Especificação de Desenvolvimento do Jogo Tiro Certo

Disciplina: Computação Gráfica
Alunos: Paulo Aragão
Gilmário Santos

Índice

1 Introdução

Este documento tem como objetivo principal detalhar o projeto do Jogo “Tiro Certo”, correspondente ao trabalho na disciplina de Computação Gráfica do 1º semestre de 2006.

2 Ambiente de Desenvolvimento

A linguagem utilizada no desenvolvimento do “Tiro Certo” foi o Java™. Esta escolha foi favorecida pela experiência dos autores na plataforma Java. Além disto, a linguagem Java é orientada a objetos, independente de plataforma e conta com inúmeros CASEs (**C**omputer **A**ided **S**oftware **E**ngineer) de modelagem UML para geração de código e engenharia reversa, possibilitando a agilidade na implementação e geração automática de documentação de desenvolvimento.

A biblioteca utilizada para o OpenGL foi o JOGL. No passado, houve inúmeras tentativas para implementação do OpenGL em Java. No entanto, o JOGL está se tornando padrão neste ambiente, já que sua implementação conta com duas peças importantes: A Sun, a criadora do Java e a SGI (Silicon Graphics), responsável pela especificação do OpenGL.

A ferramenta CASE utilizada foi o Rational Rose™. No ambiente Java, existe inúmeros CASEs disponíveis. Entre eles podemos destacar o Umbrello™ (código livre) e o Jude™ (com versão grátis). Neste projeto, pelas experiências dos autores, se optou pelo Rational Rose, hoje pertencente à IBM.

A geração do cenário em 3D contou com a ajuda do modelador geométrico MilkShape™. Esta ferramenta aceita exportação e importação em diversos formatos. Foi desenvolvido um parser em Java™ para importação do modelo feito no MilkShape e exportado em formato ascii (3dsasc) para o OpenGL.

Todo o desenvolvimento contou com a experiência dos autores em desenvolvimento orientado a objetos, linguagem UML e, sobretudo, pela utilização de uma metodologia de desenvolvimento incremental de acordo com a filosofia do RUP (Rational Unified Process). Além disto, o uso constante de Padrões de Projetos (Design Patterns) contribuiu para a reusabilidade, manutenabilidade e facilidade de extensão.

3 Arquitetura

O desenvolvimento do “Tiro Certo” se baseia em uma arquitetura baseada em camadas. O uso do Java™ permite a organização das camadas baseadas em pacotes (packages). A aplicação possui as seguintes camadas: Apresentação (pacote presentation), controle (pacote controller), negócio (pacote business) e serviços (service). A Figura 1 ilustra a arquitetura do sistema e as setas indicam as dependências entre os pacotes. As seções seguintes detalham cada uma destas camadas.

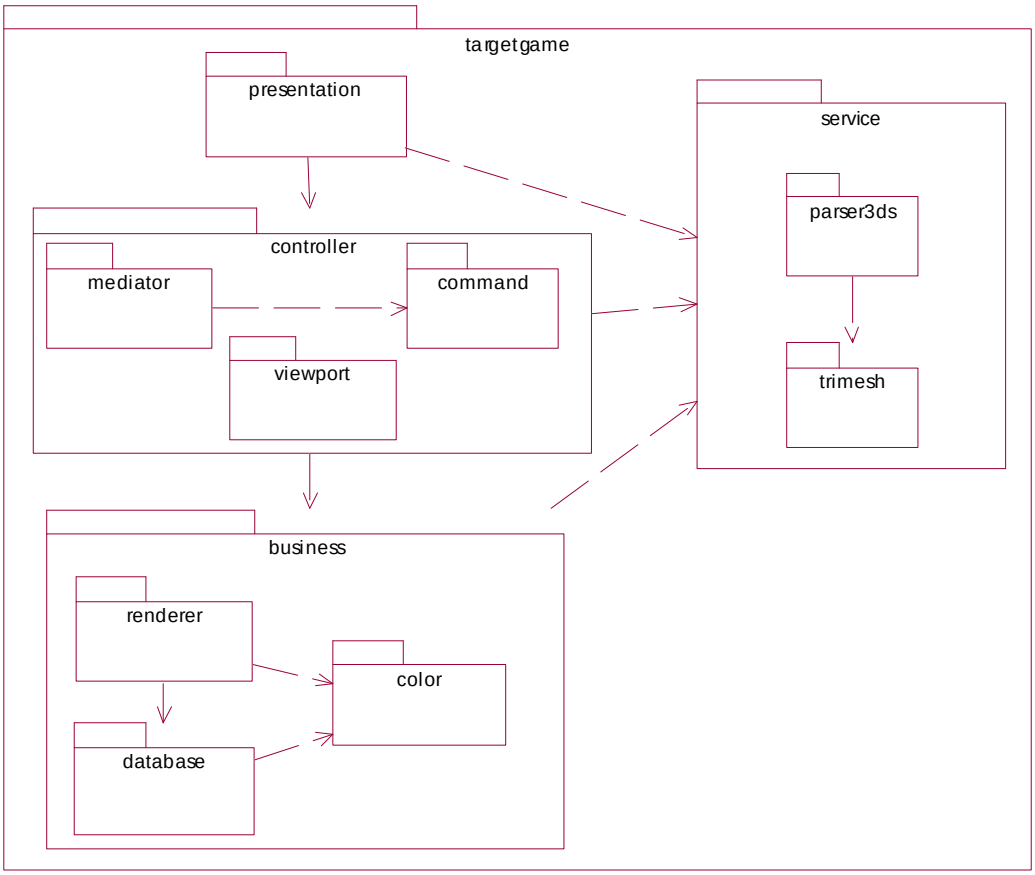

Figura 1 - Arquitetura

3.1 Camada de Apresentação

A camada de apresentação representa a entrada de dados do usuário e visualização dos dados (saída OpenGL). Portanto, este código está relacionado com a interface gráfica do usuário: Janelas, controles gráficos, menus, etc.

As classes desta camada são o *TGMainFrame* e *TGViewFrame* (Figura 2). O *TGMainFrame* representa a janela principal da aplicação. Já o *TGViewFrame* é a janela para a saída gráfica OpenGL. A aplicação "Tiro Certo" pode abrir várias janelas *TGViewFrame*, dando ao usuário uma visão do cenário sob várias perspectivas, o que contribui para um melhor ajuste de ângulo e força de disparo.

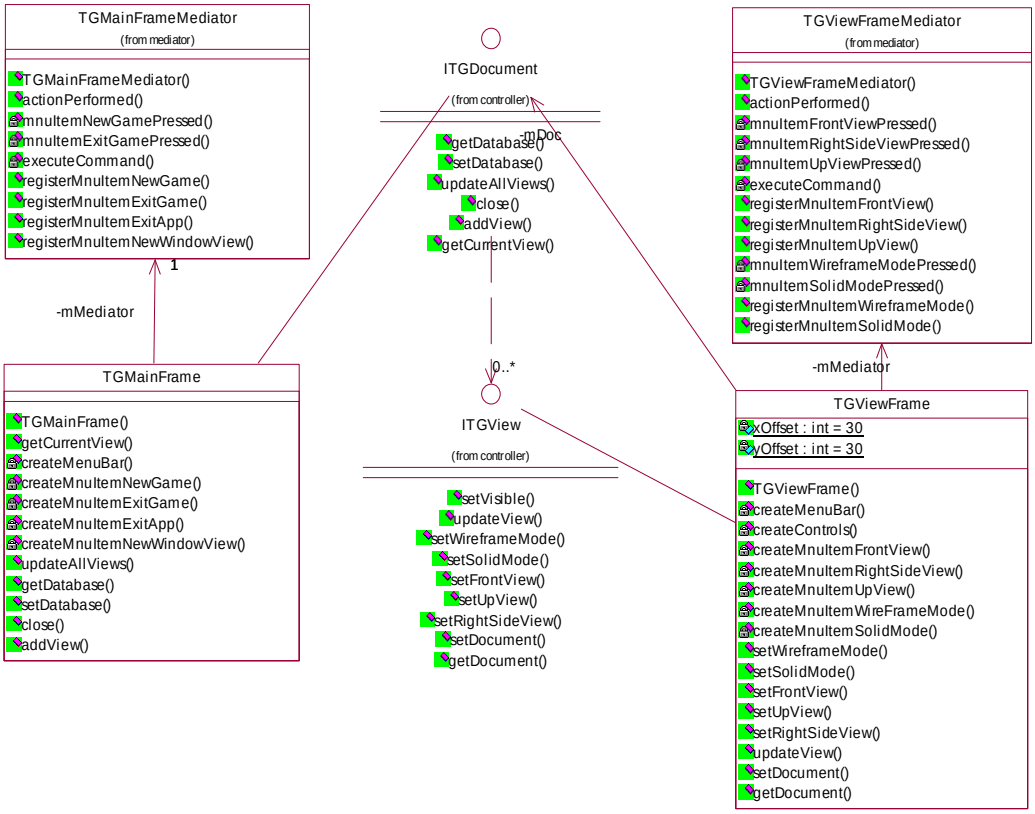

Figura 2 – Camada de Apresentação

3.2 Camada de Controle

A camada de controle segue o padrão MVC (Model-View-Controller) e permite uma clara separação entre as camadas de apresentação e negócio. Tal separação flexibiliza a aplicação quanto a uma futura manutenção do sistema referente a mudanças de interface (swing, awt, swt, etc.), o tipo de aplicação (desktop x browser) e bibliotecas de visualização (OpenGL, java3D). De acordo com a Figura 1, a camada de controle possui três pacotes principais: *mediator*, *command*, *viewport*.

O pacote *mediator* corresponde à implementação dos "mediadores" referente ao padrão de projeto "Mediator" [ ]. O mediador é o responsável por interceptar e tratar os eventos provenientes da interface gráfica. Os mediadores TGMainFrameMediator e TGViewFrameMediator intercepta os eventos provenientes de TGMainFrame e TGViewFrame respectivamente (Figura 2).

O pacote *command* implementa os comandos da aplicação. A implementação deste pacote segue a especificação do padrão de projeto "Command"[ ]. Cada mediador, ao interceptar um determinado evento, executa um dos comandos disponíveis no pacote *command*.

A Figura 3 representa as classes de comando para a aplicação: iniciar novo jogo, finalizar jogo, finalizar aplicação e criar nova janela de visualização.

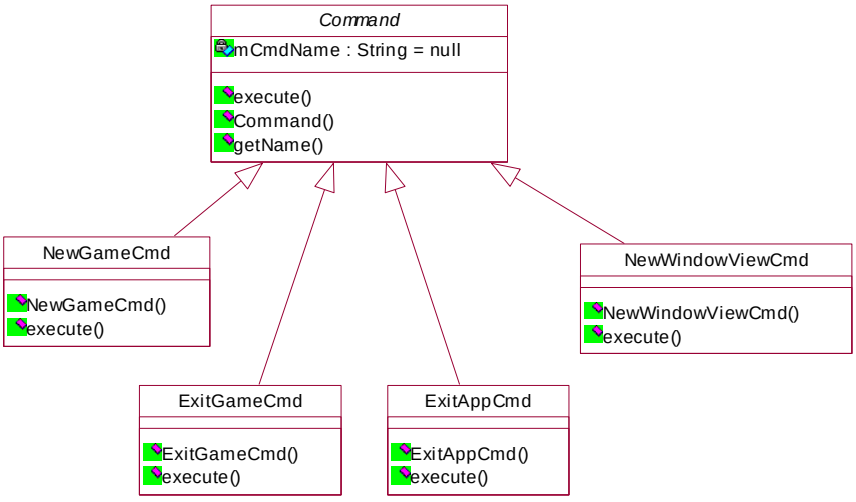

Figura 3 - Comandos da Aplicação

A Figura 4 representa as classes de comando para a visualização. Cada janela pode ver o modelo em dois modos gráficos (sólido e aramado) representados pelas classes *SolidModeCmd* e *WireframeModeCmd*. Além disto, cada janela pode estar configurado sob um determinado ponto de vista (frontal, lateral direita e superior) representados pelas classes *FrontViewCmd*, *RightSideViewCmd* e *UpViewCmd*.

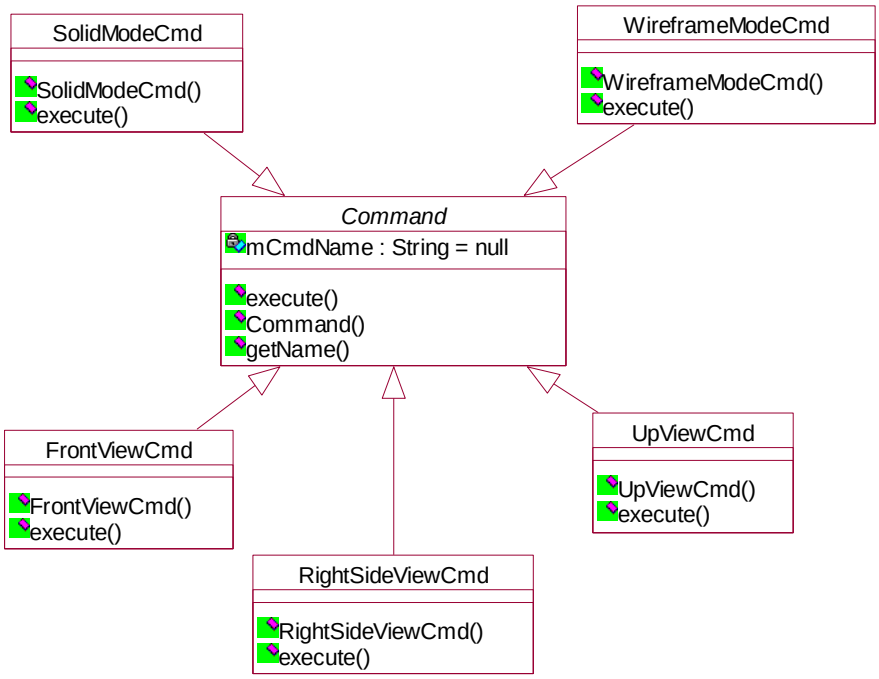

Figura 4 – Comandos para Visualização

A Figura 5 representa as classes de comando para disparo do tiro. Estas classes correspondem aos comandos para incremento e decremento do ângulo de disparo (vertical e horizontal) e ajuste da força de disparo.

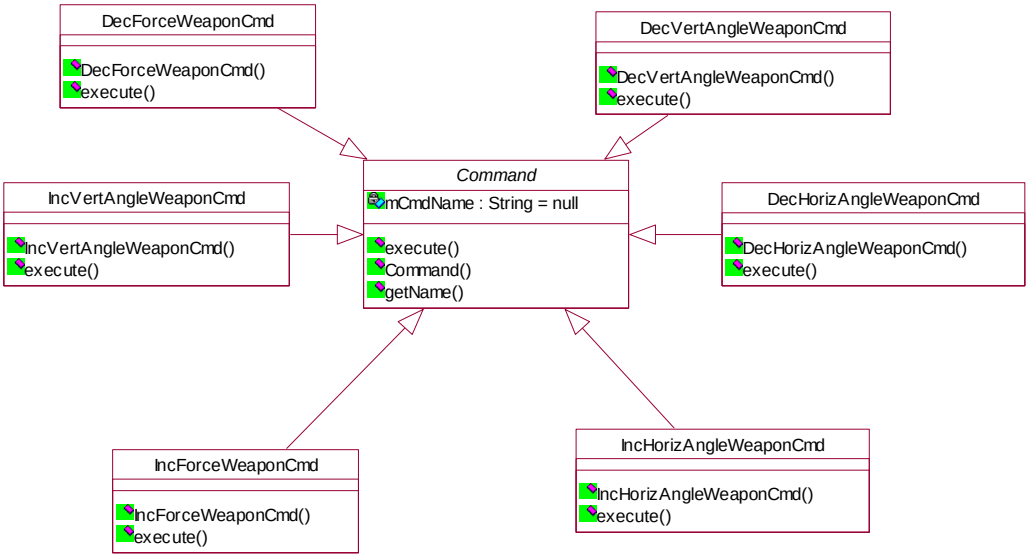

Figura 5 – Comandos para Disparo

A camada de controle também disponibiliza duas interfaces: a interface *ITGDocument* e *ITGView*. Estas interfaces são implementadas pelas classes *TGMainFrame* e *TGViewFrame* respectivamente (Figura 2). Tais interfaces se referem ao padrão "Observer" [ ], garantido ter várias visões referente a um documento. Esta implementação permite a abertura de várias janelas gráficas, onde cada uma delas pode ter uma visão em particular: lateral direita, superior, frontal, considerando ainda os modos de visualização sólido ou aramado.

3.3 Camada de Negócio

A camada de negocio se refere ao domínio do problema. De acordo com a Figura 1, esta camada possui os pacotes *color*, *database* e *renderer*.

De acordo com a Figura 6, o pacote *database* se refere às classes que representa o banco de dados da aplicação. A classe *TGDatabase* tem uma referência para a classe *TGEntityTable*. A classe *TGEntityTable* representa uma tabela onde são armazenadas as entidades gráficas (classes *TGEntity*).

A classe TGEntityTable apresenta uma referência para um iterator, a classe *TGEntityTableIterator*, que permite fazer uma iteração em todas as entidades gráficas armazenadas. O Iterator é um padrão de projeto especificado em [ ].

Figura 6 – Banco de Dados da Aplicação

A classe *TGEntity* representa uma entidade gráfica. No caso da aplicação "Tiro Certo", as entidades são representadas pelas construções (classe *TGInfraStructure*), alvo (classe *TGTarget*), arma (classe *TGWeapon*), projétil (classe *TGProjectile*), atirador (classe *TGShooter*) e mola da arma (classe *TGSpring*). A Figura 7 representa as entidades gráficas onde todas são derivadas da classe base *TGEntity*.

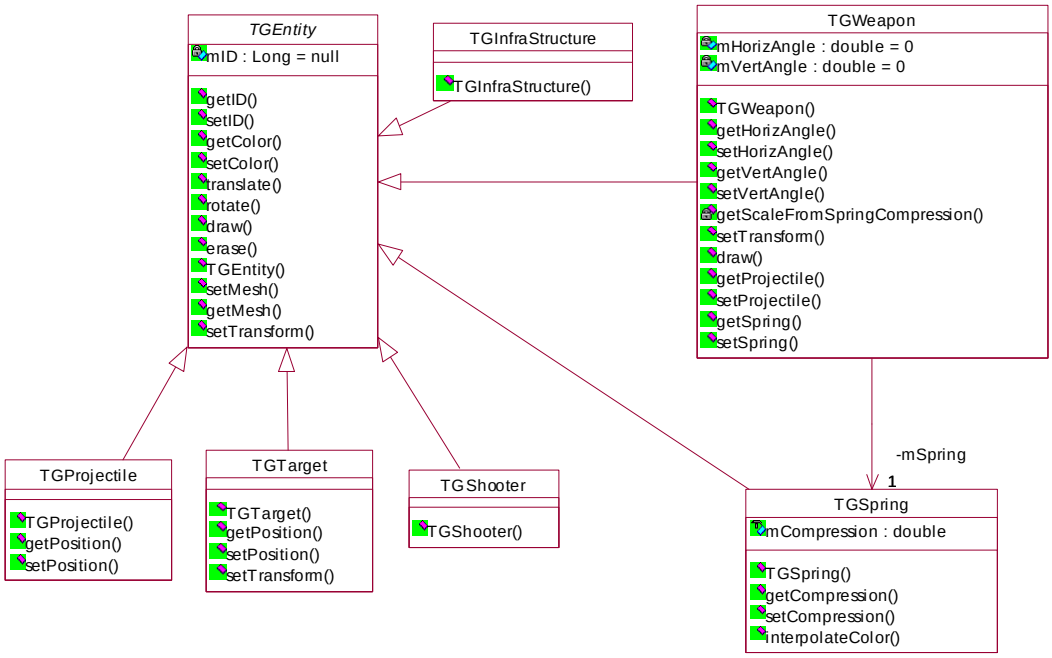

Figura 7 – Entidades Gráficas

3.4 Camada de Serviços

A camada serviços corresponde aos pacotes parser3ds e *trimesh*.

O pacote *trimesh* representa as classes para uma estrutura de dados triangularizada. A escolha desta estrutura se deve à forma como o aquivo 3dsasc representa as superfícies 3D.

O pacote *parser3ds* representa as classes para leitura do arquivo em formato 3dsasc. A classe *Parser3DS* faz a leitura do modelo geométrico e armazena uma coleção de estruturas *trimesh*.

A aplicação lê apenas uma vez o modelo no formato ascii e salva em um formato binário utilizando as funcionalidades de persistência de objetos da plataforma Java. Isto garante que a leitura dos modelos geométricos seja eficiente. Ou seja, a leitura em ascii é feita pelos desenvolvedores quando o modelo geométrico é modificado. A carga do modelo geométrica pelo jogador é feita através da leitura do arquivo binário.

4 Funcionalidades

Esta seção descreve as funcionalidades implementadas na aplicação “Tiro Certo” sob a perspectiva de desenvolvimento de software. Neste caso, a representação será baseada em diagramas de seqüência, o que mostra a interação e troca de mensagens entre as classes envolvidas. Cada diagrama de seqüência equivale a um documento de caso de uso.

As funcionalidades foram organizadas de acordo com suas características: funcionalidades da aplicação, funcionalidades de visualização, funcionalidade de disparo e funcionalidades de iluminação. Os casos de uso em linha tracejada representada no diagrama de caso de uso são funcionalidades já disponíveis na versão corrente da aplicação.

4.1 Funcionalidades da Aplicação

A Figura 8 representa o diagrama de caso de uso com as funcionalidades da aplicação.

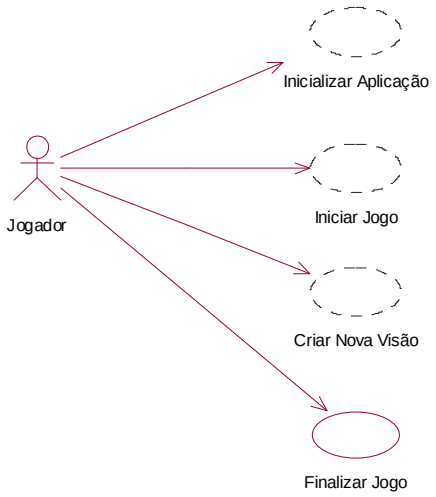

Figura 8 - Funcionalidades da Aplicação

As seções seguintes apresentam os diagramas de seqüência para as funcionalidades da aplicação.

4.1.1 Iniciar Aplicação

Esta funcionalidade se refere à inicialização principal da aplicação de acordo com a Figura 9. A execução desta funcionalidade abre a janela principal da aplicação, mas ainda não visualiza o cenário principal. A descrição deste caso de uso encontra-se na seção 5.1.

A classe principal que constrói a aplicação é a *TGTargetGameAppFactory*. Tal classe cria a aplicação (*TGTargetGameApp*), e adiciona à aplicação um documento vazio e os comandos da aplicação. Neste momento, o documento (classe *ITGDocument*) contem um banco de dados (*TGDatabase*) vazio.

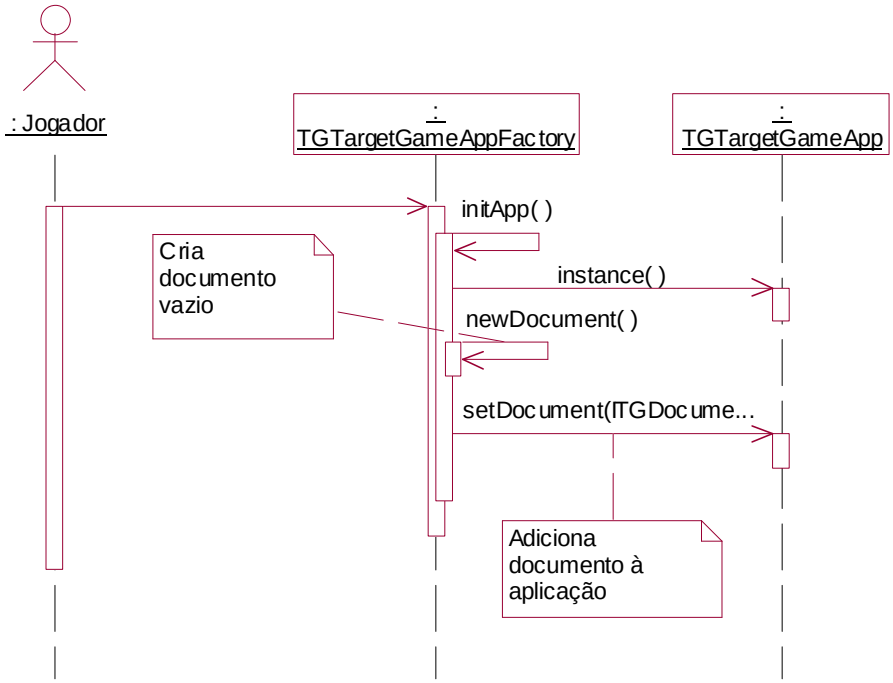

Figura 9 – Iniciar Aplicação

4.1.2 Iniciar Novo Jogo

Neste momento, o mediator chama o comando *NewGameCmd*, responsável por iniciar um novo jogo. O comando *NewGameCmd* solicita à classe *TGTargeGameAppFactory* a criação de uma nova visão através do método *newView*() e uma novo banco de dados através do método *newDatabase*().A descrição deste caso de uso encontra-se na seção 5.2.

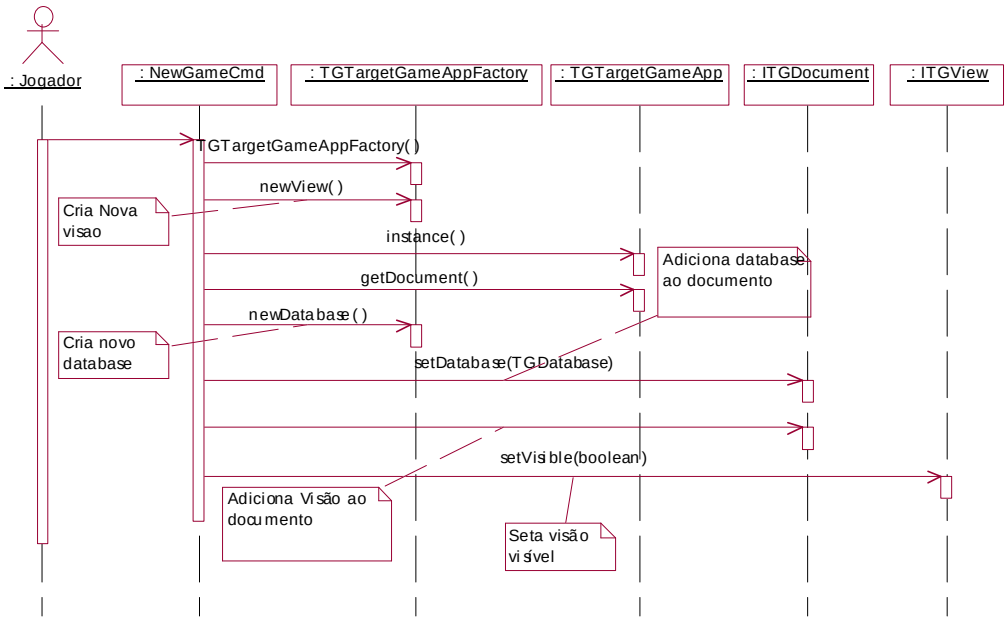

Figura 10 – Iniciar Novo Jogo

4.1.3 Criar Nova Visão

Esta funcionalidade permite ao usuário a criar uma nova janela, de forma que possa visualizar ao mesmo tempo o cenário sob diversos pontos de vista. Observe que a visão criada é associada ao documento. Como o documento possui o banco de dados da aplicação, o cenário será visualizado na nova janela. A descrição deste caso de uso encontra-se na seção 5.3 .

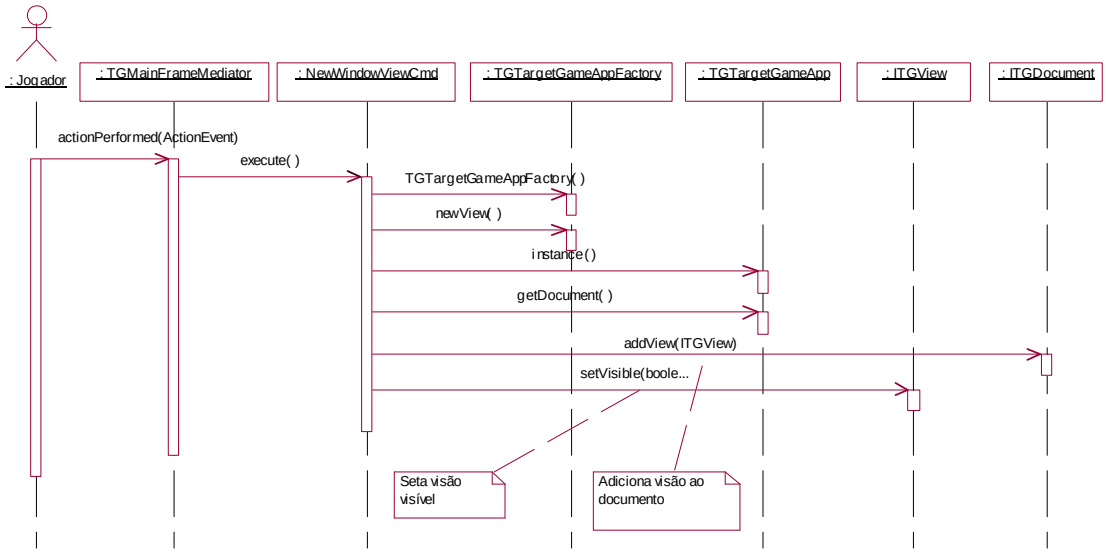

Figura 11 – Criar Nova Janela de Visualização

4.2 Funcionalidades de Visualização

A Figura 12 representa o diagrama de caso de uso para as funcionalidades de visualização.

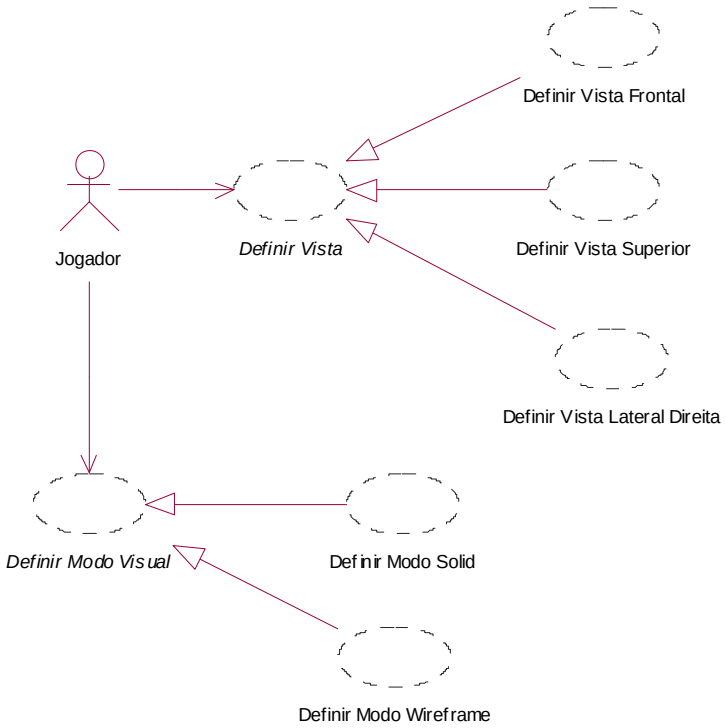

Figura 12 - Funcionalidades de Visualização

As seções seguintes apresentam os diagramas de seqüência para as funcionalidades de visualização.

4.2.1 Definir Vista

Esta funcionalidade permite a definição das seguintes vistas em uma das janelas selecionadas: Visão frontal, visão lateral, visão superior. As seções seguintes descrevem com detalhes estas funcionalidades. A descrição deste caso de uso encontra-se na seção 5.4 .

4.2.1.1 Definir Vista Frontal

A vista frontal corresponde aquela onde o jogador se encontra de frente ao alvo. A seqüência de eventos para definir uma visão frontal se encontra na Figura 13

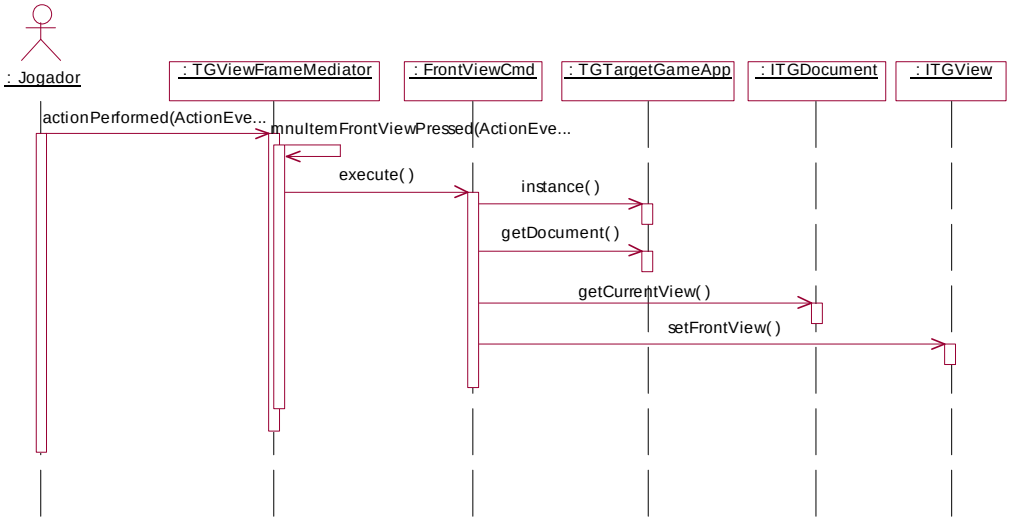

Figura 13 – Definir Vista Frontal

4.2.1.2 Definir Vista Lateral Direita

A vista lateral direita corresponde aquela onde o jogador se encontra na lateral direita do alvo, permitindo fazer um melhor ajuste do ângulo vertical de disparo. A seqüência de eventos para definir uma visão lateral direita se encontra na Figura 14.

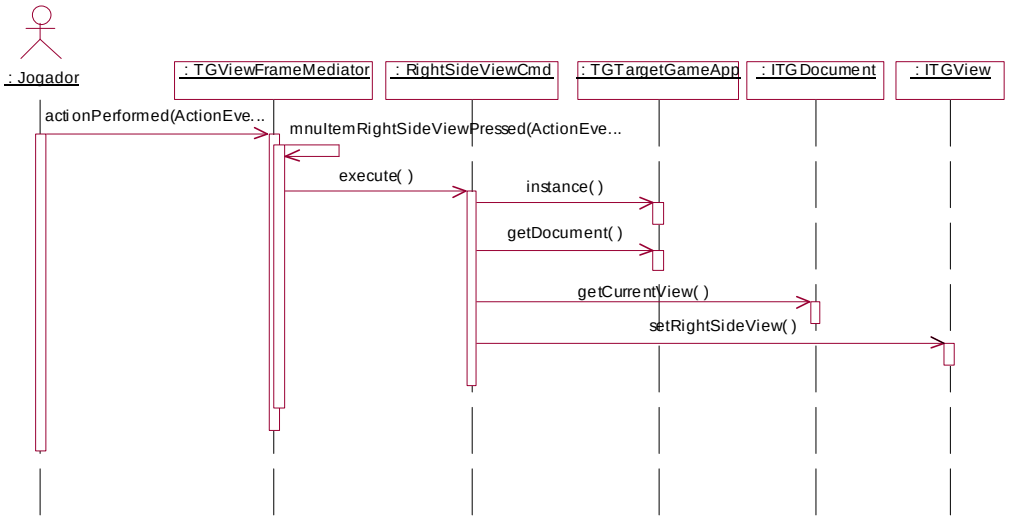

Figura 14 – Definir Vista Lateral Direta

4.2.1.3 Definir Vista Superior

A vista lateral direita corresponde aquela onde o jogador tem uma visão área do cenário. A seqüência de eventos para definir uma visão superior se encontra na Figura 15.

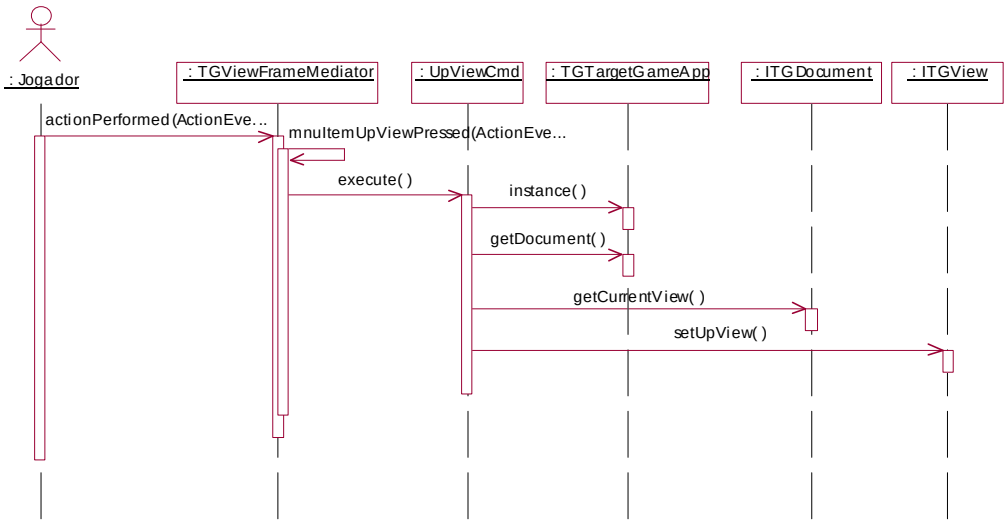

Figura 15 - Definir Vista Superior

4.2.2 Definir Modo Visual

Cada janela de visualização pode ser configurada para o modo sólido ou para o modo aramado. As próximas seções descrevem estes cenários. A descrição deste caso de uso encontra-se na seção 5.5.

4.2.2.1 Definir Modo Sólido

No modo sólido, os objetos são visualizados com informações de preenchimento e iluminação. A Figura 16 representa o diagrama de seqüência para esta funcionalidade.

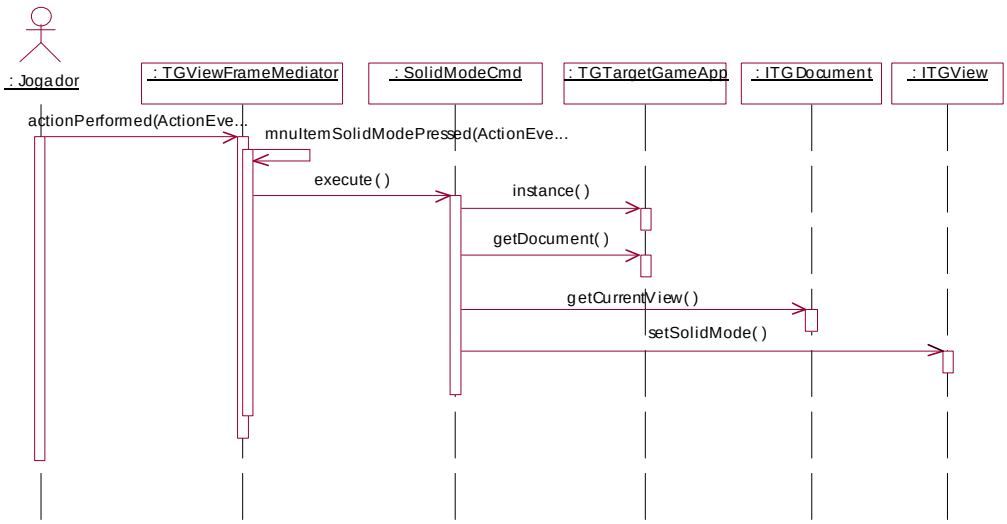

Figura 16 - Definir Modo Sólido

4.2.2.2 Definir Modo Aramado

No modo aramado, os objetos são visualizados sem o preenchimento. A Figura 17 representa o diagrama de seqüência para esta funcionalidade.

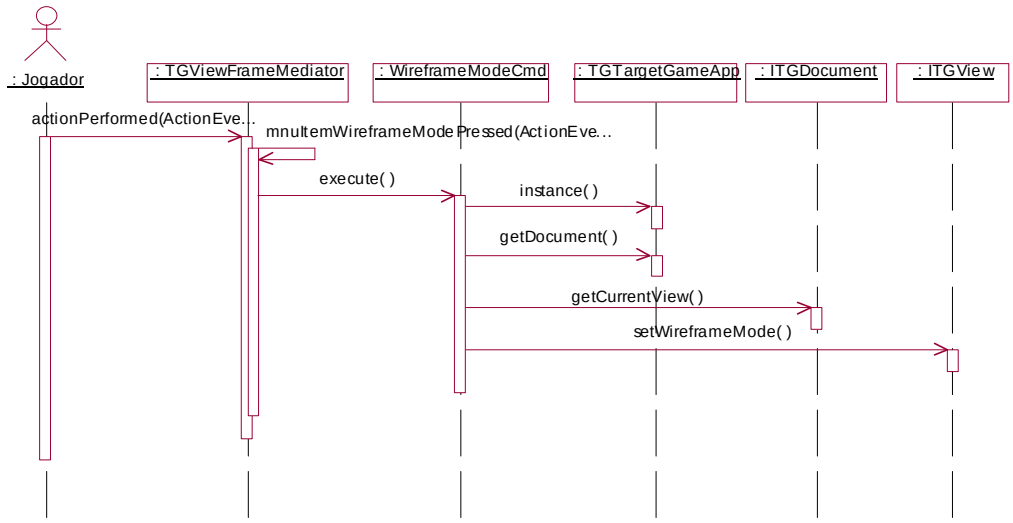

Figura 17 - Definir Modo Aramado

4.3 Funcionalidades de Disparo

As funcionalidades de disparo correspondem a funções de ajuste de ângulo, ajuste da força e o disparo propriamente dito. AFigura 18 ilustra o diagrama de caso de uso para as funcionalidades de disparo.

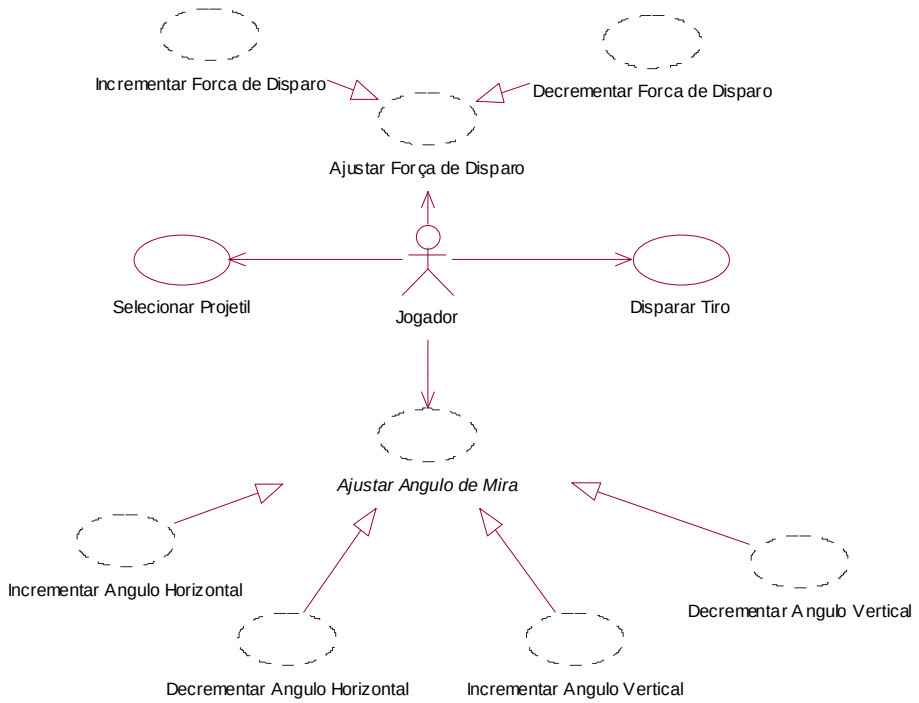

Figura 18 - Funcionalidades de Disparo

As seções seguintes descrevem o diagrama de seqüência para as funcionalidades de disparo.

4.3.1 Ajustar ângulo de Disparo

Esta funcionalidade permite o ajuste do ângulo de disparo do tiro segundo seu eixo vertical e horizontal. As seções seguintes descrevem com detalhes estas funcionalidades. A descrição deste caso de uso encontra-se na seção 5.6.

4.3.1.1 Incrementar Ângulo Horizontal de Disparo

Permite o ajuste do ângulo horizontal de disparo através de um incremento em tal ângulo. A Figura 19 representa o diagrama de seqüência para esta funcionalidade.

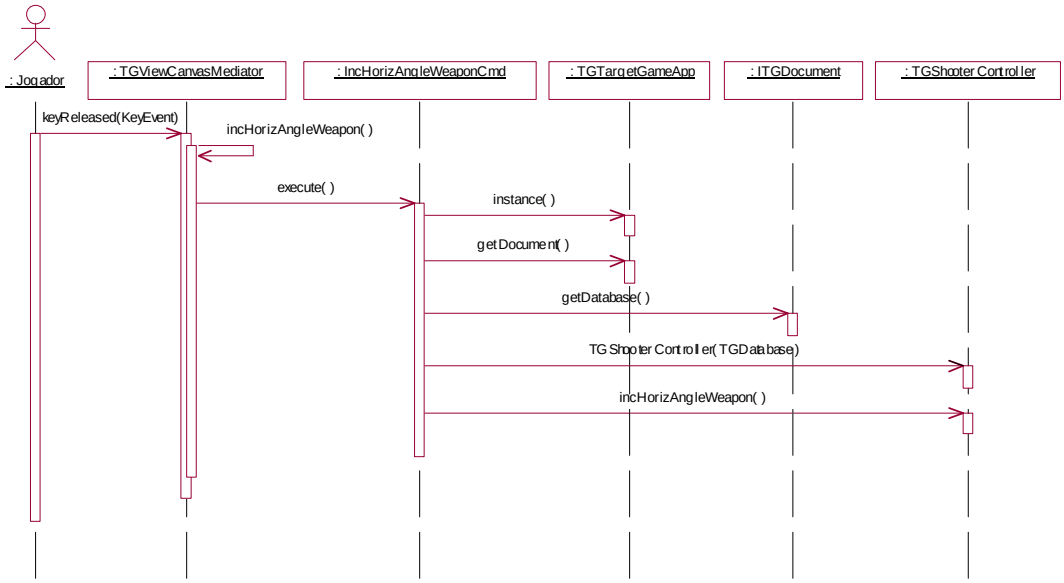

Figura 19 - Incrementar Angulo Horizontal de Disparo

4.3.1.2 Decrementar Ângulo Horizontal de Disparo

Permite o ajuste do ângulo horizontal de disparo através de um decremento em tal ângulo. A Figura 20 representa o diagrama de seqüência para esta funcionalidade.

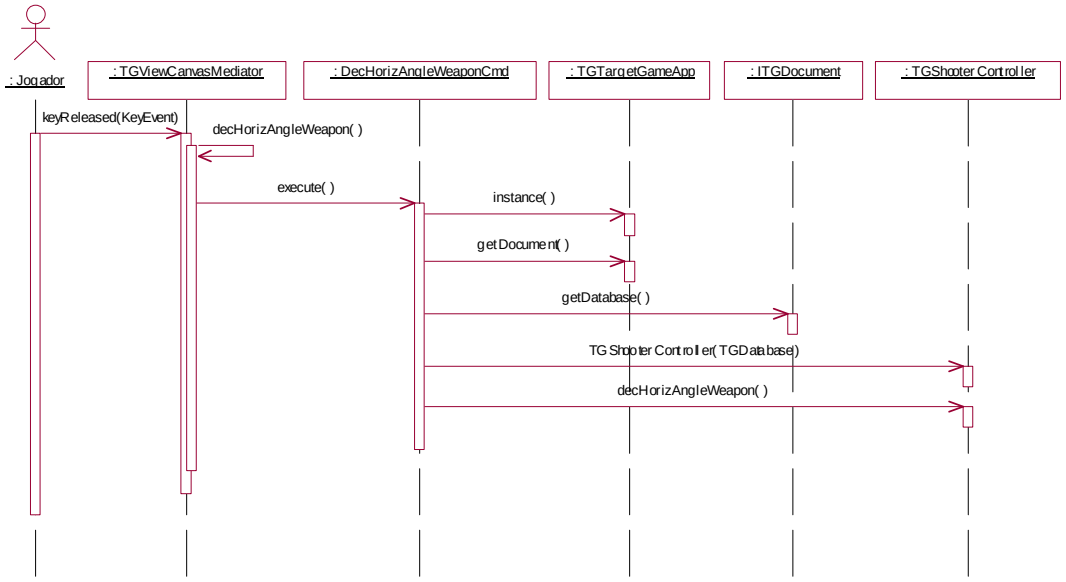

Figura 20 - Decrementar Ângulo Horizontal de Disparo

4.3.1.3 Incrementar Ângulo Vertical de Disparo

Permite o ajuste do ângulo vertical de disparo através de um incremento em tal ângulo. A Figura 21 representa o diagrama de seqüência para esta funcionalidade.

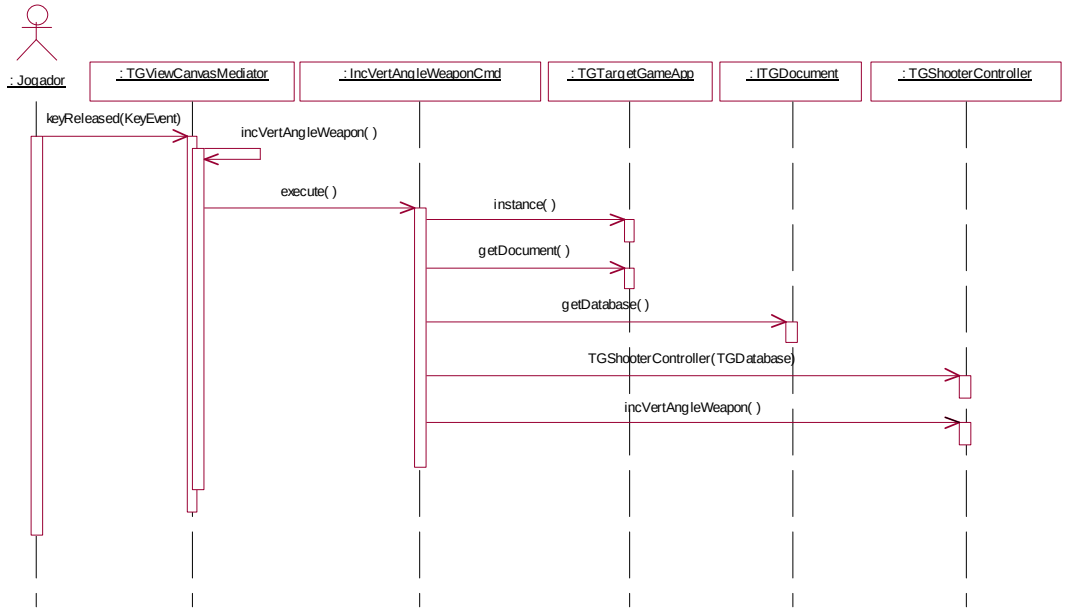

Figura 21 - Incrementar Ângulo Vertical de Disparo

4.3.1.4 Decrementar Ângulo Vertical de Disparo

Permite o ajuste do ângulo vertical de disparo através de um decremento em tal ângulo. A Figura 22 representa o diagrama de seqüência para esta funcionalidade.

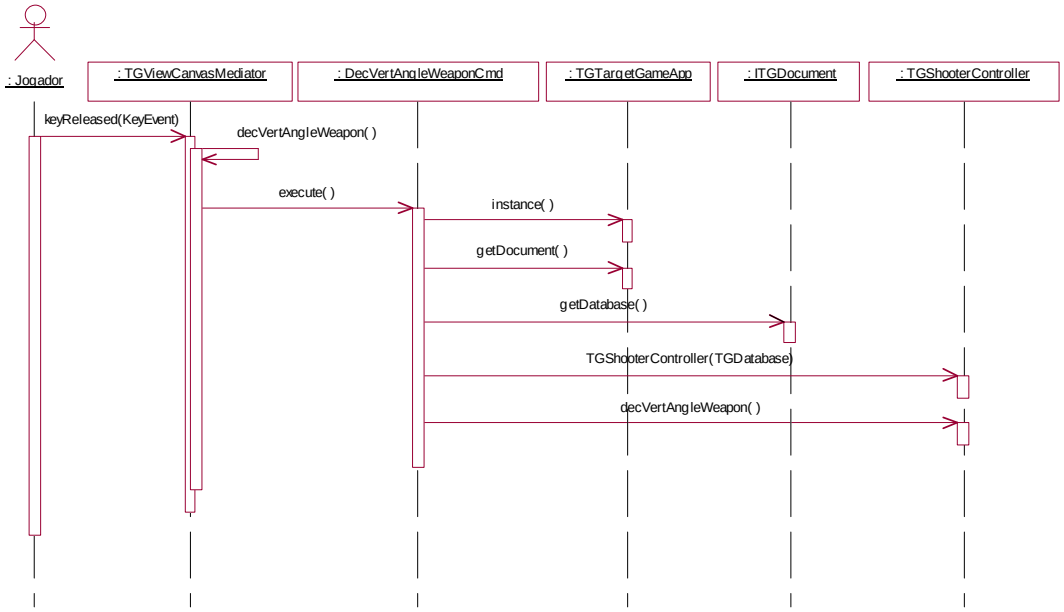

Figura 22 - Incrementar Ângulo Vertical de Disparo

4.3.2 Ajustar Força de Disparo

Permite o ajuste da força de disparo através de um incremento ou decremento da força corrente. A força de disparo é ajustada através da compressão da mola na arma. À medida que a mola vai se comprimindo, uma nova cor é calculada para a mola através de uma função de interpolação de cores. A descrição deste caso de uso encontra-se na seção 5.7.

4.3.2.1 Incrementar Força de Disparo

Incrementa a força de disparo do tiro. A Figura 23 ilustra o diagrama de seqüência para esta funcionalidade.

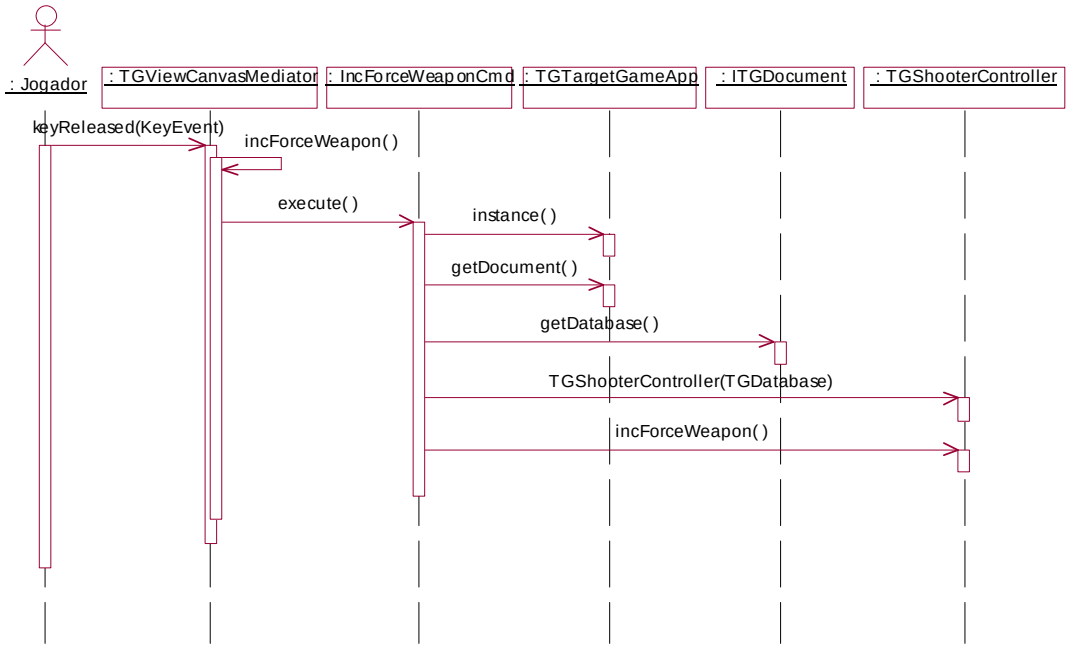

Figura 23 - Incrementar Força de Disparo

4.3.2.2 Decrementar Força de Disparo

Incrementa a força de disparo do tiro. A Figura 24 ilustra o diagrama de seqüência para esta funcionalidade.

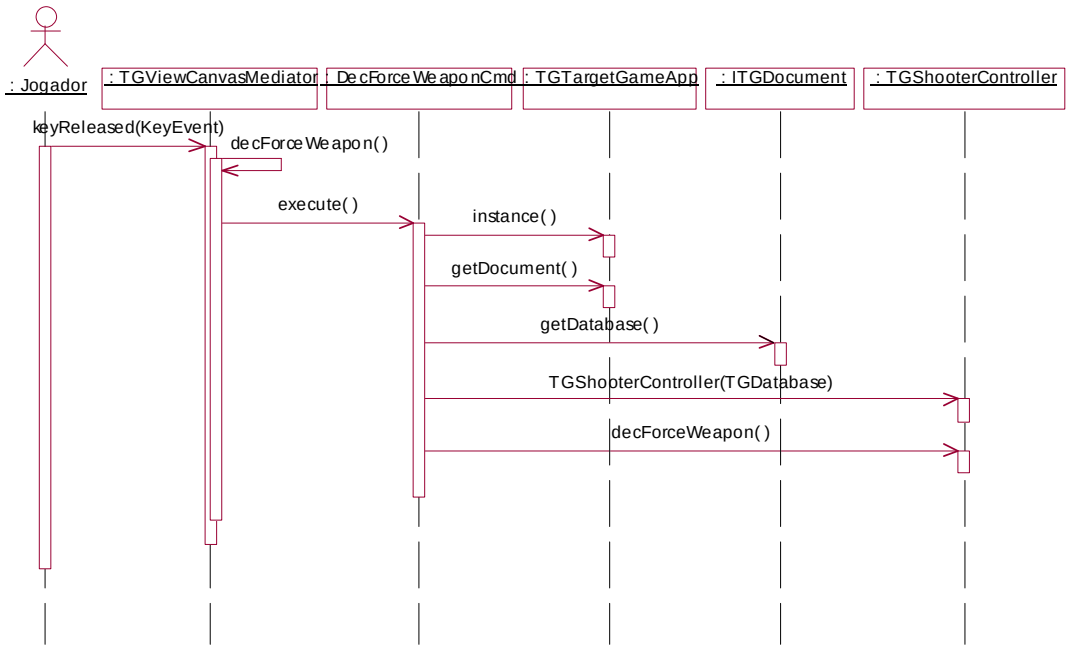

Figura 24 - Decrementar Força de Disparo

4.4 Funcionalidades de Iluminação

Estas funcionalidades são referentes para a configuração de um ambiente noturno e diurno. A Figura 25 ilustra o diagrama de caso de uso para as funcionalidades de visualização. A descrição deste caso de uso encontra-se na seção 5.8.

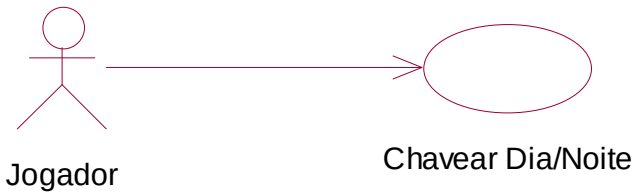

Figura 25 - Funcionalidades de Iluminação

5 – Casos de Uso

Este capítulo disponibiliza os documentos de casos de uso da aplicação. A partir destes documentos foi que se iniciou o desenvolvimento do sistema como um todo, incluindo sua arquitetura básica e os diagramas de seqüências.

5.1 Caso de Uso Iniciar Aplicação

Caso de Uso:	*Iniciar Aplicação*
Atores*:*	*Jogador*
Propósito:	Inicializar a aplicação.
Visão Geral*:*	*O Jogador Inicia a aplicação.*
Pré-Condição*:*	
Pós-Condição*:*	*Aplicação inicializada.*
Triggers*:*	*O caso de uso inicia quando o jogador executa o comando de inicialização da aplicação.*
Tipo*:*	*Primário e Essencial*
Referências*:*	*Funções: initApp()*

Fluxo Principal

Usuário	Sistema
1. O caso de uso inicia quando o jogador executa o comando de inicialização da aplicação.	2. Abre a janela principal da aplicação

Fluxos Alternativos

5.2 Caso de Uso Iniciar Jogo

Caso de Uso: *Iniciar Jogo*
Atores*:* *Jogador*

Propósito: Inicia um novo Jogo.
Visão Geral*:* *O Jogador Inicia um novo jogo e o sistema constrói o cenário do jogo.*
Pré-Condição*:* *Sistema iniciado e nenhuma existência de um jogo em andamento.*
Pós-Condição*:* *Novo jogo iniciado*
Triggers*:* *O caso de uso inicia quando o jogador seleciona a opção "Iniciar novo Jogo".*
Tipo*:* *Primário e Essencial*
Referências*:* *Funções: initializeGame*

Fluxo Principal

Usuário	Sistema
1. O caso de uso inicia quando o jogador seleciona a opção "Iniciar novo Jogo" na interface do sistema	2. Constrói o cenário do jogo onde constam as construções, o alvo, a arma e um projétil com peso default (mais leve).
	3. Mostra o cenário sob o ponto de vista do atirador
	4. Inicia a arma com força zero e ângulos horizontais e verticais também iguais a zero.

Fluxos Alternativos

5.3 Caso de Uso Criar Nova Visão

Caso de Uso: *Criar Nova Visão*
Atores*:* *Jogador*

Propósito: Abre uma nova visão.
Visão Geral*:* *O Jogador cria uma nova visão do cenário, permitindo a visualização sob diversos pontos de vistas. Isto garante um melhor ajuste de ângulo e força de disparo.*
Pré-Condição*:* *Jogo iniciado.*
Pós-Condição*:* *Novo visão criada*
Triggers*:* *O caso de uso inicia quando o jogador seleciona a opção "Criar Nova Visão".*
Tipo*:* *Primário e Essencial*
Referências*:* *Funções: newView()*

Fluxo Principal

Usuário	Sistema
1. O caso de uso inicia quando o jogador seleciona a opção "Iniciar Nova Visão" na interface do sistema	2. Abre uma nova janela com o cenário visto de uma posição default da câmera.

Fluxos Alternativos

5.4 Caso de Uso Definir Vista

Caso de Uso: *Definir Vista*
Atores*:* *Jogador*

Propósito: Selecionar um novo ponto de vista.
Visão Geral*:* *O Jogador seleciona um novo ponto de vista (frontal, lateral direita e superior) para auxiliar no disparo.*
Pré-Condição*:* *Jogo inicializado.*
Pós-Condição*:* *Cenário apresentado sob o novo ponto de vista selecionado.*
Triggers*:* *O caso de uso inicia quando o jogador seleciona a opção para seleção de um dos pontos de vista disponível: frontal, lateral direita e superior..*
Tipo*:* *Primário e Essencial*
Referências*:* *Funções: setViewPoint*

Fluxo Principal

Usuário	Sistema
1. O caso de uso inicia quando o jogador seleciona uma das vistas disponíveis: frontal, lateral direita e superior.	2. Exibe o cenário de acordo com o ponto de vista selecionado pelo jogador.

Fluxos Alternativos

5.5 Caso de Uso Definir Modo Visual

Caso de Uso: *Definir Modo Visual*
Atores*:* *Jogador*

Propósito: Selecionar um novo modo de visualização.
Visão Geral*:* *O Jogador seleciona um novo modo de visualização (sólido ou aramado) para uma das janelas do cenário. O modo de visualização é aplicado apenas na janela corrente.*
Pré-Condição*:* *Jogo inicializado.*
Pós-Condição*:* *Cenário apresentado com o modo de visualização selecionado.*
Triggers*:* *O caso de uso inicia quando o jogador seleciona a opção para seleção de um dos modos de visualização: sólido ou aramado.*
Tipo*:* *Primário e Essencial*
Referências*:* *Funções: setModeView*

Fluxo Principal

Usuário	Sistema
1. O caso de uso inicia quando o jogador seleciona um dos modos de visualização: sólido ou aramado.	2. Exibe o cenário de acordo com o modo de visualização selecionado.

Fluxos Alternativos

5.6 Caso de Uso Ajustar Ângulo de Disparo

Caso de Uso: *Ajustar ângulo de Disparo*
Atores*:* *Jogador*

Propósito: Selecionar um novo ângulo de disparo.
Visão Geral*:* *O Jogador Seleciona um novo ângulo de disparo para o projétil. O cenário é atualizado com o novo ângulo de disparo.*
Pré-Condição*:* *Jogo inicializado e projétil na arma.*
Pós-Condição*:* *Ângulo de disparo da arma ajustado.*
Triggers*:* *O caso de uso inicia quando o jogador seleciona a opção para ajustar o novo ângulo de disparo.*
Tipo*:* *Primário e Essencial*
Referências*:* *Funções: setShotAngle*

Fluxo Principal

Usuário	Sistema
1. O caso de uso inicia quando o jogador pressiona uma das teclas para o ajuste do ângulo: seta p/ baixo, seta p/ cima, seta p/ esquerda, seta p/ direita.	2. Ajusta o ângulo de disparo da arma de acordo com a tecla pressionada: • Seta para cima/baixo: ângulo vertical (0 a 90 graus) • Seta para direita/esquerda: ângulo horizontal (0 a 180 graus).
	3. Atualiza na tela o novo estado da arma: inclinação no ângulo de tiro. Se houver mais de uma janela, todas serão atualizadas.

Fluxos Alternativos

5.7 Caso de Uso Ajustar Força de Disparo

Caso de Uso: *Ajustar Força de Disparo*
Atores: *Jogador*

Propósito: Selecionar a força de disparo do projétil.
Visão Geral: *O Jogador seleciona a força de disparo do projétil. A força de disparo é uma função linear proporcional à compressão da mola existente na arma. A cor da mola é calculada através de uma função de interpolação de cores de acordo com a compressão da mola.*
Pré-Condição: *Jogo inicializado e projétil na arma.*
Pós-Condição: *Força de disparo do projétil ajustado.*
Triggers: *O caso de uso inicia quando o jogador seleciona a opção para ajustar a nova força de disparo.*
Tipo: *Primário e Essencial*
Referências: *Funções: setShotForce*

Fluxo Principal

Usuário	Sistema
1. O caso de uso inicia quando o jogador seleciona uma das teclas para ajuste da força de disparo. • <+>: mais força • <->: menos força	2. Ajusta a força de acordo com o valor selecionado pelo usuário.
	3. Atualiza cenário com a mola comprimida de acordo com a força configurada
	4. Atualiza cenário com a nova cor da mola calculada de acordo com a compressão da mola usando uma interpolação de cores.

Fluxos Alternativos

- Caso o usuário pressione a tecla de tal modo a passar do limite mínimo ou máximo, o sistema deixa inativa a tecla.

5.8 Caso de Uso Chavear Dia/Noite

Caso de Uso: *Chavear Dia/Noite*
Atores*:* *Jogador*

Propósito: Selecionar entre dia/noite para praticar o tiro ao alvo.
Visão Geral*:* *O Jogador seleciona entre dia e noite para a prática do tiro. A pontuação obtida durante a noite será três vezes maior do que durante o dia. O chaveamento poderá ser realizado durante o jogo.*
Pré-Condição*:* *Jogo inicializado.*
Pós-Condição*:* *Cenário apresentado sob as novas condições de iluminação.*
Triggers*:* *O caso de uso inicia quando o jogador seleciona a opção para chaveamento entre dia/noite.*
Tipo*:* *Primário e Essencial*
Referências*:* *Funções: setHour*

Fluxo Principal

Usuário	Sistema
1. O caso de uso inicia quando o jogador seleciona uma opção entre dia/noite.	2. Exibe o cenário de acordo com as novas condições de iluminação.
	3. Quando a iluminação for noturna, exibe o cenário com projétil e alvo emitindo luz

Fluxos Alternativos