## Game

**Responsabilidades:** Executar o loop principal do jogo, controlar a situação da partida, processar a entrada do teclado, iniciar e reiniciar as partidas e integrar Scenario com o módulo de jogadores.

**Métodos:**
void run(const std::string& apelido): Inicia o loop principal da partida, executando cenário, entrada e renderização.
void handleInput(): Captura eventos do teclado.
void showGameOver(): Exibe a tela de fim de jogo e aguarda uma entrada para iniciar.
void startNewRound(): Reinicializa o cenário e dados da rodada atual mantendo o jogador.
void savePlayerData(): Salva os dados atualizados do jogador no final da partida;

**Colaboração:** Scenario e PlayerManager.

## Bird

**Responsabilidades:** Representar o pássaro controlado pelo jogador.

**Métodos:**
void update() override: Aplica gravidade ao pássaro e atualiza a posição.
void draw(ALLEGRO_DISPLAY* display): Desenha o pássaro na tela com animação de batida de asas.
bool checkCollision (const GameObject& other) const override: Detecta colisão com canos ou cenário.
void jump(): Reage ao pressionamento da tecla espaço (pula).
void reset(): Retorna o pássaro a posição inicial e zera variáveis de estado.

**Colaboração:** Scenario, e herda de GameObject.

## Scenário

**Responsabilidades:** Gerenciar o cenário do jogo (desenho, colisões, pontuação, dificuldade).

**Métodos:**
void update(): Atualizar o estado dos objetos do jogo (pontuação, movimento do pássaro, colisões, canos).
void draw(): Desenha todos os objetos visuais do jogo na tela.
void reset(): Reinincia o estado do cenário para uma nova partida.
void addPipe(): Cria novos pares de canos e os adiciona à lista de obstáculos.
bool checkCollision(): Verifica se houve colisão entre o pássaro e algum obstáculo ou limite.
void serDifficulty(float gravity, float pipeSpeed): Configura as regras da partida.
int getScore() const: Retorna a pontuação atual do jogador.

**Colaboração:** Bird e Pipe.

## Pipe

**Responsabilidade:** Representar os obstáculos vertiais que se movimentam da direita para a esquerda, colidem com o pássaro e controlam a pontuação.

**Métodos:**
void update() override: Move o cano para a esquerda em velocidade constante.
void draw(ALLEGRO_DISPLAY* display) override: Desenha o cano na tela.
bool checkCollision(const GameObject& other) const override: Verifica a colisão com o Bird.
bool isOffScreen() const: Indica se o cano saiu completamente da tela.
bool isTop() const: Indica se o cano atual é superior (invertido verticalmente).

**Colaboração:** Scenario, e herda de GameObject.

## GameObject

**Responsabilidades:** Servir como base para todos os elementos do jogo que têm posição, movimentação, colisão e renderização. Fornece interface polimórfica para atualização e desenho.

**Métodos:**
virtual void update() = 0: Atualiza posição ou estado interno do objeto.
virtual void draw (ALLEGRO_DISPLAY* display) = 0): Desenha o objeto na tela.
virtual bool checkCollision(const GameObject& other) const = 0: Verifica se este objeto colidiu com outro.
float getX() const: Retorna a posição X atual do objeto no cenário.
float getY() const: Retorna a posição Y atual do objeto no cenário.

**Colaboração:** Superclasse abstrata para Bird e Pipe.

## Player

**Responsabilidades:** Representar um jogador do sistema, armazenando suas informações básicas e estatísticas do jogo.

**Métodos:**
std::string getApelido() const: Retorna o apelido do jogador.
std::string getNome() const: Retorna o nome completo do jogador.
int getPontuacaoMax() const: Retorna a maior pontuação feita em uma única partida.
int getPontuacaoTotal() const: Retorna a soma de todas as pontuações do jogador.
int getNumPartidas() const: Retorna o número total de partidas jogadas.
void adicionarPontuacao(int score): Atualizar os dados estatísticos do jogador com uma nova pontuação.
std::string toString() cons: Retorna uma string com todas as informações do jogador formatadas para exibição.

**Colaboração:** PlayerManager.

## PlayerManage

**Responsabilidades:** Gerenciar os jogadores cadastrados no sistema.

**Métodos:**
bool registrarPlayer(const std::string& nome, const std::string& apelido): Adiciona um novo jogador ao sistema.
bool removerPlayer(const std::string& apelido): Remove um jogador pelo apelido.
Player* buscarPlayer(const std::string& apelido): Retorna um ponteiro para o jogador com o apelido informado (ou nullptr).
void atualizarPontuacao(const std::string& apelido, int score): Atualiza as estatísticas do jogador com nova pontuação.
Player* getTopPlayer() const: Retorna o jogador com maior pontuação máxima registrada.
void salvarEmArquivo() const: Grava todos os dados dos jogadores em um arquivo.
void carregarDeArquivo(): Lê os dados do arquivo e reconstrói os objetos Player.
void listarJogadores() const: Exibe todos os jogadores cadastrados.

**Colaboração:** Player.