Ás da Aviação

GITHUB: https://github.com/Gabrui/CES22P1

BLOG: https://gamdev.com.br/grupo/

Resumo

Um jogo bidimensional de ação, (horizontal flight shooter), ambientado na segunda guerra mundial. Haverá missões e progressão de carreira do jogador, que se baseia em um sistema de pontuação. O tipo de jogo descrito está exemplificado na Figura 1.

Figura 1. Jogo bidimensional de ação horizontal.

Descrição

O jogo tem menus, mantém informações dos jogos salvos, onde cada jogo salvo tem um título, tem o seu progresso (número da missão desbloqueadas), lista de itens comprados (ids) e de pontos do jogador. Na execução do jogo em si, o jogador terá o controle de um avião, batalhando contra outros aviões inimigos.

Menus

Os menus têm música de fundo, imagens de fundo e botões que levam para outros menus ou para o jogo. Os botões são interativos e têm estados normal, selecionado (mouse em cima), clicado e inativo. Eles podem ser navegados pelo mouse (clicáveis) ou pelas setas do teclado e acionados com ENTER.

O menu inicial tem botões de “Continuar”, “Novo Jogo”, “Jogos Salvos”, “Opções de Jogo”, “Instruções”, “Créditos” e “Sair do Jogo”, a exemplo da Figura 2.

Figura 2. Exemplo de Menu Inicial com botões selecionáveis.

O menu de Novo Jogo tem um campo de entrada para o jogador digitar o seu nome, o nome do arquivo do jogo salvo e o botão de iniciar o jogo.

O menu de Jogos Salvos apresenta uma lista selecionável com os jogos salvos e um botão para iniciar o jogo.

O menu Opções de Jogo apresenta um slider (barra ajustável) para o volume da música e para o volume dos sons de efeitos, apresenta três botões de seleção para a qualidade gráfica dos efeitos especiais (densidade das partículas), apresenta dois botões para habilitar ou desabilitar o assistente de vôo e apresenta dois botões para a seleção do tipo de controle, mouse ou teclado.

O menu de instruções apresentará os comandos básicos do jogo, como teclas para atirar, se movimentar e etc.

O menu de créditos apresenta um texto com créditos e um botão para voltar para o menu inicial.

Durante o jogo, terá um menu de pause com botões Resumir, Carregar Jogo Salvo, Opções de Jogo, Sair.

Tutorial

Ao iniciar um novo jogo o jogador se deparará com um tutorial que lhe ensinará os comando para jogar, o tutorial pode ter várias páginas, sendo que cada página tem um botão de página anterior ou próxima página, a exemplo da Figura 3. Na última página haverá além do botão de página anterior e de iniciar jogo, dois botões de seleção de tipo de controle, por mouse ou por teclado e dois botões para perguntar se o assistente de vôo será habilitado ou não.

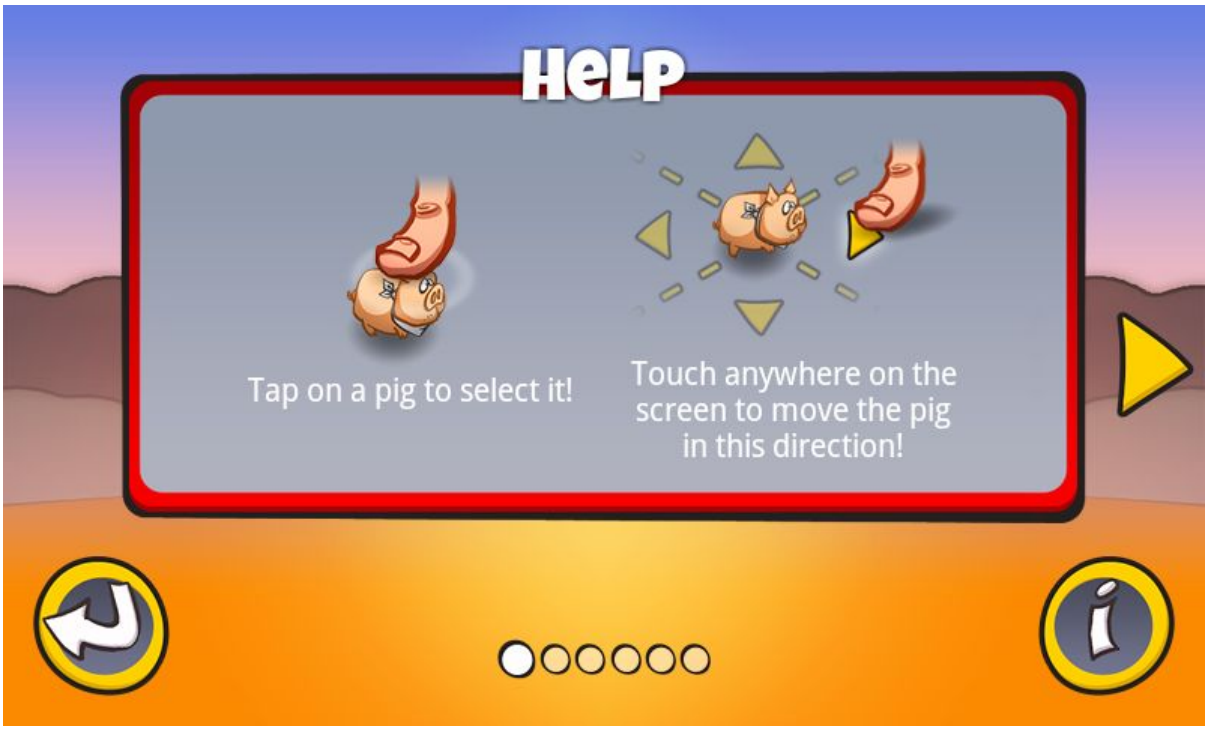


Figura 3. Exemplo de um tutorial com múltiplas páginas.

Ao clicar em iniciar, o jogador será direcionada para o menu da loja, onde, apenas nessa primeira vez, aparecerá um HUD (Heads Up Display) ensinando a usar cada item da loja. O HUD poderá ser estático e quando o jogador clicar, ele desaparece, ou poderá ser sequencial, com cada clique significando um avanço nas instruções do HUD.

Jogo Corrente

Durante o jogo corrente, haverá um avião controlado pelo jogador (mouse ou teclado), que seguirá um modelo aerodinâmico simples. Essa avião terá armas, que podem

incluir metralhadoras, foguetes e bombas, acionadas pelo teclado ou mouse. O jogador poderá controlar as manetes (potência), ligar ou desligar o motor, o manche, os flaps, o trem de pouso (que freia automaticamente), efetuar a manobra de inversão e o armamento do avião. OBS: Na dificuldade fácil, o jogador terá o controle apenas das manetes, manche, armamento e de inversão, sendo o trem de pouso, os flaps e o motor automáticos (assistente de vôo).

HUD

Existe o HUD do avião que indica a sua altitude (altímetro), velocidade (velocímetro), radar (que indica outros objetos no mapa), indicador de missão e <quantidade de danos>. Há também um botão de pause que pode ser acionado por clique ou por tecla (P ou ESC), que abrirá o menu de pause. O zoom da câmera do jogo poderá variar de acordo com a velocidade do avião, quanto maior é a velocidade do avião, mais afastado será a percepção da câmera de jogo.

Missões

A primeira missão será um hand-ons para ensinar a pilotar, com um HUD inicial que apresenta um texto instrutivo que se modifica à medida que o jogador vai executando os comandos, por exemplo, ligar o avião, dar potência máxima nas manetes (apertando + ou afastando o cursor do mouse <a decidir o controle>), puxar o manche (apertando seta para baixo ou mexendo no mouse), levantar trem de pouso, controlar flaps, atirar (apertando o botão esquerdo do mouse ou botão <de atirar> do teclado), soltar bombas ou foguetes, virar o avião, diminuir a potência das manetes, acionar os flaps, acionar o trem de pouso e pousar.

As outras missões serão definidas de acordo com um enredo, cujos os objetivos podem ser bombardear alvos específicos, destruir um número determinado de inimigos, defender a base ou bombardeiros, dentre outros. Todo inimigo destruído será usado como ponto. O início das missões será caracterizado pela decolagem do avião, e que vai percorrer uma certa distância, um cenário predeterminado que pode variar de acordo com a missão. Ao finalizar a missão atualizará o indicador de missão e sons parabenizando o jogador.

O jogador terá um determinado tempo para completar cada missão, e esse tempo será medido em termos de “combustível disponível”.

****Pensar sobre sistema de recompensas.*

Cenário

O cenário apresenta uma paralaxe, imagens de fundo, imagens intermediárias e imagens de frente. As imagens podem incluir árvores, folhagens, casas, paisagens, dentre outros.

IA

Os inimigos terão uma inteligência artificial e serão capazes de infligir danos sobre o jogador, o que pode ocasionar até a morte do seu piloto ou destruição do avião, sendo necessário a restauração do jogo para o último estado salvo.

A IA será capaz de localizar um alvo, perseguir o alvo, atirar no alvo, efetuar manobras defensivas, proteger um alvo, isto é, localizar e atacar agressores do alvo, contra-atacando-os em um raio pré definido em torno do alvo de defesa. O jogador também poderá efetuar certas manobras para confundir o IA, como por exemplo, entrar em uma nuvem, voar perto do chão, efetuar um loop dentre outras manobras.

Loja

Toda vez que o avião pousar na base ele poderá ser reparado e abrirá um “Menu da Loja”. No final das missões o avião deverá pousar na base, salvando o jogo, haverá um sistema para compra de aviões e armas, isto é, um “Menu da Loja”. Os itens da loja podem incluir novos aviões, modificações nos aviões, seja por melhorias de defesa de armamentos, metralhadoras, bombas, foguetes, e poderão ser adquiridos por meio de um sistema de pontuação.

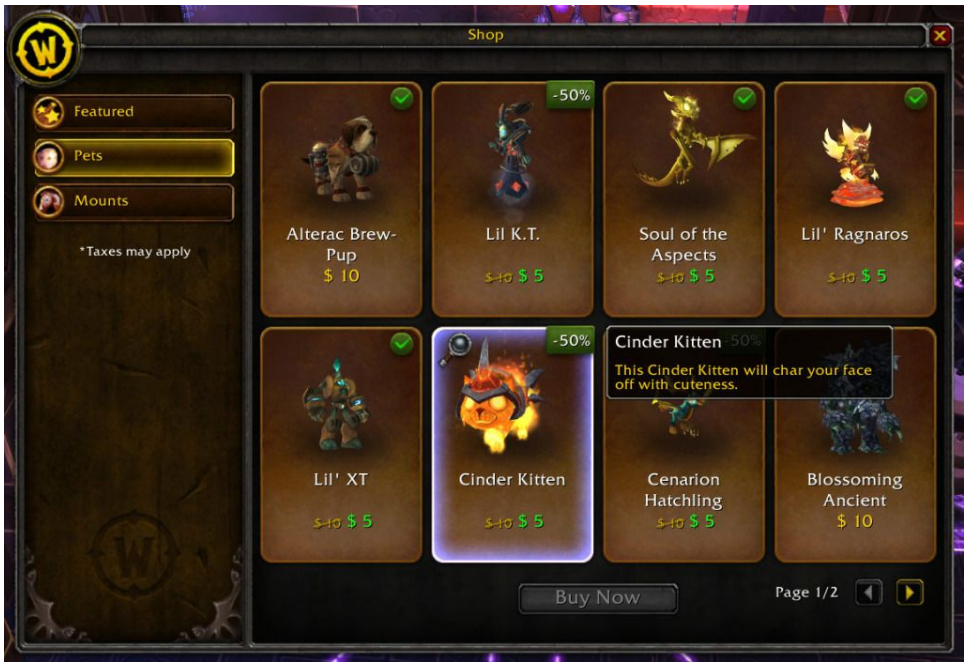


Figura 4. Exemplo do menu de loja de um jogo.

Modelagem CRC

Passo 0: Especificação do sistema

Passo 1: Identificar classes através dos substantivos do texto da especificação (potenciais classes)

Passo 2: Refinar nossa lista de substantivos, identificando classes e subclasses, descrever o que cada uma faz, e suas relações (herança, polimorfismos), e eliminando classes que não devem fazer parte (sinônimos, atributos, nomes fora do escopo , verbos enrustidos...). Classes são abstrações de objetos que apresentam características comuns, por isso devem ser no singular e devem ter a primeira letra maiúscula e CamelCase. Descrever também as classes, escrevendo o que foi planejado ou o que planejaremos para essas classes

Passo 3: Identificar as responsabilidades óbvias. São aquelas que ao olhar o nome da classe e ao olhar a descrição que nós demos para a finalidade da classe aparecem naturalmente.

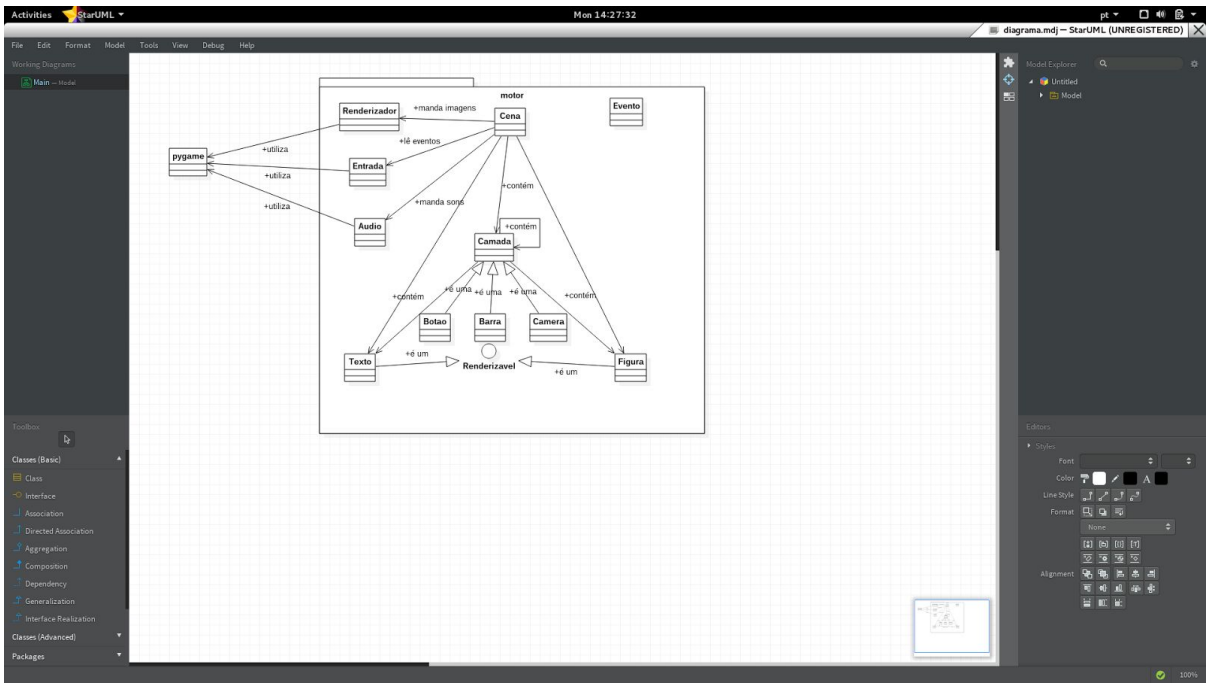
Passo 4: Verbos como responsabilidades. Listar todos os verbos de ação que eu encontro na especificação.

Passo 5: Atribuir as responsabilidades listadas anteriormente. Examinar a qual classe a responsabilidade deve ser atribuída. Cuidado com o polimorfismo, pois a mesma responsabilidade pode valer para classes diferentes com lógicas diferentes.

Passo 6: Descrever a lógica de cada responsabilidade de cada classe. Identificar também as colaborações das novas responsabilidades.

<table>
<tr><td colspan="2">ExemploDeCartao</td></tr>
<tr><td colspan="2">Descrição: Este é um exemplo de cartão CRC</td></tr>
<tr><td>Responsabilidades</td><td>Colaborações</td></tr>
<tr><td>Registra uma nova conta</td><td>Colaboradora: ContaCorrente
Colaboração: Construtor</td></tr>
<tr><td>Apresenta os números das contas ativas</td><td>Colaboradora: ContaCorrente
Colaboração: Sabe o número da conta</td></tr>
<tr><td>Obtém o saldo total do banco</td><td>Colaboradora: ContaCorrente
Colaboração: Sabe o valor do saldo atual</td></tr>
<tr><td colspan="2">Lógica interna das reponsabilidades:
Registra uma nova conta
Cria uma nova ContaCorrente e a adiciona à lista de contas
Apresenta os números das contas ativas
Para cada ContaCorrente na lista de contas, adiciona o número dessa conta à uma lista temporária e a retorna
Obtém o saldo total do banco
Para cada ContaCorrente na lista de contas, adiciona o seu saldo ao saldo total do banco</td></tr>
</table>

Biblioteca motor



OBS: Os cartões CRC e o diagrama ficaram desatualizados com relação ao código que estou escrevendo, vou atualizar depois

Cena	
Descrição: É a classe pai de Menu e da exibição do simulador, é como se fosse a tela e é quem tem contato direto com o renderizador. Herda camada, pois é uma camada raiz.	
Responsabilidades	**Colaborações**
É atualizável a cada interação	Acionado pelo Jogo
Sabe e gerencia as camadas que contém e sua ordem	
Sabe e gerencia os objetos renderizáveis que possui	
Tem o seu sistema de coordenadas	
Comunica-se com o renderizador, passando as instruções para renderizar um objeto ou limpar a tela	Renderizador
Comunica-se com a entrada, lendo os eventos registrados pela entrada e adicionando registradores de eventos	Entrada

Comunica-se com o Audio, mandando a execução ou parada de algum som.	Audio
Comentário: Precisamos de cena se temos o renderizador? O renderizador será a classe que fará contato com os métodos do Pygame, efetuando transformações necessárias nas imagens, buffers, etc. A cena será uma entidade lógica que cuidará da lógica acima da renderização, como por exemplo, saber a ordem de renderização de cada objeto, quais objetos estão dentro da tela e poderíamos até deixar responsável para cena saber qual música de fundo tocar.	

Camada	
Descrição: As camadas são acrescidas às cenas, elas são filhas de renderizável	
Responsabilidades	**Colaborações**
Conhece suas coordenadas x, y (tem seu próprio sistema de coordenadas)	
Conhece os objetos acrescidos à ela	
Conhece o seu z-index (profundidade)	

Câmera	
Descrição: Um tipo de camada especializada feita para dar zoom e acompanhar o avião do jogador , é filho da Camada	
Responsabilidades	**Colaborações**
Herdado de camada	
Sabe modifica a escala dos objetos e	

camadas contidas nela	

Renderizavel	
Descrição: É uma classe abstrata para objetos que forem inseridos na cena ou nas camadas	
Responsabilidades	**Colaborações**
Conhece suas coordenadas x, y e rotação em relação à sua camada	
Conhece alguns atributos de efeito da imagem: opacidade, escala	
Pode ser atualizada frame a frame	
Pode modificar e ler seus atributos	

Figura	
Descrição: As figuras são acrescidas às camadas ou diretamente na cena, é filho do Renderizável	
Responsabilidades	**Colaborações**
Conhece suas coordenadas x, y e rotação em relação à sua camada	Herdado de Renderizável
Conhece as coordenadas (x, y) do centro de sua imagem (centro de rotação da	

imagem)	
Conhece a sua imagem	

Texto	
Descrição: Os textos são acrescidos às camadas, é filho do Renderizável	
Responsabilidades	**Colaborações**
Conhece suas coordenadas x, y e rotação em relação à sua camada	Herdado de Renderizável
Conhece o seu conteúdo de texto	
Conhece a sua fonte e suas propriedades	
Pode ter efeitos de texto? O que é efeito de texto?	

Menu	
Descrição: Representa as telas de menus com seus respectivos botões. É um filho de cena.	
Responsabilidades	**Colaborações**
Ter botões	Botao

Ter imagem	
Saber música de fundo	Gerenciador de Música

Comentário:

-Ter botões
- Saber a quantidade de botões que possui
- Criar os objetos referentes a cada botão
    - Atribui as características particulares de cada botão (herança de botao)

-Ter imagem
- Saber em qual diretório está a imagem de fundo do menu

-Saber música de fundo
- Saber em qual diretório está a música de fundo daquele menu

Botao

Descrição: Representa o botão

Responsabilidades	**Colaborações**
Saber seu tamanho	
Saber seu evento	Menu
Saber suas imagens	
Saber o seu estado (normal, selecionado, inativo)	
Saber sua posição	
Criar botões	

Comentário:

-Saber seu tamanho
- Definição do formato geométrico do botão
- Definição das medidas dos atributos geométricos do botão

-Saber seu destino
- Saber qual classe ou objeto chamar ou saber qual evento o sucede

-Saber suas imagens
- Saber em qual diretório está a sua imagem

-Saber o seu estado
- Separar em estado clicado ou não clicado, e estado ativo ou não-ativo

-Saber sua posição
- Recebe a posição do botão na entrada da função
- Atribui a posição do botão na tela usando uma função do Pygame

-Criar botões
- Construtor de botões

Cenario

Descrição: Representa o cenário

Responsabilidades	**Colaborações**
Saber sua imagem	
Saber sua velocidade	
Saber a lógica de transição de cena	

Comentário:

Audio

Descrição: Representa o gerenciador de áudio do jogo

Responsabilidades	**Colaborações**
Saber qual som executar e como (contínuo ou uma vez)	Cena
Reprodução de efeitos sonoros	pygames

Comentário:

-Saber qual música tocar
- executar Musica

-Reproducao de efeitos sonoros
- Verificar se é gameplay ou menu
    - Se for menu,
        - executar o som dos botões quando forem clicados
    - Se for gameplay,
        - executar som do motor do avião
            - buscar o diretório e o nome do arquivo som do avião chamando a classe aviao
            - procurar o arquivo som do avião
            - abrir o arquivo
            - executar e fechar o arquivo
        - executar som da arma de cada disparo
            - buscar o diretório e o nome do arquivo som de cada arma chamando a classe arma
            - procurar o arquivo som da arma
            - abrir o arquivo
            - executar e fechar o arquivo
        - executar som de explosão de bombas ou foguetes
        - executar som de destruição de edifício
        - executar som de parabenização de conclusão de missão

Evento

Descrição: Representa o gerenciador de eventos do jogo

Responsabilidades	**Colaborações**
Tem uma lista com os eventos registráveis (que são strings) e suas funções de chamada	
Tem uma lista dos eventos registrados	

Comentário:

Renderizador

Descrição: O renderizador é a interface que se comunica diretamente com o pygames para desenhar na tela

Responsabilidades	**Colaborações**
Recebe as informações de um objeto para poder desenhá-lo na tela	Cena
Redesenha a tela	Cena
Utiliza as funções do pygames para desenhar na tela	pygames
Comentário: A lista de objetos deve conter a referência a imagem do objeto, a posição do objeto, a escala do objeto	

Específico do jogo

Jogo	
Descrição: Representa o jogo	
Responsabilidades	**Colaborações**
Controlar o loop do jogo	python time
Criar e gerenciar cenas	
Comentário: Abrir janela do jogo ● Verificar chamadas de menu ○ Chamar menu correspondente a chamada ● Verificar chamada de fechamento da janela do jogo ● Realizar fechamento do jogo ● Inicializar o simulador ● Verificar chamado do simulador ○ Realizar a chamada do simulador ● Verificar chamadas de outros objetos gerenciadores ○ Realizar chamada do gerenciador	

Simulador

Descrição: Representa o jogo corrente

Responsabilidades	**Colaborações**
Ter um jogador	Jogador
Ter objetos controlados por IA	IA
Verificar colisões	
Verificar chamada do Menu de Pause	
Verificar chamada do Menu da Loja	

Comentário:

- Ter um jogador
- Criar objeto da classe Jogador
    - Chamar classe Jogador
        - Atribuição das características para classe Jogador

- Ter objetos controlados por IA
- Chamar classe Missao
    - Verificar quantos e quais inimigos o gameplay atual possui
- Criar objetos das classes Inimigos
    - Atribuição das características dos inimigos

-Verificar colisões

- Verificar interação do avião do jogador com o cenário, com os projéteis ou bombas inimigos ou com aviões inimigos.
    - Saber a posição do jogador
        - Chama o objeto jogador e pede a sua posição
    - Saber a posição dos inimigos
        - Chama os objetos inimigos e pede a sua posição
    - Saber a posição do projétil
        - Chama o objeto projétil e pede a sua posição
    - Procura sobreposição da posição dos objetos jogador, inimigos e projéteis ou bombas.
        - Repassa a informação de colisão, caso aconteça colisão

-Verificar chamada de menu Pause
- Receber input do mouse
    - Saber posição do ponteiro do mouse no momento do input do mouse
- Comparar a posição do ponteiro com o botão do menu Pause

- Chamar menu Pause, caso o botão do menu contenha a posição do mouse
    - Congelar o gameplay e disponibilizar na janela de exibição do jogo o menu Pause

- Verificar chamada do menu da Loja
    - Saber posição do jogador
        - chamar Jogador e pedir a sua posição
    - Saber posição da base
    - Verificar condição de pouso do jogador
        - Pedir velocidade para Jogador e pontos de vida
    - Se confirmado a condição de pouso, chamar menu da Loja

Salvo

Descrição: Representa o jogo salvo

Responsabilidades	**Colaborações**
Saber o seu título	
Saber o seu progresso	
Carregar arquivo	
Escrever arquivo	

Comentário:

-Saber o seu título ou nome
- Atribuir o nome do arquivo com o mesmo nome que o jogador escolher para salvar o jogo

-Saber o seu progresso
- Armazenar os atributos do jogador e quais missões estão completas

-Carregar arquivo
- Buscar o arquivo no diretório correto
- Abrir o arquivo
- Atribuir os atributos do jogador de acordo com o que foi salvo e atribuir missão completa de acordo com o que foi salvo

-Escrever arquivo
- Criar arquivo no diretório correto com o nome do arquivo sendo o título e guardar o progresso

Jogador	
Descrição: Representa o jogador	
Responsabilidades	**Colaborações**
Saber o seu nome	
Saber sua quantidade de pontos de vida	Pontos de vida, aviao
Saber o seu progresso na missão	Missao
Saber quantos pontos possui	Simulador
Saber realizar manobra para inverter a direção de voo	
Atirar	Armas
Saber qual é o seu avião	
Saber quais são as suas armas	
Lançar foguete e bombas	Projétil
Voar	Avião
Controlar manetes, manche e flaps	
ligar ou desligar o motor	
acionar trem de pouso	
Comentário:	

GerenciadorDeMissao	
Descrição: Representa o gerador de missão	
Responsabilidades	**Colaborações**
Saber qual a missão atual	

Saber o progresso da missão	Simulador
Parabenizar o jogador quando concluir a missão com texto e som	Audio
Saber o som de parabenização	

Comentário:

-Saber qual a missão atual
- Verificar qual botão-missão o jogador selecionou
- Chamar a missao correspondente ao botão selecionado
- Acessar qual é a missao

-Saber o progresso da missão
- Contabilizar os inimigos alvo da missão que forem abatidos
    - Chamar o Simulador para saber quantos inimigos alvos forem abatidos

-Parabenizar o jogador quando concluir a missão com texto e som
- Acessar Saber o progresso da missão
    - Se a missão estiver completa, emitir som de parabenização e enviar um texto para ser exibido na tela do jogador avisando-o para voltar a base

- Saber o som de parabenização
- Saber o diretório em que está o som de parabenização

Missao

Descrição: Representa a missão

Responsabilidades	**Colaborações**
Saber qual é a missão	
Saber a música de fundo	
Saber Estado da missão (feito, não-feito)	Jogador
Saber Disponibilidade da missão	
Saber quantidade de inimigos	
Saber quais inimigos	
Saber cenário de fundo	cenario

Comentário:

-Saber qual é a missão

- Definição pelos desenvolvedores de um texto especificando a missão

-Saber a música de fundo
- Saber o diretório em que está a música de fundo
- Saber o nome do arquivo da música de fundo

-Saber Estado da missão
- Inicialmente o Estado da missão é não-feito, e após o gerenciador de missão parabenizar o jogador, o Estado da missão muda para feito

-Saber Disponibilidade da missão
- A missão anterior deve ser completada para esta estar disponível.
- Exceção: a primeira missão

-Saber a quantidade de inimigos
- Os desenvolvedores definem quantos de cada tipo de inimigo há na missão

-Saber quais inimigos
- Os desenvolvedores definem quais inimigos estarão participando dessa missão

-Saber cenário de fundo

- Saber qual o diretório está a imagem
- Saber o nome do arquivo da imagem

Arma

Descrição: Representa armas

Responsabilidades	**Colaborações**
Saber a sua imagem	
Saber o seu projétil	Projétil
Disparar	
Saber o seu som	Audio
Saber quem está portando a arma	

Comentário:

-Saber a sua imagem
- saber o diretório onde está o seu arquivo da sua imagem
    - saber o nome do arquivo da sua imagem

-Saber o seu som
- saber o diretório onde está o arquivo do seu som
    - saber o nome do arquivo do som

-Saber o seu projétil
- saber o nome da classe projétil (concreta) que possui

-Disparar
- responder a chamada do jogador para o evento “atirar”
    - chamar a classe projétil (concreta) relacionada
        - Classe projétil se responsabiliza pelo restante

-Saber quem está portando a arma
- valor passado pelo objeto que contém a arma

Objeto: Metralhadora, Rpg, Bombardeiro

Projetil

Descrição: Representa o projétil disparado pelas armas

Responsabilidades	**Colaborações**
Saber sua imagem	
Saber o seu som	Audio
Obedecer a física de voo	simulador, renderizador
Obedecer a física de impacto	simulador, renderizador
Saber o seu dano causado nos pontos de vida	
Saber por quem foi disparado (jogador ou IA)	

Comentário:

-Saber sua imagem
- saber o diretório onde está o arquivo imagem
    - saber o nome do seu arquivo imagem

-Saber o seu som
- saber o diretório onde está o arquivo de som
    - saber o nome do seu arquivo de som

-Obedecer a física de voo
- Conter toda a lógica da física de vôo
- receber informação sobre posicionamento e direção do disparo
- calcular o posicionamento do projétil (cena por cena)
- enviar informações para o renderizador de projétil

-Obedecer a física de impacto
- Conter toda a lógica da física de impacto
- receber informação se houve impacto
- responder informação para o simulador sobre o dano causado
- enviar informações para renderizador de projétil quando houver impacto
- enviar informação para o audio quando houver impacto

-Saber o seu dano
- atribuição de valor no momento da inicialização do objeto

-Saber por quem foi disparado
- valor passado pela arma

Objetos: Calibre15, Calibre20, Calibre30, Foguete e Bomba

Aviao

Descrição: Representa o avião

Responsabilidades	**Colaborações**
Saber sua imagem	
Saber o seu som	Audio
Saber o seu time	
Voar	simulador, renderizador
Decolar	simulador, renderizador
Pousar	simulador, renderizador
Saber pontos de vida	Pontos de vida

Comentário:

-Saber sua imagem
- saber o diretório onde está o arquivo imagem
    - saber o nome do arquivo imagem

-Saber o seu som
- saber o diretório onde está o arquivo de som
    - saber o nome do arquivo de som

-Voar
- Sabe a lógica de vôo do avião
    - recebe informação sobre posicionamento inicial
    - recebe informação sobre comando do piloto (jogador ou IA)
    - calcula sua posição, velocidade e aceleração seguinte
    - envia essas informações para o renderizador e ou simulador

-Decolar
- Sabe a lógica de decolar do avião
    - recebe informação sobre o estado inicial (posição, velocidade,aceleração)
    - recebe informação sobre comando do piloto (jogador ou IA)
    - calcula o estado seguinte
    - envia essas informações para o renderizador e ou simulador

-Pousar
- Sabe a lógica de pousar do avião
    - recebe informação sobre o estado inicial (posição, velocidade,aceleração)
    - recebe informação sobre comando do piloto (jogador ou IA)
    - calcula o estado seguinte
    - envia essas informações para o renderizador e ou simulador

-Saber pontos de vida
- Atribuição de inicialização

Objetos: P-51, P-38, Spitfire

Vida

Descrição: Representa os pontos de vida

Responsabilidades	**Colaborações**
Saber pontos de vida máximo	
Saber pontos de vida atual	
Saber quando subtrair	Simulador
Saber quanto subtrair	Projétil

Comentário:

-Saber pontos de vida máximo
- Atribuição de inicialização do objeto ou contém no objeto derivado de aviao

-Saber pontos de vida atual
- Saber pontos de vida máximo
    - subtrair quando for atingido
    - subtrair a quantidade de acordo com o que for colidido
    - resetar para pontos de vida máximo quando iniciar missão

-Saber quando subtrair
- subtrair quando houver colisão com projétil e com objetos do cenário
    - receber essas informações do simulador

-Saber quanto subtrair
- subtrair de acordo com o que for colidir
    - receber essa informação do simulador

IA

Descrição: Representa a inteligência artificial dos inimigos

Responsabilidades	**Colaborações**
Localizar o alvo	
Perseguir o alvo	
Mirar no alvo	
Voar	Avião
Atirar	Armas

Comentário:

-Localizar o alvo
- receber informação da posição do inimigo (Jogador)

-Perseguir o alvo
- Ter a lógica de perseguição do alvo
    - receber localização do alvo
    - executar algoritmo de perseguição
    - executar comandos de movimento

-Mirar no alvo
- Ter a lógica de mira ao alvo
    - receber localização do alvo
    - saber para onde está mirando
    - executar algoritmo de mira
    - executar comando de mira

-Voar
- Se for avião,
    - Ter a lógica de vôo
        - conferir informações de vôo
        - calcular trajetória de vôo
        - executar comando de movimento
- Se não for avião,

- habilidade desabilitada

-Atirar
- Mirar
- executar comando de disparo de acordo com algoritmo implementado

AntiAereo

Descrição: Representa a torreta antiaérea inimiga

Responsabilidades	**Colaborações**
Saber sua posição fixa	
Ter sua IA	IA
Ser rotativa	
Mirar	IA
Atirar	IA
Ser destrutível	
Saber pontos de vida	Pontos de vida
Saber qual sua arma	
Saber sua imagem	
Recompensa (xp,gp)	

Comentário:

-Saber sua posição (fixa no cenário)
- atribuição de inicialização

-Ter sua IA
- Herda IA

-Ser rotativa
- propriedade de imagem

-Mirar
- Herda IA

-Atirar
- Herda IA

-Ser destrutível
- possuir pontos de vida
- pertencer ao gameplay
- Ser objeto de colisão

-Saber pontos de vida
- inicializa objeto pontos de vida

-Saber qual sua arma
- atributo de inicialização

-Saber sua imagem
- Saber diretório onde está o arquivo imagem
    - Saber nome do arquivo imagem

-Recompensa
- atributo de inicialização

AviaoInimigo

Descrição: Representa o avião inimigo

Responsabilidades	**Colaborações**
Saber sua posição inicial	
Ter sua IA	IA
Ser rotativa	
Mirar	IA
Atirar	IA
Voar	IA
Ser destrutível	
Saber pontos de vida	Pontos de vida
Saber qual sua arma	

Saber sua imagem	
Recompensa (xp,gp)	

Comentário:

-Saber sua posição (fixa no cenário)
- atribuição de inicialização

-Ter sua IA
- Herda IA

-Ser rotativa
- propriedade de imagem

-Mirar
- Herda IA

-Atirar
- Herda IA

-Voar
- Herda IA

-Ser destrutível
- possuir pontos de vida
- pertencer ao gameplay
- Ser objeto de colisão

-Saber pontos de vida
- inicializa objeto pontos de vida

-Saber qual sua arma
- atributo de inicialização

-Saber sua imagem
- Saber diretório onde está o arquivo imagem
    - Saber nome do arquivo imagem

-Recompensa
- atributo de inicialização

EdificioInimigo

Descrição: Representa a edificação inimiga

Responsabilidades	**Colaborações**
Saber sua posição fixa	
Ser destrutível	
Saber pontos de vida	Pontos de vida
Saber sua imagem	
Recompensa (xp,gp)	

Comentário:

-Saber sua posição (fixa no cenário)
- atribuição de inicialização

-Ter sua IA
- Herda IA

-Ser rotativa
- propriedade de imagem

-Mirar
- Herda IA

-Atirar
- Herda IA

-Ser destrutível
- possuir pontos de vida
- pertencer ao gameplay
- Ser objeto de colisão

-Saber pontos de vida
- inicializa objeto pontos de vida

-Saber qual sua arma
- atributo de inicialização

-Saber sua imagem
- Saber diretório onde está o arquivo imagem
    - Saber nome do arquivo imagem

-Recompensa
- atributo de inicialização

Item	
Descrição: Representa os itens vendidos na loja	
Responsabilidades	**Colaborações**
Saber sua imagem	
Saber o seu som	Audio
Saber seu nome	
Saber qual objeto representa (qual avião ou arma)	
Saber se está disponível para compra	Jogador
Saber quanto custa	
Saber se já foi comprado ou não	
Saber sua posição na loja	
Ter um botão de compra	Botao

Comentário:

-Saber sua imagem
- Saber o diretório onde está o arquivo imagem
    - saber o nome do arquivo imagem

-Saber seu som
- saber o diretório onde está seu arquivo de som
    - saber o nome do arquivo de som

-Saber seu nome
- atribuição de inicialização

-Saber qual objeto representa
- atribuição de inicialização

-Saber se está disponível para compra
- saber requerimento de experiência
    - comparar com a xp do jogador
        - Se a xp do jogador for igual ou maior
            - item habilitado para compra
        - Caso contrário
            - item desabilitado para compra

-Saber quanto custa
- atribuição de inicialização

-Saber se já foi comprado ou não
- inicializado como não (0)
    - muda para sim (1) quando comprado

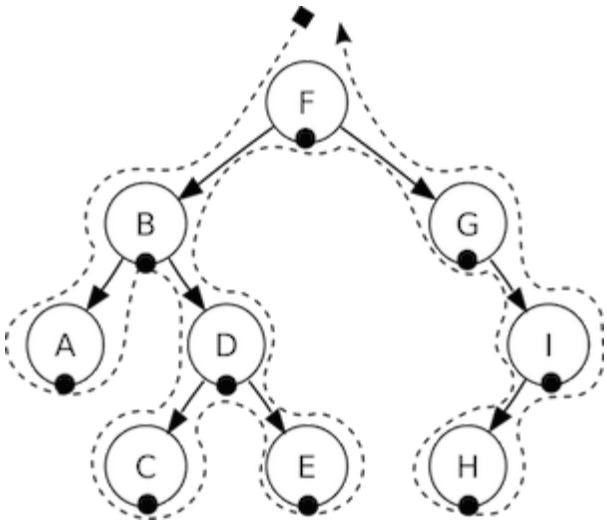
-Saber sua posição na loja
- atributo de inicialização

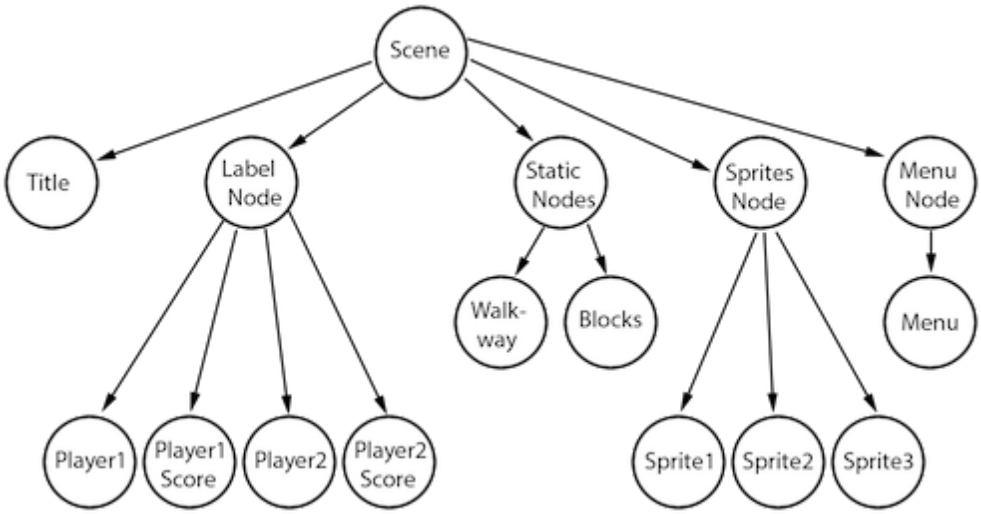
-Ter um botão de compra
- criar objeto da classe botao

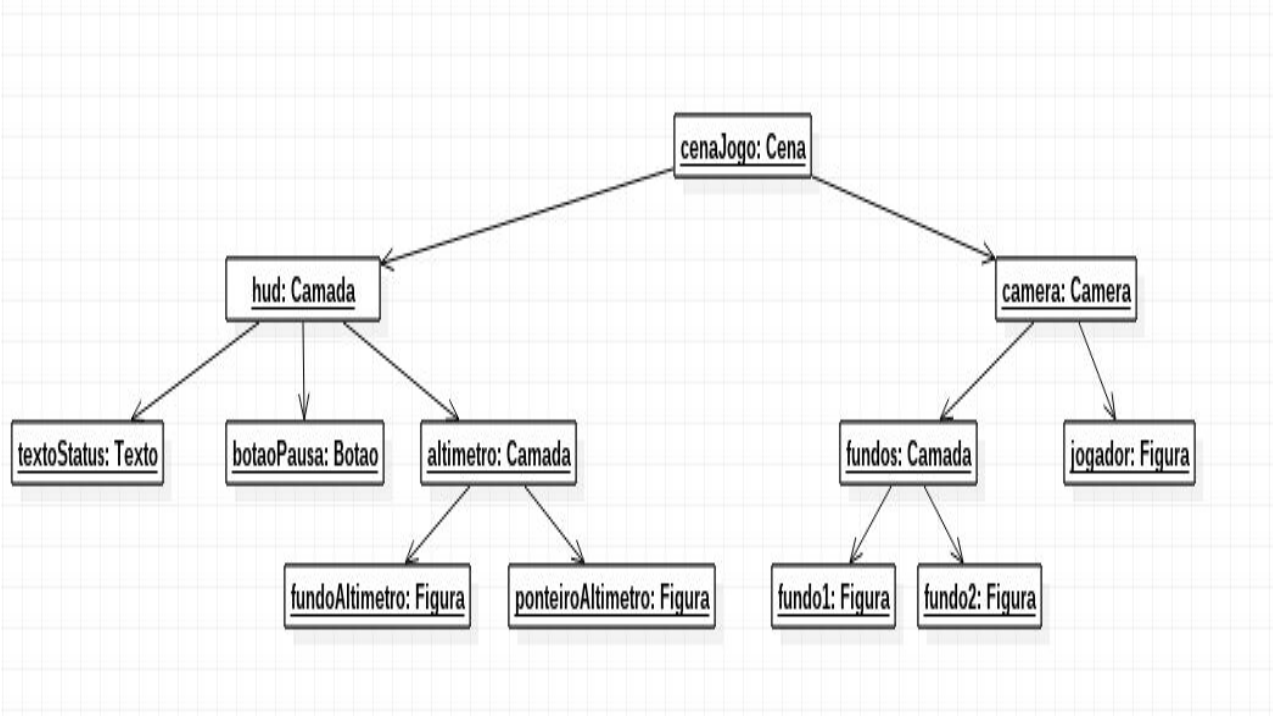
Propagação de Eventos

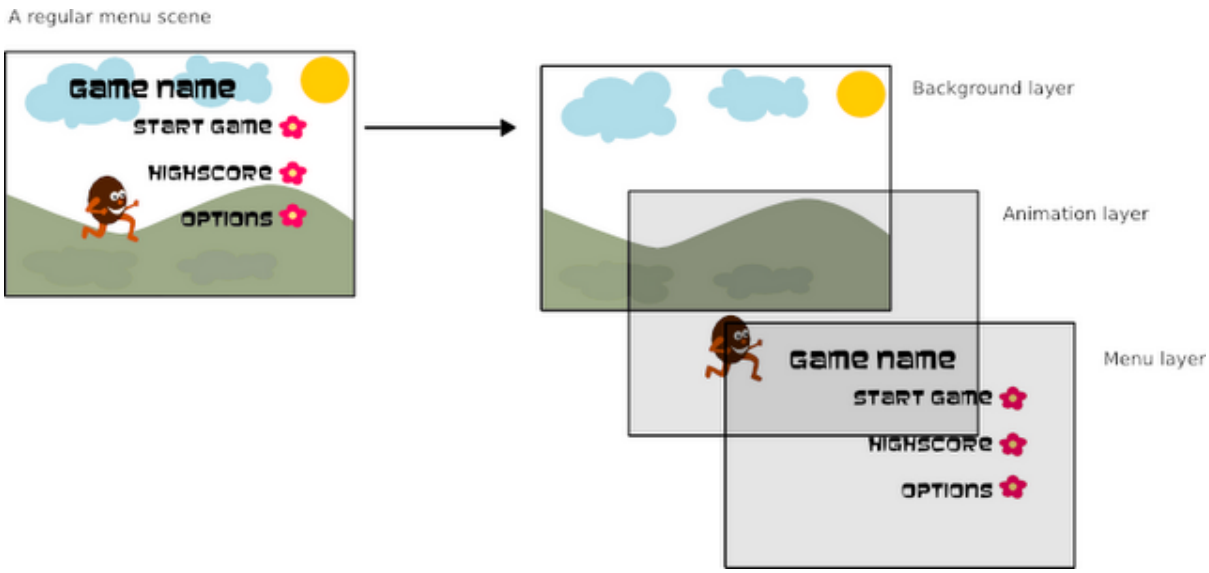
A propagação de eventos se dará na , conforme a figura abaixo:

The most basic style of event system is the 'bag of handler methods', which is a simple implementation of the Observer pattern. Basically, the handler methods (callables) are stored in an array and are each called when the event 'fires'.






cenaJogo: Cena
hud: Camada
camera: Camera
textoStatus: Texto
botaoPausa: Botao
altimetro: Camada
fundos: Camada
jogador: Figura
fundoAltimetro: Figura
ponteiroAltimetro: Figura
fundo1: Figura
fundo2: Figura


A regular menu scene
Game name
START GAME
HIGHSCORE
OPTIONS
Background layer
Animation layer
Menu layer
Game name
START GAME
HIGHSCORE
OPTIONS

EXEMPLO DE gameloop from How to Think Like a Computer Scientist: Learning with Python 3

```
1  import pygame
2
3 def main():
4         """ Set up the game and run the main game loop """
5         pygame.init()         # Prepare the pygame module for use
6         surface_sz = 480    # Desired physical surface size, in pixels.
7
8        # Create surface of (width, height), and its window.
9        main_surface = pygame.display.set_mode((surface_sz, surface_sz))
10
11       # Set up some data to describe a small rectangle and its color
12       small_rect = (300, 200, 150, 90)
13       some_color = (255, 0, 0)       # A color is a mix of (Red, Green, Blue)
14
15       while True:
16                ev = pygame.event.poll()       # Look for any event
17                if ev.type == pygame.QUIT:    # Window close button clicked?
18                        break                        # ... leave game loop
19
20               # Update your game objects and data structures here...
21
22               # We draw everything from scratch on each frame.
23               # So first fill everything with the background color
24               main_surface.fill((0, 200, 255))
25
26                # Overpaint a smaller rectangle on the main surface
27                main_surface.fill(some_color, small_rect)
28
29               # Now the surface is ready, tell pygame to display it!
30                pygame.display.flip()
31
32       pygame.quit()             # Once we leave the loop, close the window.
33
34 main()
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Mar 27 08:25:06 2017

@author: gabrui
"""

import pygame

pygame.init()



class Aux:
	"""Classes com funções auxiliares para mexer com listas etc..."""
	@staticmethod
	def removeTuplas1Elem(lista, elem):
	i = 0
	while i < len(lista):
	if lista[i][0] == elem:
		del lista[i]
	i += 1

	@staticmethod
	def existeTupla1Elem(lista, elem):
	for tupla in lista:
	if tupla[0] == elem:
		return True




class Evento:
	"""É uma classe que todo objeto que registra eventos deve ter uma instância"""
```

```
_escutaveis = [] #É uma lista de tuplas (stringEvento, funcao_de_chamada) APARADOS
_disparados = []  #É uma lista de tuplas (stringsEventos, objeto_do_evento) LANÇADOS

def adicionaEscutavel(this, string_evento, callback):
this._escutaveis.append((string_evento, callback))

def removeEscutavel(this, string_evento, callback = None):
if callback is None:
Aux.removeTuplas1Elem(this._escutaveis, string_evento)
else:
this._escutaveis.remove((string_evento, callback))

def adicionaDisparo(this, string_evento, objeto_do_evento):
if Aux.existeTupla1Elem(this._escutaveis, string_evento):
this._disparados.append((string_evento, objeto_do_evento))

def removeDisparo(this, string_evento, objeto_do_evento = None):
if objeto_do_evento is None:
Aux.removeTuplas1Elem(this._disparados, string_evento)
else:
this._disparados.remove((string_evento, objeto_do_evento))

def removeTodosDisparos(this):
del this._disparados[:]

def escuta(this, evento):
"""Executa as funções de escuta dado os disparos de outro evento"""
```

```
        for escutavel, callback in this._escutaveis:
        for disparo, objeto_do_evento in evento._disparados:
                if escutavel == disparo:
                callback(objeto_do_evento)

        def fala(this, evento):
        """Fala para outro evento o que você disparou, e se ele não souber responder,
        ele pega para ele o que você falou"""
        for disparo, objeto_do_evento in this._disparados:
        escutou = False
        for escutavel, callback in evento._escutaveis:
                if escutavel == disparo:
                callback(objeto_do_evento)
                escutou = True
        if not escutou:
                evento.adicionaDisparo(disparo, objeto_do_evento)
        this.removeDisparo((disparo, objeto_do_evento))



# Devaneios geométricos
class Ponto:
        """Classe que representa um ponto 2d do tipo (x, y)"""
        def __init__ (this, x, y):
        this._x = x
        this._y = y


        def setXY(this, x, y):
        this._x = x
        this._y = y


        def setX(this, x):
        this._x = x
```

```
    def setY(this, y):
    this._y = y


    def getX(this):
    return this._x


    def getY(this):
    return this._y


    def getXY(this):
    return (this._x, this._y)


    def distancia2(this, ponto):
    return (this._x-ponto._x)*(this._x-ponto._x) + (this._y-ponto._y)*(this._y-ponto._y)


    def distancia(this, ponto):
    from math import sqrt
    return sqrt(this.distancia2(ponto))



class Retangulo:
    def __init__(this, x, y, largura, altura):
    this._p1 = Ponto(x, y)
    this._p2 = Ponto(x + largura, y + altura)
```

```
class Angulo:
        """Classe que cuida dos ângulos, que devem estar entre pi e -pi"""
        def __init__ (this, angulo, estaEmRadianos = True):
        if estaEmRadianos:
        this._angulo =  angulo
        else: #Suponho que esteja em graus
        this._angulo = Angulo.grausParaRadianos(angulo)
        this._validaAngulo()


        @staticmethod
        def grausParaRadianos(angulo):
        from math import radians
        return radians(angulo)


        @staticmethod
        def RadianosParaGraus(angulo):
        from math import degrees
        return degrees(angulo)


        def _validaAngulo(this):
        from math import pi
        while this._angulo <= -pi:
        this._angulo += 2*pi
        while this._angulo > pi:
        this.angulo -= 2*pi


        def getAngulo(this):
        return this._angulo


        def setAngulo(this, angulo):
```

```
        this._angulo = angulo
        this._validaAngulo()




class Renderizavel:
        even = Evento()
        pos = Ponto(0, 0)
        centro = Ponto(0, 0)
        escala = Ponto(1, 1)
        retang = Retangulo(0, 0, 0, 0)
        rot = Angulo(0)
        opacidade = 1
        cor = (255, 255, 255)

        def atualiza(this, dt):
        pass




class Figura(Renderizavel):

        def __init__(this, string_imagem):
        this.string_imagem = string_imagem




class Texto(Renderizavel):

        def __init__(this, string_texto, tupla_fonte):
        this.string_texto = string_texto
        this.tupla_fonte = tupla_fonte
```

```
class Camada(Renderizavel):
        filhos = []


        def adicionaFilho(this, filho):
        this.filhos.append(filho)


        def removeFilho(this, filho):
        this.filhos.remove(filho)


        def atualiza(this, dt):
        for filho in this.filhos:
        filho.atualiza(dt)


        def _propagaEventoDeCimaParaBaixo(this, evento):
        this.even.escuta(evento)
        for filho in this.filhos:
        if type(filho) is Camada:
                filho._propagaEventoDeCimaParaBaixo(evento)
        else:
                filho.even.escuta(evento)


        def _propagaEventoDeBaixoParaCima(this):
        for filho in this.filhos:
        if type(filho) is Camada:
                filho._propagaEventoDeBaixoParaCima()
        filho.even.fala(this.even)
```

```
        def _observaFilhos(this, estado, callbacks):
        for filho in this.filhos:
        if type(filho) is Camada:
                filho._observaFilhos(filho._transforma(estado), callbacks)
        elif type(filho) is Figura:
                pass
        elif type(filho) is Texto:
                pass


        def _transforma(this, estado):
        """Converte as coordenadas e transformações de um Renderizável para o seu"""
        pass




class Cena(Camada):
        """Classe que representa a cena do jogo, no qual existem as camadas e objetos
        renderizáveis. Ela é responsável pela propagação de eventos. Se comunica com
        a Entrada, com o Audio e com o Renderizador. """

        def __init__ (this, audio, entrada, renderizador):
        """Precisa-se da referência aos objetos de Audio, Entrada e Renderizador"""
        this.audio = audio
        this.entrada = entrada
        this.renderizador = renderizador


        def atualiza(this, dt):
        """Propaga o loop do jogo, sabendo o intervalo de tempo dt transcorrido"""
        this.entrada.atualiza()
        this.even.removeTodosDisparos()
```

```
        this.entrada.even.fala(this.even)
        this._propagaEventoDeCimaParaBaixo(this.even)
        this.even.removeTodosDisparos()
        this._propagaEventoDeBaixoParaCima()
        this._propagaEventoDeCimaParaBaixo(this.even)

        this.renderizador.iniciaQuadro()

        this.renderizador.finalizaQuadro()




class Entrada:
        """Classe que faz interface com o pygames e registra todos os eventos de entrada"""
        even = Evento()

        def __init__(this):
        pass



        def atualiza(this):
        """Atualiza os seus eventos"""
        this.verTeclado()
        this.verMouse()



        def verTeclado(this):
        """Observa quais teclas estão pressionadas e se está focado"""
        vazio = True
        for ide, val in enumerate(pygame.key.get_pressed()):
        if val == True:
                vazio = False
                this.even.adicionaRegistro("K_"+pygame.key.name(ide), None)
        if vazio:
```

```
        this.even.adicionaRegistro("K_vazio", None)
        if not pygame.key.get_focused():
        this.even.adicionaRegistro("K_desfocado", None)


        def verMouse(this):
        """Observa a posição do ponteiro e se clica"""
        pass



class Renderizador:
        _listaImagens = []

        def __init__(this, largura, altura, corFundo = (0, 0, 0)):
        this.tela = pygame.display.set_mode((largura, altura))
        this.corFundo = corFundo


        def iniciaQuadro(this):
        this.tela.fill(this.corFundo)


        def finalizaQuadro(this):
        pygame.display.flip()


        def desenhaImagem(this, string_imagem, posXY):
        this.tela.blit(this._bancoImagens(string_imagem), posXY)


        def desenhaTexto(this, string_texto, posXY, tuplaFonte = (None, 12)):
        texto = pygame.font.Font(tuplaFonte[0], tuplaFonte[12])
        this.tela.blit(texto, posXY)
```

```
def _bancoImagens(this, string_imagem):
imagem = this._listaImagens.get(string_imagem)
if imagem == None:
import os
caminho = string_imagem.replace('/', os.sep).replace('\\', os.sep)
imagem = pygame.image.load(caminho)
this._listaImagens[string_imagem] = imagem
return imagem
```