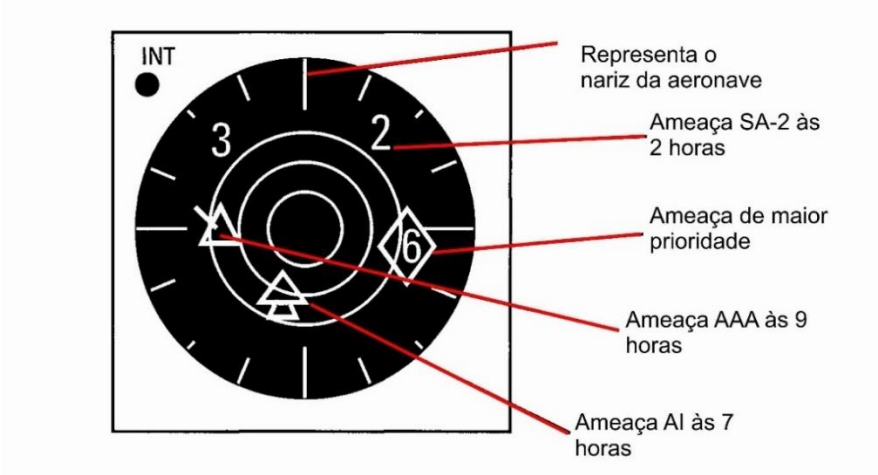
> São equipamentos de banda larga (tipicamente de 2 a 18GHz) que utilizam receptores de cristal-vídeo ou super-heteródinos. Medem frequências (super-heteródinos) ou somente a banda de frequências (cristal-vídeo) (…), dados considerados suficientes para reconhecer o tipo de emissão e alertar a tripulação sobre o provável tipo de ameaça e sua direção.
>
> Os RWR, geralmente, operam associados a sistemas de autodefesa, tais como interferidores e lançadores de *chaff* e *flare*. Sua função de alerta também permite que sejam adotadas as devidas ações táticas, por parte da tripulação, como redução de altura e execução de manobras evasivas.
> (COMGAR, 2013)

De uma forma mais breve, são equipamentos instalados nas aeronaves militares, que detectam as frequências eletromagnéticas utilizadas por radares inimigos e alertam a tripulação quanto a direção, tipo de sinal, e modo de operação do radar.

> O RWR, identificando radares de ameaças, determina cada uma das ameaças presentes. (…) Analisando os parâmetros do radar, o RWR determina também a localização da ameaça relativa à aeronave protegida e o modo de operação da ameaça. Modos de operação da ameaça são geralmente: busca, rastreamento e lançamento.
> (ADAMY, 2006, p. 36, tradução própria)

A Figura 1 representa um *display* de um RWR, onde pode-se observar os símbolos representantes de ameaças, dispostos de maneira a indicar sua posição relativa à aeronave. O número ou forma geométrica indica o tipo de ameaça e a prioridade da ameaça. A proximidade do símbolo com o centro do *display* indica a intensidade do sinal recebido.

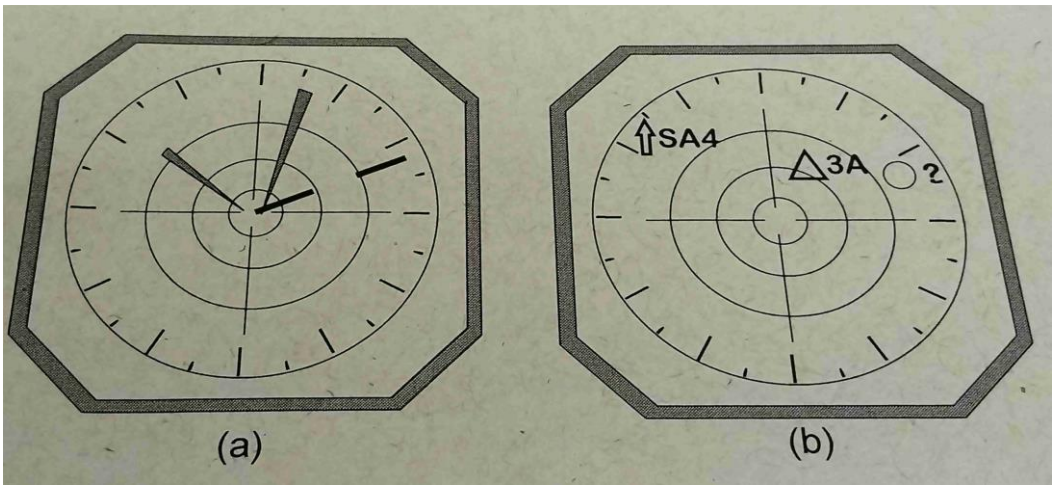
Figura 1: Representação de um *display* de RWR



Fonte: Adamy (2013)

A Figura 2 representa dois tipos de *displays* de RWR, sendo um deles analógico, ou vídeo bruto (a), que indica a direção e intensidade do sinal de forma bruta ao operador, e o outro digital, ou vídeo sintético (b), que filtra as informações e representa as ameaças de maneira detalhada ao operador, além de identificar as ameaças baseado nos dados de memória. Os dois *displays* apresentam o mesmo cenário, três ameaças, sendo uma delas (3A) com maior intensidade.

Figura 2: Modelos de RWR



Fonte: COMGAR (2013)

2.3 MAWS

MAWS, ou *Missile Approach Warning System* (Sistema de Alarme de Aproximação de Míssil), pode ser um sistema por infravermelho ou ultravioleta. Um MAWS tem como objetivo detectar e identificar ameaças que estejam associadas a emissões infravermelhas ou ultravioletas, como mísseis ar-ar e mísseis ar-superfície (COMGAR, 2013). São equipamentos passivos, instalados nas aeronaves para alertarem as tripulações de mísseis lançados que não podem ser detectados com o uso de RWR. Como exemplo de mísseis passivos, ou seja, que não podem ser detectados pelo RWR, pode-se citar os mísseis guiados por espectro infravermelho. Esses mísseis não emitem ondas eletromagnéticas, por isso é necessário um equipamento como o MAWS para detectar e alertar o piloto de seu lançamento. (COMGAR, 2013)

> A história do emprego de sistemas de alarme na faixa do infravermelho teve início em meados de 1950. As investigações básicas procuravam determinar quando deveriam ser iniciadas contramedidas para autodefesa da plataforma, através do lançamento de *flares* (bastonetes incandescentes), contra os sistemas de guiagem ou de busca. Como os ataques com infravermelho podem originar-se em qualquer direção, os requisitos técnicos para os receptores são muito exigentes.
> (COMGAR, 2013)

Um problema enfrentado por esses equipamentos é o alto índice de falsos alarmes, por diversos motivos inerentes à operação dos mesmos. As emissões que escapam de um míssil não são constantes, podendo causar leituras equivocadas pelos sensores. Uma forma de evitar tal problema é utilizar circuitos de memória que armazenam dados da queima e emissão dos mísseis enfrentados pelas aeronaves equipadas, para que os sensores captem a mesma variação nas emissões. Além disso, o uso inicial de sensores infravermelhos levava a altos índices de falsos alarmes, o que levou ao uso de sensores ultravioletas. Existem também equipamentos de alarme de mísseis ativos, consistindo de radares do tipo pulso-Doppler para detectar objetos se aproximando da aeronave. (COMGAR, 2013)

Os equipamentos MAWS em operação atualmente em geral são associados com o sistema de lançamento de *flares* da aeronave em que está embarcada. Dessa forma, as contramedidas são automatizadas, a fim da resposta às ameaças ser mais rápida e eficiente, em comparação com o lançamento manual feito pelo piloto. (COMGAR, 2013)

2.4 Sistemas de Guiagem de Mísseis

Serão apresentados três principais tipos de guiagem utilizados por mísseis na atualidade: A guiagem infravermelha; a guiagem semi-ativa; e a guiagem ativa.

2.4.1 Guiagem infravermelha: é um método passivo, onde o *seeker* (sensor de guiamento) do míssil recebe as ondas infravermelhas do alvo. Não é detectável por sistemas RWR, porém a tripulação pode ser alertada por sistemas MAWS. *Flares*, que são bastonetes incandescentes lançados da aeronave ao comando do piloto (ou automaticamente nos sistemas mais modernos, associados ao MAWS), geralmente são empregados para ajudar a despistar esses mísseis, pois os bastonetes incandescentes podem confundir a leitura do *seeker*. (COMGAR, 2013)

2.4.2 Guiagem semi-ativa: o míssil utiliza a “iluminação” (ondas eletromagnéticas emitidas e refletidas) do radar da própria aeronave ou bateria SAM que o lançou. O RWR consegue dessa forma detectar o lançamento do míssil, pois o modo de operação do radar muda para guiar o mesmo ao alvo, logo são essas emissões que são detectáveis pelo RWR. (COMGAR, 2013)

2.4.3 Guiagem ativa: o míssil possui um radar próprio, e o ativa apenas a uma curta distância do alvo, tendo sido guiado até ele através de um *data-link* com o radar da aeronave que o lançou. Dessa forma, o radar da aeronave não necessita denunciar o lançamento do míssil ao RWR do alvo. Porém, ao ativar seu próprio radar, o míssil emite ondas de radar que são detectáveis pelo RWR e alertam a tripulação, porém com uma janela de resposta bastante curta. *Chaffs* são geralmente empregados tanto contra mísseis ativos quanto semi-ativos, pois as lâminas metálicas ajudam a confundir as leituras dos radares. (COMGAR, 2013)

2.4.4 *Track-While-Scan,* TWS (Rastreio enquanto escaneia): Esse modo de radar existe nas aeronaves de caça mais modernas, e permite com que o radar trave em alvos enquanto continua escaneando, ao contrário do convencional, onde o radar tem de mudar de modo para cumprir uma das duas funções.

> Ameaças do tipo *Track-while-scan* (TWS) são diferentes, pois o processo da busca e a aquisição são feitos pelo mesmo radar, e de modo simultâneo com o rastreio (trava) da ameaça. Isso permite ao radar rastrear múltiplos alvos enquanto procura por mais.
>
> O alvo sendo rastreado não detectará mudanças no modo de radar que o trava ao ser selecionado como alvo.
>
> (ADAMY, 2006, p. 142, tradução própria)

O modo TWS é em geral utilizado em conjunto a mísseis com guiagem ativa, pois, dessa forma, o alvo não receberá em seu RWR o alerta que está sendo travado, nem do disparo do míssil, até o momento do míssil estar próximo o suficiente para ativar o próprio radar a fim de atingir o alvo com precisão. (COMGAR, 2013)

2.5 Simulação de GE e RWR

A simulação e modelagem de comportamento de equipamentos de GE é, por si só, um desafio. Muitas são as propriedades dos sistemas envolvidos, muitas variáveis existem nos cenários simulados, e são emuladas nos programas ondas eletromagnéticas de diversas

frequências e pulsos, e com isso diversas interpretações de sinais e interferências do meio ambiente. (ADAMY, 2006)

Para entender a simulação de GE deve-se entender a simulação em si. Qualquer simulação é uma criação artificial de uma situação ou estímulo que cria um resultado como se a situação real correspondente estivesse acontecendo. Qualquer nível de simulação pode existir para o mesmo objetivo, a exemplo a simulação de voo: um simulador pode ser apenas visual, em uma tela de computador simulando os instrumentos e o cenário voado, mas também pode ser acrescentada uma cabine em tamanho real, com *displays* e instrumentos o mais próximos da realidade possível, com uma tela panorâmica para simular o ambiente externo, e, além disso, é possível acrescentar movimentos tridimensionais a essa simulação para estimular a sensação de movimento ao piloto em treino. Dessa forma, vemos que existem diferentes níveis de realismo para a mesma simulação. (ADAMY, 2006)

Na simulação existem em geral três categorias: modelagem, simulação interface-operador, e emulação. “Modelagem” também é chamada de *simulação de computador*, e “simulação interface-operador” também é comumente referenciada como apenas *simulação*, o que pode acarretar em confusão de termos.

Na simulação interface-operador, o operador deve receber os estímulos que o colocam na situação real correspondente, com os equipamentos que o mesmo opera tendo tido seu funcionamento e comportamento modelados em um computador (modelagem), e devem ser realistas o suficiente para prover ao operador os estímulos que encontraria na realidade, porém o funcionamento em si dos equipamentos simulados por muitas vezes não é replicado com acurácia, mas sim para imitar o aparente funcionamento. Em contrapartida a emulação é a replicação de sinais que serão interpretados pelos equipamentos reais utilizados. Esses sinais devem ser realistas o suficiente, próximos aos sinais que seriam recebidos em um cenário real, para que os equipamentos realmente correspondam da forma como fariam em uma situação real. Ambos os métodos podem ser usados em simulação para treinamento. (ADAMY, 2006)

Na simulação de GE, por muitas vezes são feitas simulações de sinais reais de operadores inimigos para serem adquiridos e interpretados por sistemas de guerra eletrônica reais (emulação). Isso tem como objetivo testar o funcionamento dos próprios equipamentos, e prover o treinamento não precisando da modelagem dos próprios. As desvantagens dessa forma de simulação é o custo de uso e manutenção desses equipamentos, sendo mais barato utilizar um computador dedicado para a simulação, além de muitas vezes não ser possível

simular um cenário próximo do real para o operador. Uma outra prática é simular esses equipamentos junto com simuladores de veículos militares correspondentes, como simuladores de voo no caso da GE na aeronáutica, porém dessa forma é necessário simular (modelagem) o comportamento do equipamento em um programa que será utilizado pelo computador do simulador. (ADAMY, 2006)

Existem no mercado de equipamentos militares diferentes empresas fornecendo diferentes soluções de simulação para equipamentos de GE, em geral focando-se nos métodos de emulação de sinais para uso em conjunto com os equipamentos já empregados. Existem equipamentos feitos especialmente para a simulação com o objetivo de estudos dessas ondas e sinais eletromagnéticos, isso incluindo simulação do funcionamento de radares, RWRs, ou até mesmo interferidores (*Jammers*), mas têm em geral um preço elevado, sendo também equipamentos complexos de difícil operação. Um exemplo de empresa que possui um grande catálogo de equipamentos destinados à simulação de GE é a LabVolt, sendo seus sistemas de treinamento de radar utilizados no Laboratório de Guerra Eletrônica do ITA (Instituto Tecnológico de Aeronáutico). (LABGE, acesso em 2020)

Simulações de equipamentos de GE embarcados em aeronaves, a exemplo o RWR, geralmente vêm em conjunto com o simulador da aeronave em si, salvo algumas exceções, como simuladores de voo que não possuem os equipamentos de GE presentes na aeronave real. Esses simuladores dedicados são uma boa solução para o treinamento de tripulações, quando devidamente desenvolvido com alta fidelidade, apesar do custo elevado.

Existem, sim, no mercado de simulação virtual, simuladores de voo de aeronaves que operam sistemas de GE que podem ser comprados e acessados por usuários casuais, não necessitando utilizar simuladores dedicados das Forças Aéreas operadoras das aeronaves, por exemplo. Exemplos desses simuladores que, apesar de abertos ao público, têm um nível de realismo elevado são: *Digital Combat Simulator* (DCS) da empresa *Eagle Dynamics,* cuja imagem está representada na Figura 3; *Falcon BMS* (4.34), da *Benchmark Simulations*; e *F-18E Superbug*, da *VRS Simulations*.

Figura 3: Imagem do jogo DCS

Fonte: Disponível em: <https://www.digitalcombatsimulator.com/en/downloads/screenshots/1201/ >. Acesso em: 13 abr. 2020.

O *Digital Combat Simulator*, ou DCS, segundo o próprio site do produto, é um jogo de simulação que consiste em uma plataforma única, chamada DCS World, na qual é possível comprar e instalar diversos "módulos", que são aeronaves simuladas, a maioria sendo aeronaves de combate de diversos períodos da história, desde a Segunda Guerra Mundial até a Guerra Fria. Alguns exemplos de aeronaves simuladas que possuem equipamentos de GE como o RWR podem ser listadas: F/A-18 *Hornet*, F-16 *Fighting Falcon*, A-10 *Thunderbolt II*, entre outras mais. (DIGITAL COMBAT SIMULATOR, acesso em 2020)

O *Falcon BMS* 4 é um jogo de simulação que foca na aeronave F-16 *Fighting Falcon*, apesar de possuir outras aeronaves jogáveis como opções, porém não com o mesmo grau de fidelidade. (BENCHMARKSIMS, acesso em 2020)

O *Superbug* foca no F/A-18 *Super Hornet*, e é uma adição (*add-on*) ao *Flight Simulator X* da *Microsoft*. (VRSIMULATIONS, acesso em 2020)

Esses simuladores são focados em uso pessoal para entusiastas de combate aéreo, e não para simulação de treinamento real de pilotos. Apesar disso possuem, por simularem os sistemas das aeronaves, simulação de equipamentos de GE. O uso desses equipamentos

dentro da simulação do jogo é baseado em manuais de voo reais disponíveis pelas empresas que os fabricam, retirando da simulação apenas informações sensíveis sobre o equipamento, julgadas assim pela empresa ou pelas Forças Aéreas que utilizam essas aeronaves. (DIGITAL COMBAT SIMULATOR, acesso em 2020)

As vantagens desse tipo de simulação é o acesso e requisitos baixos de gastos, tanto com o *software* em si, quanto com o computador utilizado para o jogo. As desvantagens podem ser a falta de fidelidade máxima que seria possível com um simulador dedicado, e a falta de equipamentos ou sistemas considerados sensíveis para o uso público.

3 CRIAÇÃO E FUNCIONAMENTO DO JOGO

3.1 O Clube de Guerra Eletrônica e Cibernética (CGEC)

O CGEC da Academia da Força Aérea possui em seu estatuto sua missão e seu objetivo:

> Art. 2° O CGEC tem por missão proporcionar ao cadete mais uma atividade extracurricular, motivando-o a aprender sobre os aspectos gerais das Guerras Eletrônica e Cibernética, fortalecendo seus conhecimentos científico-militares.
>
> Art. 3° O objetivo do CGEC é estimular a busca por conhecimento e pelo entendimento sobre os conceitos das áreas de interesse, com foco em suas aplicações operacionais, por meio de palestras, instruções, viagens de estudo e outras atividades que contribuam com esse objetivo.
>
> (CGEC, 2020)

Em acordo com o objetivo do CGEC, o estudo de equipamentos importantes para a GE é uma das inspirações para o desenvolvimento do jogo, específico para o RWR, visto que o mesmo é um equipamento que ajudou a moldar o uso da GE no combate aéreo. Além disso, o fato de ser apenas um clube para atividades extracurriculares, com foco na motivação ao aprendizado sobre o assunto, justifica a falta de necessidade de simuladores complexos e de nível profissional para cumprir o objetivo final.

Dessa forma foi escolhido desenvolver o próprio jogo, focado em apresentar e demonstrar o funcionamento, a nível operador, de um RWR, ou seja, um simulador simples de RWR. O objetivo final desse trabalho é utilizar esse jogo para incentivar o estudo do RWR e outras áreas da GE, e manter o mesmo no Clube para que as futuras gerações possam aprimorar o jogo, ou até mesmo se inspirar nele e criar novas formas de estudo, incluindo novos jogos e simuladores.

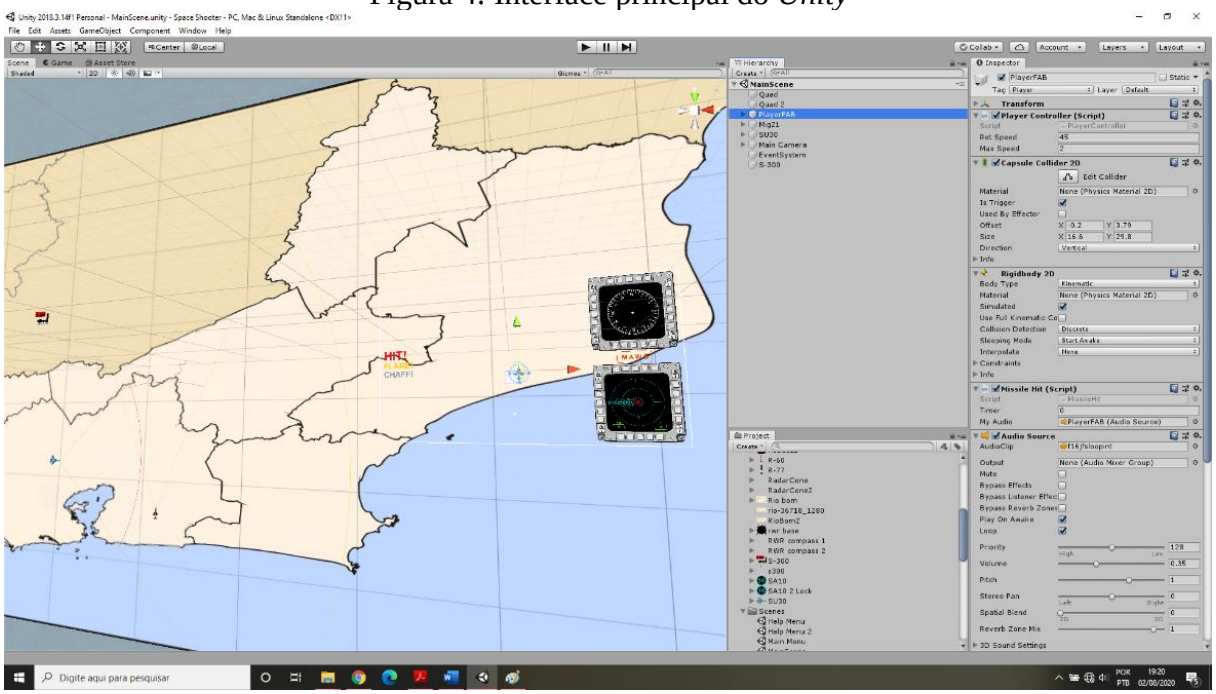
3.2 *Softwares* Utilizados

Para o desenvolvimento do jogo foi optado por utilizar o motor de jogos *Unity* 3d, versão 2018 (Unity 2018.3.14f1 64-bit). Um motor de jogos é um *software* de criação simplificada de jogos, que possui diversas ferramentas ao dispor do usuário para facilitar de forma significativa o desenvolvimento de jogos. (UNITY, acesso em 2020) É possível cria-los dessa forma sem a necessidade de começar do zero através de programação, além de permitir criações mais complexas com o uso das ferramentas embutidas.

> Uma *engine* de jogo (motor de jogo) é o software que fornece aos criadores de jogos o conjunto necessário de recursos para criar jogos de maneira rápida e eficiente.
> Uma engine de jogo é um framework de desenvolvimento de jogos que suporta e reúne várias áreas importantes. Você pode importar gráficos e recursos 2D e 3D de outros softwares, como Maya, 3s Max ou Photoshop, montá-los em cenas e ambientes, adicionar iluminação, áudio, efeitos especiais, física e animação, interatividade e lógica de jogo; e editar, depurar e otimizar o conteúdo para suas plataformas de destino.
> (UNITY, acesso em 2020)

Diversos motores de jogos estão disponíveis para uso, alguns de *software* livre, alguns pagos, alguns baratos e outros caros, para uso pessoal ou para grandes empresas de jogos. O *Unity* foi a opção escolhida por possuir uma versão não paga, que, apesar de não conter todas as funcionalidades e não poder anunciar o jogo comercialmente, atende as necessidades para cumprimento dos objetivos do trabalho. Além disso, o *Unity* é uma plataforma pouco especializada, podendo criar jogos em 3d e 2d, e utiliza a linguagem de programação C# (bastante básica e conhecida na computação), sendo por isso muito utilizada por iniciantes em programação e desenvolvimento de jogos. Isso faz com que haja diversos vídeos, aulas e instruções online sobre iniciar o uso do motor. A Figura 4 apresenta a interface principal do *Unity*, onde é possível ver a "visualização de cena" na esquerda, à direita as seções de "hierarquia", "projeto" e "inspeção", e no topo a barra de ferramentas. A imagem foi obtida por meio de um *Print Screen* da tela do computador.

Figura 4: Interface principal do *Unity*



Fonte: Própria, 2020

Para a criação do jogo também foi necessário utilizar programas de edição e criação de imagens, e para isso foi utilizado o programa *Corel Draw* X8 *Home & Student*, por ser uma plataforma já conhecida e barata.

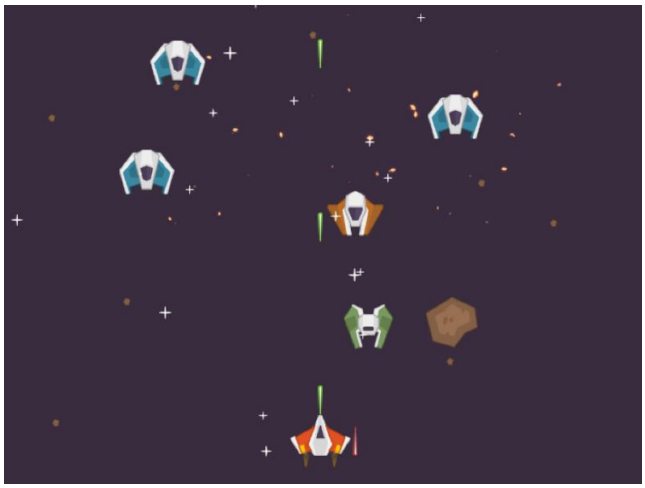
3.3 Estruturação do Jogo

O *Unity* tem a seguinte forma de criar os jogos: os objetos são as criações em 3d ou 2d, pontos de foco das animações e eventos do jogo; Os *scripts* são páginas atreladas aos objetos e são escritos em linguagem de computação, no caso escolhido a linguagem C#, e descrevem as interações que objetos proporcionarão ao jogo e ao jogador.

O objetivo era criar um jogo do tipo *top-down*, ou seja, com a visão do jogador vinda de cima, onde o avião controlado pelo jogador estaria no centro da câmera, com os instrumentos no canto direito. Os instrumentos são o RWR em si, e um HSI (*Horizontal Situation Indicator* – Indicador de Situação Horizontal), que resumidamente, funciona como uma bússola. O jogo se passa em um cenário simples, com três objetos representando "inimigos", um deles fixo e os outros dois rodando em uma trajetória circular. Esses inimigos aparecem no RWR conforme o radar deles atinge a aeronave do jogador, e disparam mísseis de diferentes tipos no jogador que poderão ou não, dependendo do tipo, aparecer no RWR. Apesar de o jogador poder ser atingido pelos mísseis no jogo, não existe simulação da destruição da aeronave e portanto o jogo não pode ser perdido, apenas reiniciado. O jogador tem a sua disposição comandos de *flare* e *chaff* que irão despistar os mísseis, e cada inimigo pode lançar até dois mísseis cada.

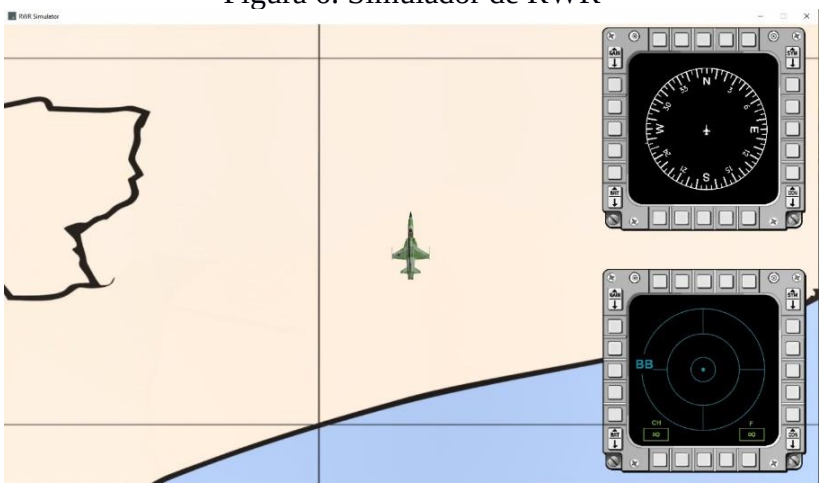
Na Figura 5 é possível ver um exemplo de jogo no estilo *Top-Down*, onde a câmera se encontra visualizando o jogo por cima. O mesmo jogo também pertence à categoria de *Space-Shooter*, outra inspiração para o jogo desenvolvido. A Figura 6 é uma imagem retirada do jogo desenvolvido neste trabalho, já pronto, para comparação da perspectiva.

Figura 5: Jogo *Space-Shooter*



Fonte: Disponível em:
< http://developers.gausstoys.com/projects/26>

Figura 6: Simulador de RWR



Fonte: Própria, 2020

3.3.1 *Scenes* ou Cenas

O jogo está dividido em três cenários, o Menu do jogo (com uma seção de ajuda ao jogador), o jogo no modo normal, e o jogo no modo sem mísseis. Sendo o primeiro subdividido em “Main Menu” e “Help Menu” e “Help Menu 2”, o segundo em “Main Scene” e o último em “NoMissiles”

O jogador começa no “Main Menu” ao iniciar, podendo ir então para a página de ajuda onde encontra explicações sobre o jogo, ou escolhe entre um dos modos, o normal ou sem

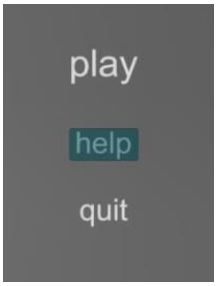
mísseis, para jogar. Durante um desses modos, se o jogador pressionar a tecla “P” o jogo pausa. Se pressionar a tecla “Escape”, volta ao Menu principal.

As Figuras a seguir (7 a 10) são imagens retiradas do jogo e representam os diferentes “Menus” e telas que o jogador encontrará:

Figura 7: Menu principal do jogo

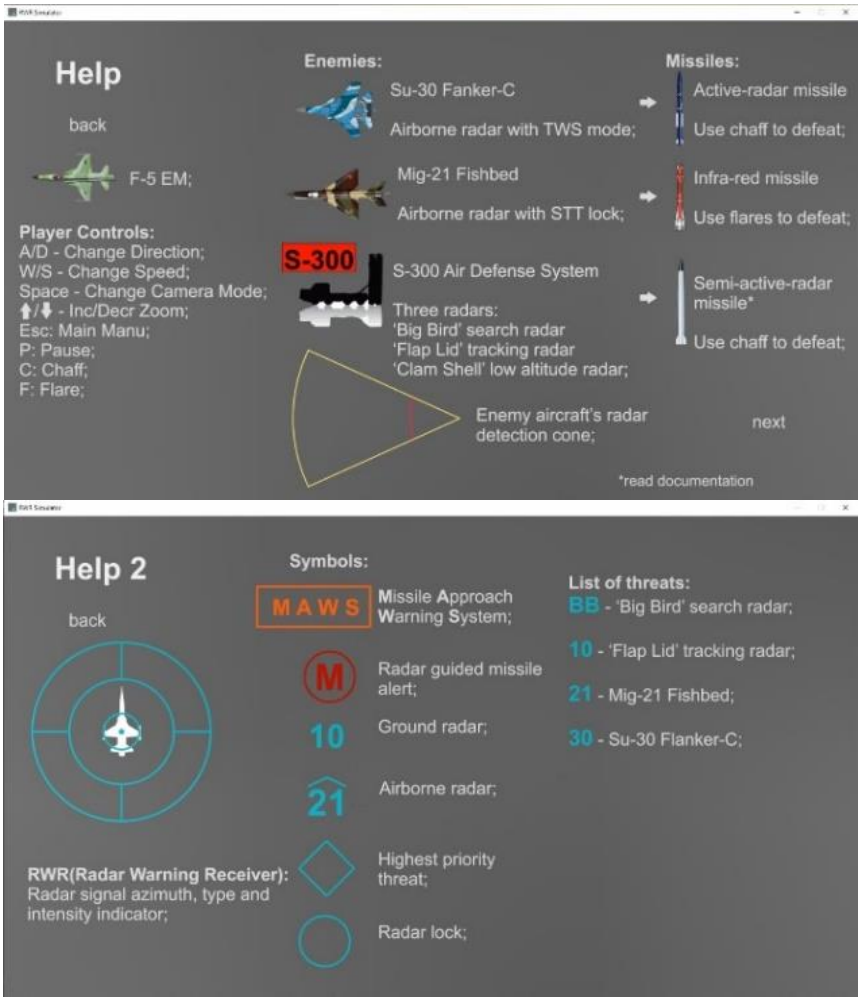
Fonte: Própria, 2020

Figura 8: Botão para o Menu de Ajuda



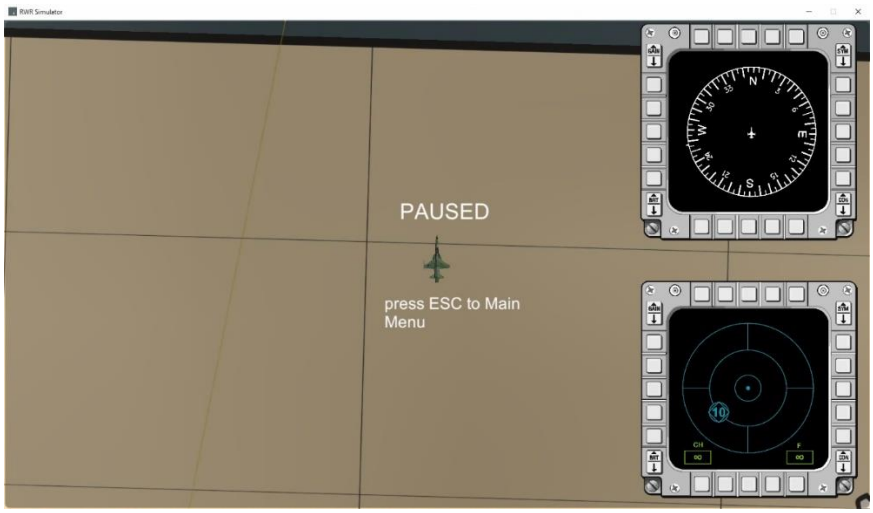
Fonte: Própria, 2020

Figura 9: Menus de Ajuda



Fonte: Própria, 2020

Figura 10: Pausa no jogo



Fonte: Própria, 2020

3.3.2 Objetos

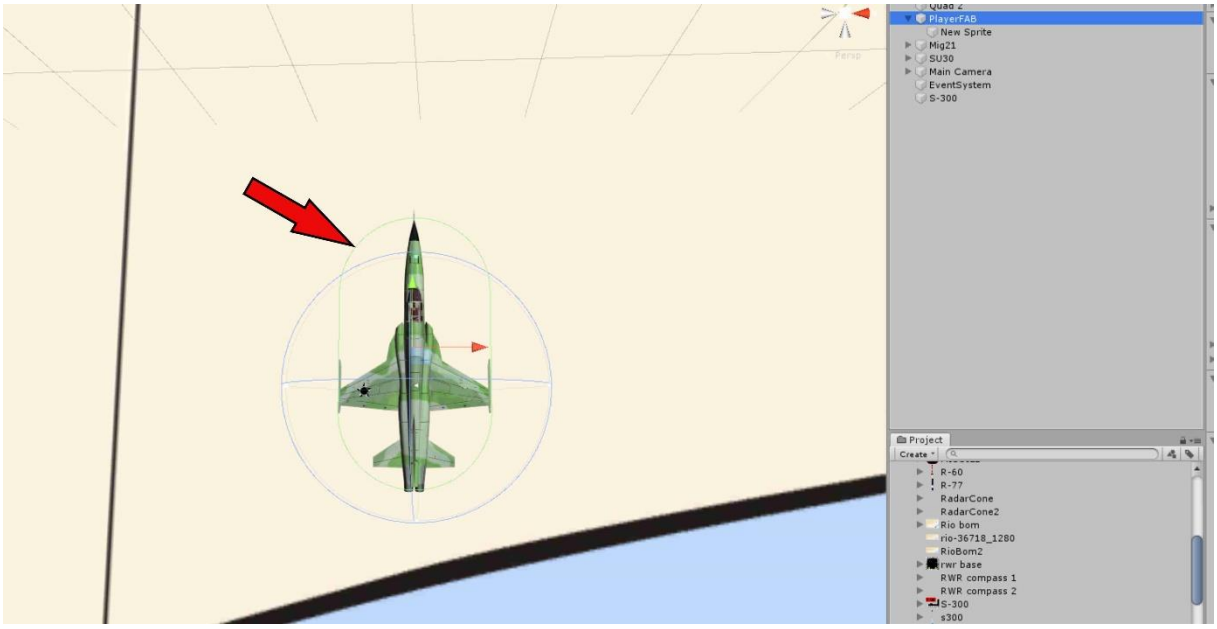
Os objetos principais no jogo em si são:

O cenário, sendo um plano de dimensões bem maiores que o campo de visão da câmera, com uma textura de um mapa da cidade do Rio de Janeiro, em uma resolução alta para ter bastante qualidade de imagem durante o jogo em si;

A aeronave do jogador, outro plano, com o formato de um F-5EM da Força Aérea Brasileira, com uma cápsula de detecção de colisões em formato de oval. Essa cápsula no objeto serve para criar eventos quando ocorrem colisões, e isso para o jogo serve para simular a detecção do RWR e quando um míssil atinge a aeronave;

A figura 11 mostra a cápsula de colisão em verde da aeronave do jogador:

Figura 11: Cápsula de colisão do jogador

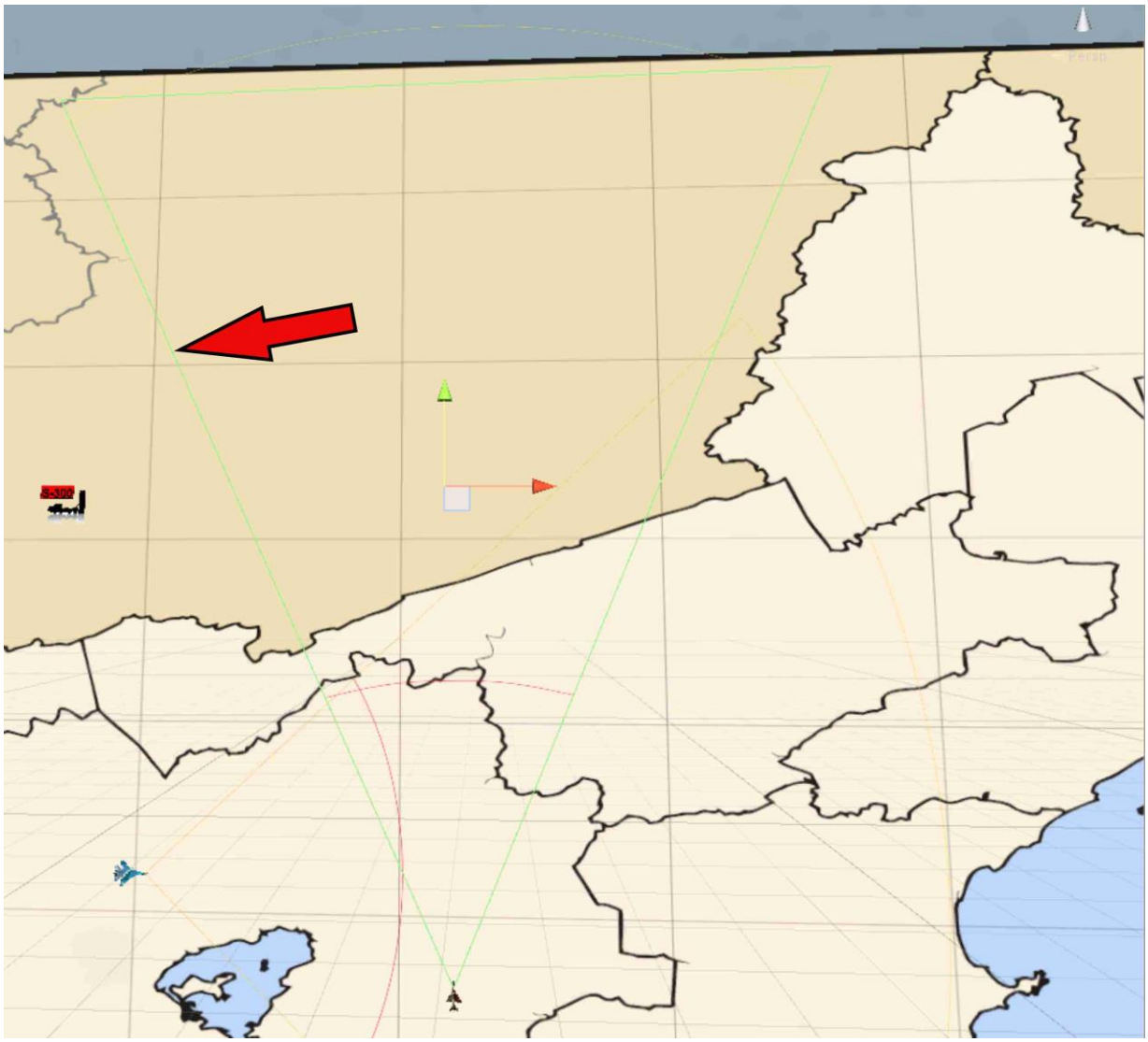


Fonte: Própria, 2020

Os objetos inimigos são um SAM do tipo S-300, fixo, e dois em movimento, um Mig-21 e um Su-30 da Força Aérea da Rússia. Esses objetos inimigos lançam mísseis que são objetos em si com cápsulas de colisão. Acoplados aos inimigos em movimento estão objetos em formato de cone, ainda em 2D, que simulam o cone de detecção do radar, e esses cones, respectivamente, contém cápsulas de colisão com o formato do cone, para servir de base para o *script* de detecção do RWR;

Na Figura 12 é possível observar em verde a cápsula de colisão do cone de detecção do inimigo, no caso do Mig-21:

Figura 12: Cone de detecção e sua cápsula de colisão



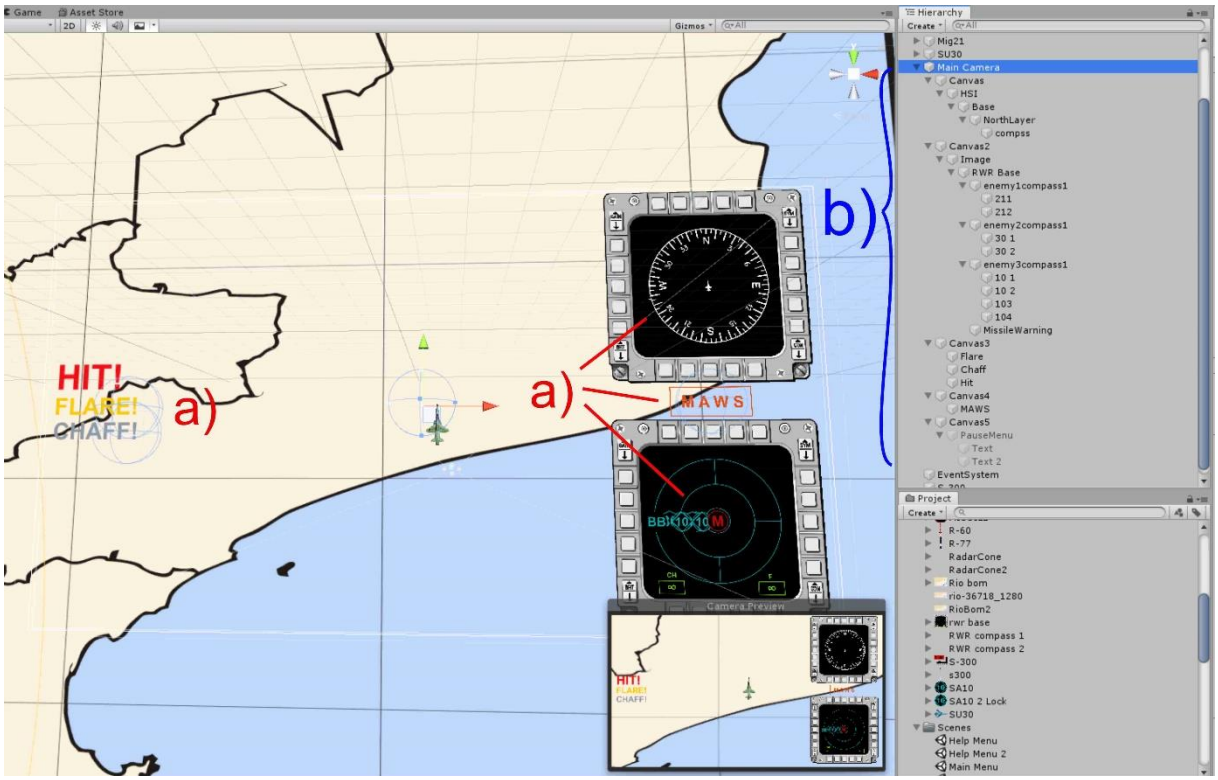
Fonte: Própria, 2020

Acoplados à câmera do jogo estão os objetos que farão parte da simulação dos instrumentos, no HSI tem a base fixa e a bússola que se move, no RWR têm diversos, sendo: uma base fixa, “bússolas” que giram na direção do inimigo (uma para cada inimigo), ou seja, três “bússolas”, dois símbolos de RWR para cada inimigo móvel e quatro para o fixo, e um símbolo para o alerta de mísseis. Também acoplados à câmera estão objetos como o símbolo

para o MAWS (*Missile Approach Warning System*), alertas de lançamento de *flares*, *chaffs*, e *hit* (para quando a aeronave é atingida pelo míssil), e a imagem para o menu de pausa do jogo.

Na Figura 13 é possível perceber os diferentes objetos relativos à câmera do jogo (a), e na janela no canto superior direito está a lista de hierarquia (*Hierarchy*) de objetos (b), com a câmera principal (*Main Camera*) selecionada.

Figura 13: Objetos subordinados à câmera



Fonte: Própria, 2020

Diversos outros objetos estão presentes no cenário principal e de Menu para o funcionamento do jogo.

3.3.3 *Scripts*

O jogo utiliza *scripts* baseados em outros tipos de jogos. Como o jogo é em *top-down*, e os controles da aeronave permitidos são apenas para um lado e outro, e diminuir e aumentar a velocidade, foi escolhido basear-se em um jogo do tipo "*Space Shooter*". O próprio *Unity* contém um tutorial para criar esse tipo de jogo sem custos adicionais. Diferente do "*Space Shooter*" convencional, no simulador de RWR a aeronave controlada mantém-se em constante

movimento à frente, podendo ser apenas modificada sua velocidade e sua direção, guinando no sentido horário ou anti-horário. Existe no *script* do jogador um limite para na movimentação o mesmo não ultrapassar o canto do cenário.

Para os inimigos em movimento, o *script* é modificado para manter os mesmos em movimento circular constante.

A parte mais complicada para o *script*, e também o objetivo principal do jogo, foi a parte de simular o comportamento do RWR. O símbolo que aparece no RWR deve deslocar-se como uma bússola pelo *display* "apontando" para o alvo que representa, considerando a parte de cima do instrumento como se fosse o nariz da aeronave. Em diversos jogos, existe um tipo de "bússola" bastante utilizada, chamada de "bússola de missão" ou *Mission Compass*, que indica a direção para objetos de importância no jogo, e está presente em diferentes tipos de jogos. Foi utilizado um *script* baseado nessa *Mission Compass*, porém, foi necessário fazer algumas modificações como descrito abaixo.

O *script* da *Mission Compass* utilizado foi retirado de um jogo em 3D com visão em primeira pessoa do jogador. Dessa forma foi necessário modificar as fórmulas para que os eixos tanto da visão quanto do giro da "bússola" fossem coincidir com o jogo em *top-down*. Além disso, cada inimigo tem uma *Compass* específica para sempre girar apontando para o mesmo.

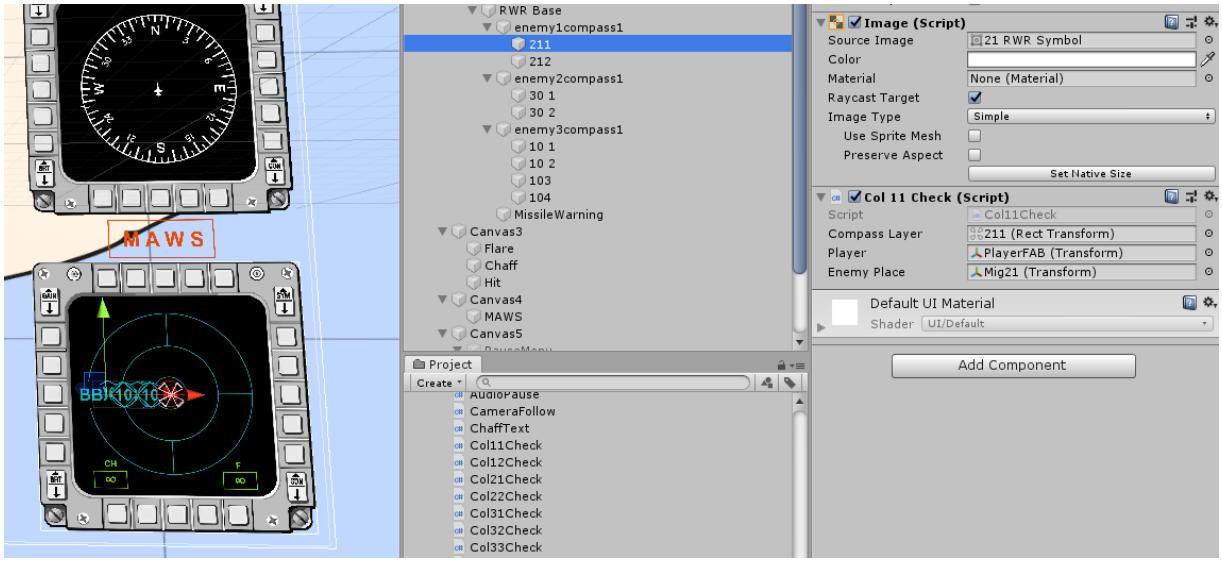
As "bússolas" do RWR são invisíveis, mas acopladas a elas estão os símbolos do RWR que representam os inimigos. Cada símbolo, sendo dois para os inimigos móveis e quatro para o fixo, está preso em uma ponta da "bússola" correspondente, e gira para o lado contrário da mesma para que sempre esteja legível pelo jogador. Cada símbolo contém um *script* que especifica o momento que estará visível e a maneira como foi feito será explicado a seguir.

Para o inimigo fixo, que representa um S-300, não foi necessário um cone para simular o radar porque simula um radar giratório, então o símbolo estaria sempre presente no RWR independentemente da posição do jogador no cenário. Porém, foram criados quatro símbolos para os diferentes estados do radar conforme a aeronave se aproximasse do inimigo, pois o sinal ficaria cada vez mais forte e também o S-300 travaria seu radar no jogador. Baseado então na distância foi criado no *script* regras que fariam com que em cada faixa de distância estivessem aparecendo um dos quatro símbolos do S-300 e os outros três estivessem invisíveis.

As fórmulas desenvolvidas foram mais complexas para os inimigos móveis. Primeiramente foi feito da mesma forma que o fixo, baseado em distância um dos dois símbolos estaria presente no RWR para cada inimigo. Após isso foi criada uma fórmula para fazer com que esses símbolos só pudessem aparecer no *display* se o cone do radar estivesse atingindo a aeronave do jogador, e para isso foram utilizadas as regras do *Unity* para cápsulas de colisão.

A Figura 14 apresenta um dos símbolos do RWR selecionados, podendo ser visto na janela à direita que há um *script* (Col 11 Check), o qual verifica se o objeto deve ou não aparecer no RWR, de acordo com a distância e se está dentro do cone de detecção. Na mesma janela é possível verificar os três objetos que estão subordinados ao *script*: o jogador, o inimigo a que o símbolo pertence (Mig-21), e o símbolo em si. A Figura 15 apresenta o *script* em si:

Figura 14: Símbolo do RWR no *Unity*



Fonte: Própria, 2020

Figura 15: *Script* de “detecção” do RWR

```
Col11Check.cs
Assets > Scripts > Col11Check.cs
using System.Collections.Generic;
using UnityEngine;

public class Col11Check : MonoBehaviour
{
    float distance;
    Vector3 AngleToRotate;
    public RectTransform compassLayer;
    public Transform Player;
    public Transform enemyPlace;
    private int col;


    void LateUpdate()
    {
        distance = Vector3.Distance (Player.transform.position, enemyPlace.transform.position);
        col = GameObject.Find("RadarCone").GetComponent<Enemy1Col>().enm1Col;
        colCheck1();
    }

    public void colCheck1(){
        if(col == 0 || distance < 20f){
            AngleToRotate = new Vector3 (0, 90, 0);
           compassLayer.localEulerAngles = AngleToRotate;
        }
        else
            return;
    }
}
```

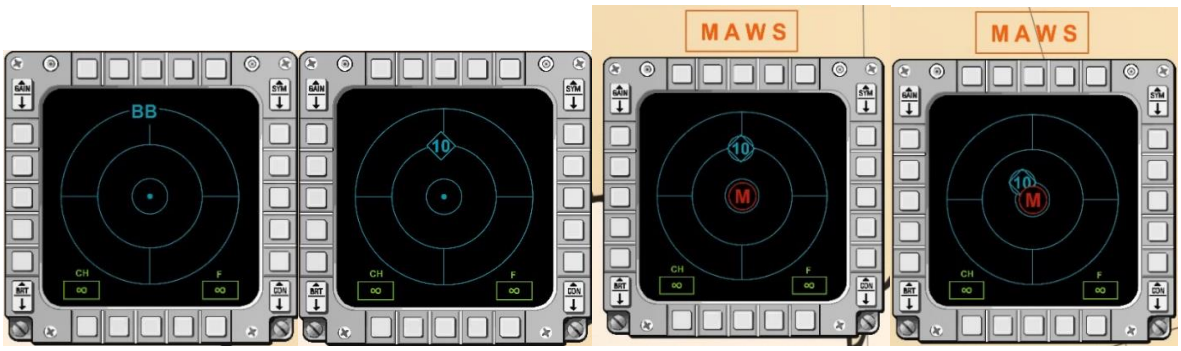
Fonte: Própria, 2020

A Figura 16 é uma imagem retirada do jogo, e nela o RWR, no canto inferior direito, apresenta os três inimigos presentes na tela, e é também possível ver os cones de detecção das aeronaves inimigas. A Figura 17 demonstra a mudança no símbolo e na indicação de intensidade (aproximação do símbolo ao centro) representando o inimigo fixo (S-300), além de apresentar o símbolo do MAWS e do alerta de míssil do RWR, representado por um “M”.

Figura 16: RWR em funcionamento no jogo

Fonte: Própria, 2020

Figura 17: Aumento da intensidade do sinal no RWR



Fonte: Própria, 2020

Os mísseis em si são objetos criados a partir dos *scripts* presentes nos inimigos, simulando o lançamento dos mesmos pelos inimigos. Para o fixo, baseado apenas na distância. Para os móveis, os mísseis são criados baseado na distância e também, se o cone do radar está "colidindo" com a cápsula de colisão do jogador. O *script* acoplado aos mísseis aumenta a velocidade dos mesmos de maneira constante, e gira o mesmo para que sempre esteja apontando para o jogador, simulando dessa forma o comportamento de perseguição dos mísseis. Se a cápsula de colisão acoplada ao míssil atinge a cápsula do jogador, o míssil desaparece, e surge na tela por alguns segundos um objeto, que é uma imagem escrita "*Hit*", para representar que o jogador foi atingido. Se, durante o trajeto do míssil, o jogador apertar a tecla "C", para *chaff*, ou "F", para *flare*, o míssil poderá desaparecer dependendo do tipo, e aparecerá na tela por alguns segundos uma imagem correspondente a *chaff* ou *flare*.

No momento em que qualquer um dos mísseis aparece no jogo, surge na tela do jogador o alerta do MAWS, aparecendo também o alerta de míssil do próprio RWR caso o míssil seja guiado por radar.

Os mísseis lançados pelo Mig-21 no jogo representam mísseis guiados por infravermelho, ou seja, no jogo, se o jogador lançar *flares* o míssil desaparece, sendo "despistado". Além disso, no RWR, não aparece o alerta de lançamento de míssil porque na realidade não apareceria, pois mísseis infravermelhos têm guiamento passivo, e por isso não emitem ondas de radar detectáveis pelo RWR. Aparece apenas o alerta MAWS.

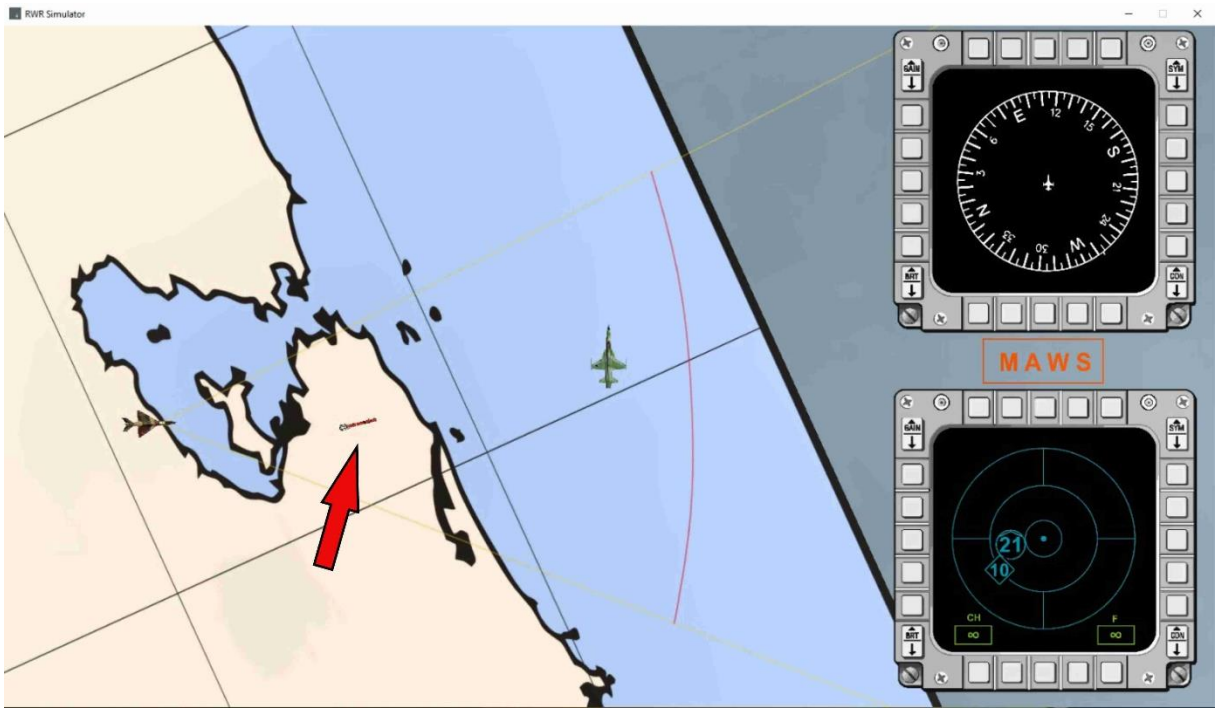
Os mísseis lançados pelo Su-30 representam no jogo os mísseis guiados por radar ativo. Dessa forma, são "despistados" pelo jogador que acionar o *chaff*. Assim que são lançados, surge o alerta MAWS, porém só surgirá o alerta de míssil do RWR se a distância for menor que a especificada no *script*. Isso visa simular o comportamento desse tipo de míssil na

realidade, pois o míssil só ativa seu radar de guiamento quando estiver bem próximo do alvo em si, sendo até lá guiado pelo radar da própria aeronave, só aparecendo o sinal de alerta de míssil no RWR quando o mesmo ativa seu radar nos momentos finais da trajetória. Existem, na realidade, muitos fatores tanto do míssil quanto do radar da aeronave para determinar que esse será o comportamento dos instrumentos, porém, para fins de simplificar o jogo, foi escolhido basear-se em um cenário geral.

Os mísseis do S-300, o inimigo fixo, representam no jogo os mísseis de guiamento semiativo. Esse tipo de bateria de SAM, em sua versão real, utiliza um tipo de guiamento não convencional, porém não o foi implementado no jogo para poder demonstrar o funcionamento de um míssil semiativo no RWR. O míssil, dessa forma, é guiado pelo radar da própria estação que o lançou, nesse caso o S-300, e por isso o RWR indica seu lançamento desde o momento inicial até o impacto.

Na Figura 18 é observado um míssil no jogo, vermelho, lançado em direção ao jogador. O míssil da imagem é do tipo guiado por infravermelho, que é usado pelo inimigo "Mig-21", e, por esse motivo, o RWR não apresenta o símbolo de alerta de míssil, apenas o alerta do MAWS dispara, de maneira a simular o comportamento real desse tipo de míssil:

Figura 18: Míssil guiado por infravermelho



Fonte: Própria, 2020

Existem, tanto no cenário principal quanto no “sem mísseis”, objetos, um para cada cenário, invisíveis, que estão lá para o funcionamento da interface do jogo. Neles há um *script* que, quando o jogador pressiona a tecla “P”, o jogo pausa, e isso é feito transformando o tempo do jogo, que é usado como base para as animações, em nulo, além de interromper qualquer efeito de som e mostrar uma tela diferente no jogo. Esses objetos também permitem comandos para sair do jogo e ir para o Menu principal.

Os *Scripts* de maior importância do jogo estarão disponíveis em apêndice neste trabalho.

3.3.4 Imagens e figuras

As imagens importadas no jogo para as texturas dos objetos foram feitas no programa Corel Draw, utilizando em alguns casos como base imagens retiradas da web. Foi por vezes criada a imagem do zero, e por outras, utilizando uma imagem como base. Para tanto foi feito um rastreio de contorno da imagem, transformando a mesma em *Bitmap*, através de uma função do Corel Draw. Dessa maneira foi possível modificar tais imagens a fim de adequá-las ao jogo. As figuras finais utilizadas no jogo estavam em formato PNG.

3.3.5 Efeitos sonoros

Os efeitos sonoros utilizados no jogo desenvolvido foram arquivos em formato MP3 retirados do jogo *Falcon BMS* 4.34[1], incluindo efeitos para o som dos motores da aeronave do jogador, alerta de aproximação de míssil, alerta de trava de radar, lançamento de *chaff* e *flare*, aeronave atingida (*hit*) e sons da interface do jogo.

Scripts em diversos objetos chamam para si os arquivos de som para diferentes situações, e todos são silenciados quando o jogador coloca o jogo em pausa.

3.4 Literatura

Para o correto funcionamento do RWR e dos outros instrumentos e equipamentos no jogo, foram utilizadas diferentes fontes. A principal fonte para a visualização da simulação foi do jogo/simulador de combate DCS (*Digital Combat Simulator*), e seus respectivos manuais,

[1] O jogo *Falcon BMS* 4.34 é um mod, e não pode ser usado com propósitos comerciais de qualquer espécie, segundo os Termos de Condição da *Benchmark Simulations*, porém o *software* está disponível para o público. (BENCHMARKSIMS, acesso em 2020)

principalmente o manual da aeronave F-16 simulado no mesmo. Também foram consultados o Manual de Guerra Eletrônica do COMGAR e o livro de David Adamy, *Introduction to Electronic Warfare Modeling and Simulation* (Introdução à modelagem e simulação de Guerra Eletrônica – Tradução própria).

Foi possível verificar, durante a pesquisa para o projeto, a falta de material disponível que permita visualizar o funcionamento desses equipamentos de GE, mesmo um considerado mais simples como o RWR. Também foi verificado o quanto o aprendizado é superior ao utilizar de fontes como jogos e simuladores aliados aos manuais, em oposição a apenas leitura de artigos e manuais. Essa seria novamente a justificativa para o desenvolvimento do jogo deste trabalho, ou seja, facilitar o aprendizado sobre os equipamentos simulados.

4 CONSIDERAÇÕES FINAIS

O Clube de Guerra Eletrônica e Cibernética da AFA reconhece a importância do aprendizado antecipado sobre o tópico para os futuros oficiais da Força Aérea Brasileira, e, sendo o RWR um importante, porém, complexo equipamento nessa área, viu-se a possibilidade de introduzir o mesmo para os membros do Clube de uma maneira mais eficaz se comparado aos estudos tradicionais. A escolha de se desenvolver um jogo foi para poder cumprir essa tarefa, simultaneamente não restringindo-se apenas à teoria como também não adentrando nas complexidades de seu funcionamento, sendo apenas para conhecimento simples, introdutório, de nível operador.

Dessa forma, visando o objetivo do CGEC, foi feito o desenvolvimento de um jogo simples de simulação de RWR, que demonstra de maneira básica o comportamento do sistema na visão do piloto, sem entrar em detalhes de seu funcionamento. O jogo contém explicações dentro e fora do mesmo, tanto de sua jogabilidade quanto das teorias envolvidas sobre os equipamentos simulados, incluindo páginas em inglês e em português. O jogo de simulação de RWR teve seu desenvolvimento iniciado em junho de 2019, com sua finalização em setembro do mesmo ano.

Para o cumprimento do objetivo deste trabalho, foi primeiramente apresentado a revisão bibliográfica, contendo o histórico da Guerra Eletrônica e do equipamento em questão, a fim de apresentar a importância do tópico, principalmente em se tratando de Força Aérea. Foi visto que a Guerra Eletrônica é uma área da guerra que evolui constantemente, e está fortemente atrelada ao sucesso ou fracasso em operações, batalhas, e até mesmo guerras. Em relação ao RWR, o *Radar Warning Receiver*, que teve seu uso expandido a partir de seu papel na Guerra do Vietnã, foi apresentado a teoria por trás de seu funcionamento e de seu comportamento. Além disso foi também visto a teoria por trás de algumas tecnologias utilizadas em aeronaves de combate, como guiamento de mísseis, que seriam importantes para a simulação do cenário de combate no jogo.

Em seguida, foi visto a criação em si do jogo. Primeiro foram apresentados os objetivos do Clube de Guerra Eletrônica da AFA, e como, para o cumprimento desses, a introdução a um equipamento de tamanha importância para a guerra aérea poderia ser feita através de um jogo simples. Logo em seguida foi analisada a estrutura por trás do jogo, e sua lógica de funcionamento, que passaria para os *scripts* dentro do motor de jogo, para prover a mecânica do mesmo. Através dessa análise foi possível perceber o quanto a criação de um

jogo é uma série de solução de problemas, junto com criatividade, buscando soluções em outras fontes, como jogos e mecânicas já existentes. Foram apresentadas as teorias por trás dos equipamentos simulados, assim como a maneira como foi adaptada o comportamento dos mesmos da vida real para o motor de jogo, de forma a simplificar o máximo, mantendo a base da simulação no comportamento do RWR. O simulador de RWR do Clube de Guerra Eletrônica da AFA é, então, capaz de demonstrar o funcionamento do equipamento, a nível operador e introdutório, em um cenário de guerra aérea.

Por fim, com o jogo criado e funcionando corretamente, o objetivo final do trabalho foi concluído com êxito, disponibilizando o *software* para os membros do Clube para o aprendizado e incentivo à pesquisa na área da Guerra Eletrônica. Demonstrou-se, então, a viabilidade de soluções providas por desenvolvimento de *software*, por sua capacidade de foco em objetivos específicos, e variedade de custo e possibilidade de desenvolvimento, desde o nível amador para uso pessoal até o nível profissional.

REFERÊNCIAS:

ADAMY, David L. **Introduction to Electronic Warfare Modeling and Simulation**. Raleigh, NC, EUA: SciTech, 2006

KOPP, Carlo. Radar Warning Receivers and Defensive Electronic Countermeasures. **Australia Aviation**. Setembro 1988 – 2005. Disponível em: <https://www.ausairpower.net/TE-RWR-ECM.html>. Acesso em: 11 Jan 2020

NALTY, Bernard. **Tactics and Techniques of Electronic Warfare in Vietnam**. Newtown, CT, EUA: Defense Lion, 2013

COMGAR. **Manual de Guerra Eletrônica**. Brasília, DF: Centro de Guerra Eletrônica, 2013

SCREESHOTS. **Digital Combat Simulator**, 2020
Disponível em: <https://www.digitalcombatsimulator.com/en/downloads/screenshots/1201/ >. Acesso em: 13 abr. 2020.

WHAT IS A GAME ENGINE. **Unity 3d**, 2020
Disponível em:
<https://unity3d.com/pt/what-is-a-game-engine >. Acesso em: 20 abr. 2020.

WORLD. **Digital Combat Simulator**, 2020
Disponível em:
<https://www.digitalcombatsimulator.com/en/products/world/>. Acesso em: 20 abr. 2020.

FREQUENTLY ASKED QUASTIONS. **Benchmarksims**, 2020
Disponível em:
<https://www.benchmarksims.org/forum/content.php?131&s=1c82748eff06cfc60635fbfd6b7dced3>. Acesso em: 20 abr. 2020.

SUPERBUG. **VRSimulations**, 2020
Disponível em:
<https://www.vrsimulations.com/superbug.php>. Acesso em: 20 abr. 2020.

Estatuto do Clube de Guerra Eletrônica e Cibernética 2020 – CCAER (Corpo de Cadetes da Aeronáutica - AFA). **Clube de Guerra Eletrônica e Cibernética** – AFA, 2020.

LABGE - LABORATÓRIO DE ENSINO EM RADAR E RF, **Laboratório de Guerra Eletrônica**, 2020
Disponível em:
<http://www.ele.ita.br/~labge/index_arquivos/ge_radar.htm>. Acesso em: 09 ago. 2020.

APÊNDICE A – *Scripts* do HSI e RWR

O *Script* “HSI” possui o código para o funcionamento do HSI e do RWR, pois realiza a movimentação das bússolas, ou *compass*, que apontam para o “Norte” do cenário, no caso do HSI, ou para os respectivos inimigos do jogo, no caso do RWR. Os *Scripts* tipo ColxxCheck, estão associados aos diferentes símbolos do RWR, e fazem o papel de deixar os mesmos aparecerem apenas quando os parâmetros são atingidos, como distância ao inimigo, e se há ou não colisão entre o jogador “*player*” e o cone de detecção.

HSI ----------

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class HSI : MonoBehaviour
{
   public Vector3 NorthDirection;
   public Transform Player;
   public Transform enemyPlace;
   public Transform enemy2Place;
   public Transform enemy3Place;

   public RectTransform NorthLayer;
   public RectTransform Enemy1Layer;
   public RectTransform Enemy2Layer;
   public RectTransform Enemy3Layer;
   public RectTransform Compass211Layer;
   public RectTransform Compass212Layer;
   public RectTransform Compass301Layer;
   public RectTransform Compass302Layer;
   public RectTransform Compass101Layer;
   public RectTransform Compass102Layer;
   public RectTransform Compass103Layer;
   public RectTransform Compass104Layer;

     void Update()
   {
      ChangeNorthDirection();
      ChangeEnemy1Direction();
      ChangeEnemy2Direction();
      ChangeEnemy3Direction();
   }
```

```
  public void ChangeNorthDirection(){
     NorthDirection.z = -Player.eulerAngles.z;
     NorthLayer.localEulerAngles = NorthDirection;
  }

public void ChangeEnemy1Direction(){
    Vector2 dir = transform.position - enemyPlace.position;
    float enmDirection = Mathf.Atan2(dir.y, dir.x) * Mathf.Rad2Deg;
    Enemy1Layer.localEulerAngles = new Vector3(0, 0, (enmDirection - Player.eulerAngles.z));
    Compass211Layer.localEulerAngles = -Enemy1Layer.localEulerAngles;
    Compass212Layer.localEulerAngles = -Enemy1Layer.localEulerAngles;
 }

  public void ChangeEnemy2Direction(){
    Vector2 dir = transform.position - enemy2Place.position;
    float enmDirection = Mathf.Atan2(dir.y, dir.x) * Mathf.Rad2Deg;
    Enemy2Layer.localEulerAngles = new Vector3(0, 0, (enmDirection - Player.eulerAngles.z));
    Compass301Layer.localEulerAngles = -Enemy2Layer.localEulerAngles;
    Compass302Layer.localEulerAngles = -Enemy2Layer.localEulerAngles;
 }

   public void ChangeEnemy3Direction(){
    Vector2 dir = transform.position - enemy3Place.position;
    float enmDirection = Mathf.Atan2(dir.y, dir.x) * Mathf.Rad2Deg;
    Enemy3Layer.localEulerAngles = new Vector3(0, 0, (enmDirection - Player.eulerAngles.z));
    Compass101Layer.localEulerAngles = -Enemy3Layer.localEulerAngles;
    Compass102Layer.localEulerAngles = -Enemy3Layer.localEulerAngles;
    Compass103Layer.localEulerAngles = -Enemy3Layer.localEulerAngles;
    Compass104Layer.localEulerAngles = -Enemy3Layer.localEulerAngles;
 }

}
```

ColxxCheck----------

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Col11Check : MonoBehaviour
{
   float distance;
   Vector3 AngleToRotate;
   public RectTransform compassLayer;
   public Transform Player;
   public Transform enemyPlace;
   private int col;

      void LateUpdate()
```

```
    {
        distance = Vector3.Distance (Player.transform.position, enemyPlace.transform.position);
        col = GameObject.Find("RadarCone").GetComponent<Enemy1Col>().enm1Col;
        colCheck1();
    }

    public void colCheck1(){
        if(col == 0 || distance < 20f){
            AngleToRotate = new Vector3 (0, 90, 0);
           compassLayer.localEulerAngles = AngleToRotate;
        }
        else
            return;
    }
}
```

APÊNDICE B – *Scripts* de movimento dos objetivos

Os *Scripts* a seguir estão associados ao jogador, aos inimigos móveis, e aos mísseis, e permitem os movimentos dos mesmos pelo cenário.

EnemyX ----------

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Enemy1 : MonoBehaviour
{
   public float rotSpd = 8f;
   public float maxSpd = 1f;

   void Update()
   {
      Quaternion rot = transform.rotation;
      float z = rot.eulerAngles.z;
      z += rotSpd * Time.deltaTime;
      rot = Quaternion.Euler( 0, 0, z );
      transform.rotation = rot;

      Vector3 pos = transform.position;
      Vector3 velocity = new Vector3(0, maxSpd * Time.deltaTime, 0);
      pos += rot * velocity;
      transform.position = pos;
   }
}
```

PlayerController ----------

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerController : MonoBehaviour
{
  public float rotSpeed = 45f;
  public float maxSpeed = 2f;

   void Start()
   {
   }

   void changeSpeed()
```

```
    {
        if(Input.GetKey(KeyCode.W))
            maxSpeed+=0.2f;
        if(Input.GetKey(KeyCode.S))
            maxSpeed-=0.2f;

        if(maxSpeed>10f)
            maxSpeed=10f;
        if(maxSpeed<0.6f)
            maxSpeed=0.6f;
    }

    void Update()
    {
        changeSpeed();
        Quaternion rot = transform.rotation;
        float z = rot.eulerAngles.z;

        if(Input.GetKey(KeyCode.A))
        {
            z += rotSpeed * Time.deltaTime;
        }

        if(Input.GetKey(KeyCode.D))
        {
            z -= rotSpeed * Time.deltaTime;
        }

        rot = Quaternion.Euler( 0, 0, z );
        transform.rotation = rot;

        Vector3 pos = transform.position;
        Vector3 velocity = new Vector3(0, maxSpeed * Time.deltaTime, 0);
        pos += rot * velocity;

        pos.x = Mathf.Clamp(pos.x, -75, 75);
        pos.y = Mathf.Clamp(pos.y, -37, 37);
        transform.position = pos;
    }
}
```

XXX_Control (Mísseis) ----------

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class R60_Control : MonoBehaviour
{
```

```
  GameObject go;
  Transform player;
  float mslSpeed = 3f;

  void Start()
  {
     go = GameObject.Find ("PlayerFAB");
  }

   void Update()
   {
     mslSpeed += 3f * Time.deltaTime;
     if(mslSpeed>11f){
        mslSpeed=11f;
     }

     player = go.transform;

     Vector3 dir = player.position - transform.position;
     dir.Normalize();
     float zAngle = Mathf.Atan2(dir.y, dir.x) * Mathf.Rad2Deg - 90;
     transform.rotation = Quaternion.Euler(0, 0, zAngle);

      Vector3 pos = transform.position;
      Vector3 velocity = new Vector3 (0, mslSpeed * Time.deltaTime, 0);
      pos += transform.rotation * velocity;
      transform.position = pos;

      if(Input.GetKeyDown(KeyCode.F))
         Destroy(gameObject);
   }

   void OnTriggerStay2D(Collider2D col){
      if(col.gameObject.tag == "Player"){
         Destroy(gameObject);
      }
   }
}
```

APÊNDICE C – *Scripts* de detecção e lançamento de mísseis

Os *Scripts* EnemyxCol estão associados aos objetos que são os cones de detecção dos inimigos, sendo que os mesmos possuem uma cápsula de colisão para simular essa detecção através da colisão do cone com o jogador. Os *Scripts* EnemyxShooting fazem surgir, ou "instanciar", os objetos pré-fabricados que são os mísseis, baseado na distância com o jogador e se estiver ocorrendo a colisão, além de haver uma função de tempo, para que os lançamentos dos mísseis tenham um período de segundos entre si. Além disso o código cria um limite de dois mísseis para cada inimigo.

EnemyxCol ----------

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Enemy1Col : MonoBehaviour
{
   public int enm1Col = 0;

   public void OnTriggerEnter2D(Collider2D col)
   {
      if(col.gameObject.tag == "Player"){
         enm1Col = 1;
      }
      else{
         enm1Col = 0;
      }
   }

   public void OnTriggerExit2D(Collider2D col){
      enm1Col = 0;
   }
}
```

EnemyxShooting ----------

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Enemy1Shooting : MonoBehaviour
{
  public GameObject missilePrefab;
```

```
  float distance;
  int mslCount = 2;
  float fireDelay = 8;
  float cooldownTimer = 0;
  int col;
  public Transform Player;
  public Transform enemyPlace;

   void Update()
   {
     cooldownTimer -= Time.deltaTime;
     if(cooldownTimer<0){
        cooldownTimer=0;
     }
      distance = Vector3.Distance (Player.transform.position, enemyPlace.transform.position);
      col = GameObject.Find("RadarCone").GetComponent<Enemy1Col>().enm1Col;
      if(col == 1 && distance <20f && cooldownTimer==0 && mslCount>0){
          cooldownTimer = fireDelay;
          mslCount --;
          Instantiate(missilePrefab, enemyPlace.transform.position, enemyPlace.transform.rotation);
      }
   }
}
```

APÊNDICE D – *Scripts* dos objetos da câmera

O *Script* CameraFollow está associado à câmera do jogo, e faz com que a câmera siga o jogador, além de conter os códigos para que o jogador mude o zoom, ou mesmo mude o modo de câmera, que seriam os dois disponíveis: em um deles, a câmera estará sempre orientada com o "Norte" do cenário, e no outro modo a câmera estará sempre orientada pela direção da aeronave do jogador. Os arquivos tipo XXXText, são usados no aparecimento do texto de *chaff*, *flare*, e *hit*, e funcionam com certos gatilhos para ativar, como apertar teclas, ou quando algum míssil atinge a aeronave. Além disso, o código também aciona o respectivo efeito sonoro. O *Script* MAWS prevê o aparecimento do símbolo do alerta sempre que um míssil surgir no jogo.

CameraFollow ----------

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CameraFollow : MonoBehaviour
{
   public int cameraMode = 0;
   float camSize=7f;

   void ModeChange(){
      if (Input.GetKeyDown(KeyCode.Space)){
         if(cameraMode==0)
            cameraMode=1;
            else
            {
               cameraMode=0;
            }
      }
   }

   void zoomChange(){
      if(Input.GetKey(KeyCode.UpArrow))
         camSize-=0.2f;
      if(Input.GetKey(KeyCode.DownArrow))
         camSize+=0.2f;

      if(camSize>18f)
         camSize=18f;
      if(camSize<6f)
         camSize=6f;
```

```
        Camera.main.orthographicSize=camSize;
    }
    public Transform myTarget;

        void Update()
    {
        ModeChange();

        zoomChange();

        if(myTarget != null) {
            Vector3 targPos = myTarget.position;
            targPos.z=transform.position.z;
            transform.position=targPos;
        }

        if(cameraMode==1){
            transform.rotation=myTarget.rotation;
        }
        else
            transform.rotation=Quaternion.identity;
    }
}
```

XXXText ----------

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class ChaffText : MonoBehaviour
{
    float popupDelay = 3;
    float timer = 0;
    Text myText;
    AudioSource myAudio;

     void Start(){
        myText = GetComponent<Text>();
        myAudio = GetComponent<AudioSource>();
     }

     void Update()
     {
       timer -= Time.deltaTime;
       if(timer<0)
        timer=0;
```

```
        if(Input.GetKeyDown(KeyCode.C)){
            timer=popupDelay;
            myAudio.Play();
        }
        if(timer>0)
            myText.enabled = true;

        if(timer==0)
            myText.enabled = false;
    }
}
```

MAWS ----------

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class MAWS : MonoBehaviour
{
    public int mslCheck=0;
    Image myImage;
    GameObject go;
    AudioSource myAudio;

    void Start()
    {
        myImage = GetComponent<Image>();
        myAudio = GetComponent<AudioSource>();
    }


    void Update()
    {
        if(mslCheck == 0)
        {
            go = GameObject.FindWithTag ("Missile");

            if(go != null){
                mslCheck = 1;
            }
        }
        if(go == null){
            mslCheck=0;
        }
    }
```

```
    void LateUpdate(){
        if(mslCheck==1){
            myImage.enabled=true;
            myAudio.enabled=true;
        }
        if(mslCheck==0){
            myImage.enabled=false;
            myAudio.enabled=false;
        }

        if(PauseMenu.GameIsPaused){
            myAudio.enabled=false;
        }
    }
}
```