URIAH 10

A BATALHA DE CENTAURO

***

NOVO_JOGADOR JOGAR BATALHAS RECURSOS

Requisitos:

O Universo:

Em um universo 2D, dois jogadores realizarão uma batalha com as suas naves estelares. O universo apresenta as naves e o menu de opções para os jogadores. O universo pode possui uma imagem de fundo, que represente o cenário da batalha.

Os jogadores:

Os jogadores possuem nome, vitórias, derrotas, quantidade de disparos bem sucedidos, quantidade de naves perdidas, além de seu exercício de naves e seus recursos. Os recursos ajudam os jogadores a comprarem as naves. Quando se vence uma batalha, o jogador adquire novos recursos. Um novo jogador inicia suas atividades com um nível padrão de recursos. Quando perde uma batalha, o jogador perde os recursos em naves. O jogador que ganhar, receberá uma certa quantidade de recursos. Um jogador deve estar presente na batalha, pilotando uma das naves. Se a sua nave for abatida, o jogo termina imediatamente e o jogador é eliminado do Universo. Os jogadores podem realizar as seguintes ações: mover uma nave, atirar com uma nave, ligar a mira de uma nave e fugir. O jogador que eliminar as naves adversárias primeiro vence a batalha. Um jogador pode fugir da batalha a qualquer momento, para não morrer e salvar os seus recursos empregados na batalha.

Naves :

As naves são de dois tipos: cruzadoras e interceptoras. As naves interceptoras possuem mais agilidade, são menores, e de pequeno raio de ação. As naves cruzadoras são maiores, mais lentas e de longo alcance. As naves possuem integridade e são comandadas pelos jogadores. Cada nave possui um número de identificação único, além de um jogador.

As naves são formadas por uma nuvem de pontos, conectados de tal forma a se obter um polígono. As naves possuem um ponto central que definem suas posições no espaço, e um vetor direção. Quando uma nave é abatida a sua imagem piscará na tela 3 vezes antes de desaparecer. Quando uma nave ataca, ela pode utilizar um disparo com mira (grid polar), mas isso implica em uma ação. As naves podem possuir uma imagem associada para visualização.

As naves Cruzadoras necessitam de três pontos de recurso para serem criadas, enquanto as Interceptadoras necessitam de 1 ponto. As Cruzadoras possuem uma distribuição de atributos distinta das Interceptadoras. As naves também possuem integridades diferentes. As Cruzadoras podem resistir a três disparos antes de morrerem, enquanto as Interceptadoras só resistem a um disparo. As Cruzadoras possuem campo de força, que se desliga apenas depois da eliminação de suas três naves protetoras (três naves secretas do tipo Interceptadora).

Disparos:

Quando uma nave dispara, o jogador define dois parâmetros: Ψ e Δ. O primeiro parâmetro determina o ângulo do ganhão da nave (em relação ao seu vetor posição) enquanto o segundo define a distância do disparo em porcentagem [0 – Alcance máximo]. O disparo gera um ponto a mais no universo e este ponto gera um segmento de reta, ou seja, um laser. Este laser tem origem na nave atacante e fim no ponto do disparo. Se o ponto final do disparo estiver dentro de outra nave, a mesma explodirá. Caso contrário, nada ocorrerá além de uma mensagem de insucesso no ataque.

Banco de dados:

O jogo possuirá um banco de dados que armazenará as estatísticas dos jogadores. Sempre que uma nova batalha se inicia, é possível carregar um jogador. O jogador decidirá com quantas naves lutará (limitado a uma quantidade máxima preestabelecida). O banco também armazenará as naves, que são únicas.

Dinâmica:

Quando a batalha se inicia, as naves são posicionadas pelos jogadores em duas modalidades: (1) às cegas e (2) de maneira sequencial. Depois de posicionadas, o universo checa colisões entre as naves para realizar a primeira eliminação. Uma eliminação retira a nave do banco de dados permanentemente. A batalha prossegue alternando as vezes de cada jogador, que decidirá por uma ação no seu turno. Cada jogador deverá se decidir contra um relógio, como no xadrez. O jogador que esgotar o seu tempo, terá que atuar dentro de 3 segundos apenas, perdendo a vez se não conseguir. Cada jogador decide sua ação em linha de comando, conforme os exemplos:

#n12.a30r5	**movimentar a nave 12 na direção de 30 graus por uma distância de 5 unidades.**
#n43.a45r12	**atacar com a nave 43 na direção de 45 graus com raio de 12 unidades.**
#out	**fugir da batalha**
#n15g	**ligar o grid da nave 15**

Depois que um grid é ligado o mesmo fica aparecendo na tela. Se uma nave ataca com grid ligado, seu grid é automaticamente desligado.

Os jogadores saberão sobre as ações no terminal, que imprimirá seus nomes com cores diferentes (azul e vermelho).

NOME: #n45.a50r4

NOME: #out

Ao final da batalha, o registro de todas as ações será gravado em um arquivo, para consulta posterior. Se um jogador não digita corretamente seus comandos, a tela é renovada para que o mesmo repita a operação, respeitando o seu tempo limite para ação.

Lista de substantivos e verbos:

Substantivos	Atribuição	Checagem
Universo	Classe	ok
Jogador (Player)	Classe	ok
Nave	Classe	ok
Menu (manu)	→ verbalizado	ok
Cenário e naves	Imagens	NA
Nome (Jogador)	Atributo do player	ok
Vitórias (Jogador)	Atributo do player	ok
Derrotas (Jogador)	Atributo do player	ok
Recursos (Jogador)	Atributo do player	ok
Quantidade de disparos bem sucedidos (Jogador)	Atributo do player	ok
Exército (Spacecrafts)	Atributo do player	ok
Batalha (Battle)	Classe	ok
Tipo (Nave): Interceptora e Cruzadora	Atributo da nave	ok
Integridade (Nave)	Atributo da nave	ok
Grid (Nave)	Atributo da nave	ok
Relógio (Universo)	Atributo da nave	ok
Identificador (Nave)	Atributo da nave	ok
Agilidade (Nave)	Atributo da nave	ok
Alcance (Nave)	Atributo da nave	ok
Polígono (Nave)	Classe	ok
Vetor (Nave)	Atributo	ok
Laser: Origem e Destino (Point)	Classe	ok
Disparo: Ψ e Δ	→ verbalizado	ok
Mensagem (println)	→ verbalizado	ok
Bando de dados: Battle	Classe	ok
Quantidade de naves por jogador (Batalha)	Atributo	ok
Colisão	→ verbalizado	ok
Linha de comando (Terminal)	Classe	ok
Ação	→ verbalizado	ok
Modalidade de jogo	Atributo	ok

Turno	→ verbalizado	ok
Cor	Atributo	ok
Frota	Classe	ok
Turno (round)	Atributo	ok
Universe	Classe	ok
Campo de força (shield)	Atributo	ok
Pilotagem (spacecraft)	Atributo	ok

***

Verbos	Atribuição	Checagem
Universo apresenta as naves (view) - Terminal	Membro	ok
Universo mostra as opções (menu)	Membro	ok
Universo inicia a batalha (fight)	Membro	ok
Jogador compra naves (buy)	Membro	ok
Jogador adquire recursos (earn) - Update()	Membro	ok
Jogador perde recursos (lose) - Update()	Membro	ok
Criar jogador	Membro	ok
Matar jogador	Membro	ok
Mover nave	Membro	ok
Atacar	Membro	ok
Fugir (Run)	Membro	ok
Ligar a mira	Membro	ok
Carregar campo de força	Membro	ok
Checar colisão	Membro	ok
Play (ação de jogador)	Membro	ok
Posicionar naves	Membro	ok
Timer	Membro	ok
ChatAction	Membro	ok
Verificar Grid (net)	Membro	ok
Save	Membro	ok

Detalhamento de requisitos e identificação de Classes:

1. Os Players receberam uma ação de morrer. Através desta ação os mesmos modificarão o status “alive”, que permitirá a sua permanência no universo.

2. O radar (grid) foi substituído por uma mira. As miras serão linhas desenhadas na direção do canhão das naves, cruzando todo o espaço.
3. As naves receberam um vetor de referência e um vetor para o canhão.
4. Vetores, são entidades matemáticas representadas por uma origem (ponto) e um ângulo. Os Vetores podem ou não serem desenhados no espaço (enable).
5. Os Pontos são entidades matemáticas representadas por duas coordenadas polares: ângulo e raio. Eles podem ou não serem desenhados no espaço (enable).
6. As naves possuirão uma função de dano, que vai decrementar suas integridades em casos de colisões.
7. O atributo “net” é uma variável que indicará se a mira da espaçonave está ligada ou desligada.
8. O atributo “power” é a variável que armazenará a força do disparo subsequente. Esta variável deverá ser ajustada sempre que o Player escolher atacar com a espaçonave.
9. A classe “Poligon” terá um membro capaz de checar se um ponto está dentro ou fora de seu perímetro.
10. O membro “print” fará a impressão das informações de classe no terminal. O membro “show” fará a plotagem da entidade no espaço.
11. O membro “rotate” realizará a rotação do Vetor em questão.
12. A classe “Interceptor” foi criada.
13. A classe “Cruiser” foi criada.
14. Cada espaçonave Cruzadora terá uma lista de naves protetoras de seu campo de força. Esta ação é executada pelo membro “setFleet()” do Player.
15. Em cada batalha, será registrada a quantidade de naves, a quantidade de rodadas, e as ações dos jogadores. Após as batalhas, o jogo deverá gravar um arquivo para conferência posterior.
16. As ações dos jogadores serão tratadas antes de serem validadas, por uma operação denominada chatAction(). Em cada batalha será registrado o apelido do vencedor (Red Eagle, Blue Lizard, etc).
17. As regras físicas do jogo pertence ao Universo. Os jogadores realizarão suas ações e o Universo tratará de validar as ações e suas consequências. Propriedades tais como gravidade, colisões, atritos envolvidos, dentre outras que se fizerem necessárias. Ou seja, o Universo realizará as alterações no exercito de cada jogador.
18. O Universo possui uma Tela (Screen), que será responsável pela apresentação dos elementos gráficos que estiverem habilitados.
19. Os jogadores possuem um status de jogo, para o Universo saber se um jogador está na batalha ou se o mesmo fugiu, ou ainda se morreu.
20. O Universo possui um Terminal (Terminal) para receber as ações dos jogadores, apresentá-las aos mesmos, analisar as ações e executá-las no Universo.
21. A Tela (Screen) também é responsável pelos áudios do jogo: explosão, tema da batalha, movimento, laser e fuga (radio).
22. As ações serão representadas por um modelo chamado Action. Cada modelo terá sua Action que, no final da batalha, será registrada no banco de dados.

Esta etapa de detalhamento dos requisitos é importante para refinar a lista de substantivos e verbos. É interessante realizar esta etapa em paralelo com o desenho das classes. O diagrama de classes pode ficar para depois da conclusão desta etapa. Ainda assim, é possível que novos atributos e membros sejam inseridos no projeto, durante a confecção dos demais diagramas, o que é muito natural. É melhor você voltar e modificar os mapas do que você voltar no projeto para modificar códigos. Ao final da confecção dos mapas propostos, o código poderá ser realizado com mais assertividade e rapidez.

Viewers

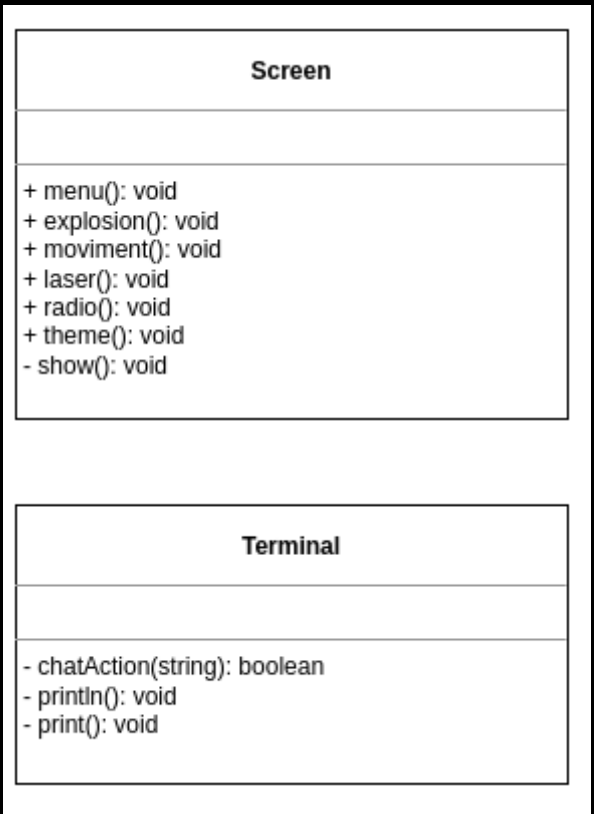

Figura 1 – Models.

Controllers

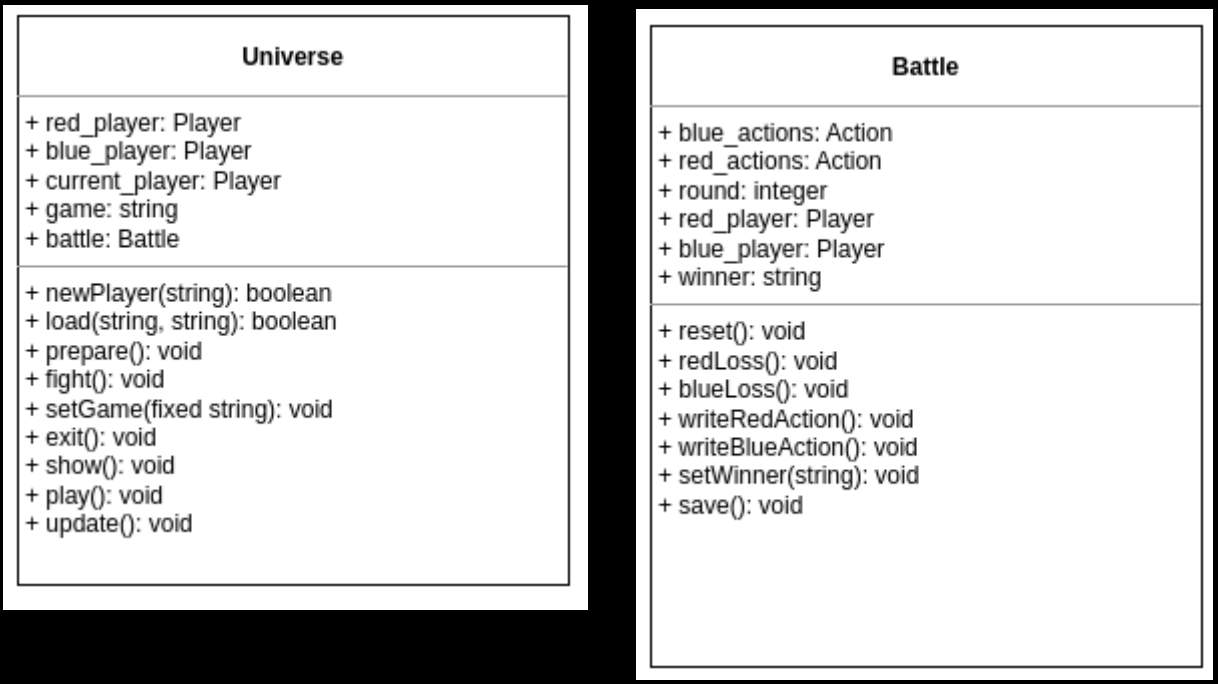

Figura 2 – Controllers.

Models

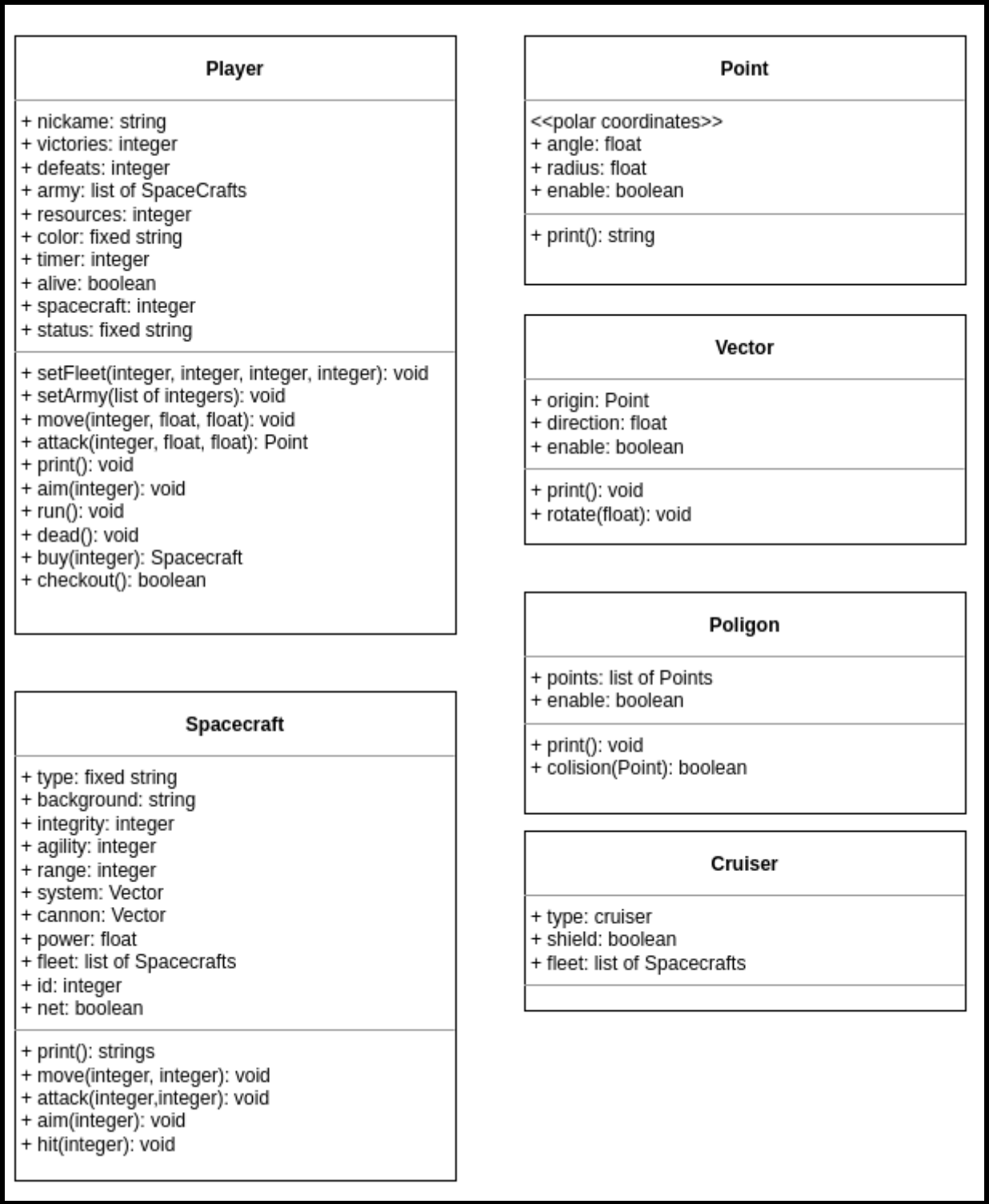

Figura 3 – Models.

No modelo MVC, as classes são desenhadas para cumprirem missões específicas e bem definidas dentro do sistema. As classes controladoras realizam a conexão das entradas dos usuários com os modelos e o banco de dados. As classes visualizadoras apresentam as informações e executam as interações entre os usuários externos e o sistema como um todo. Os modelos guardam as informações preciosas do sistema, sendo estas os objetos de interesse para o banco de dados.’

Diagrama de Classes

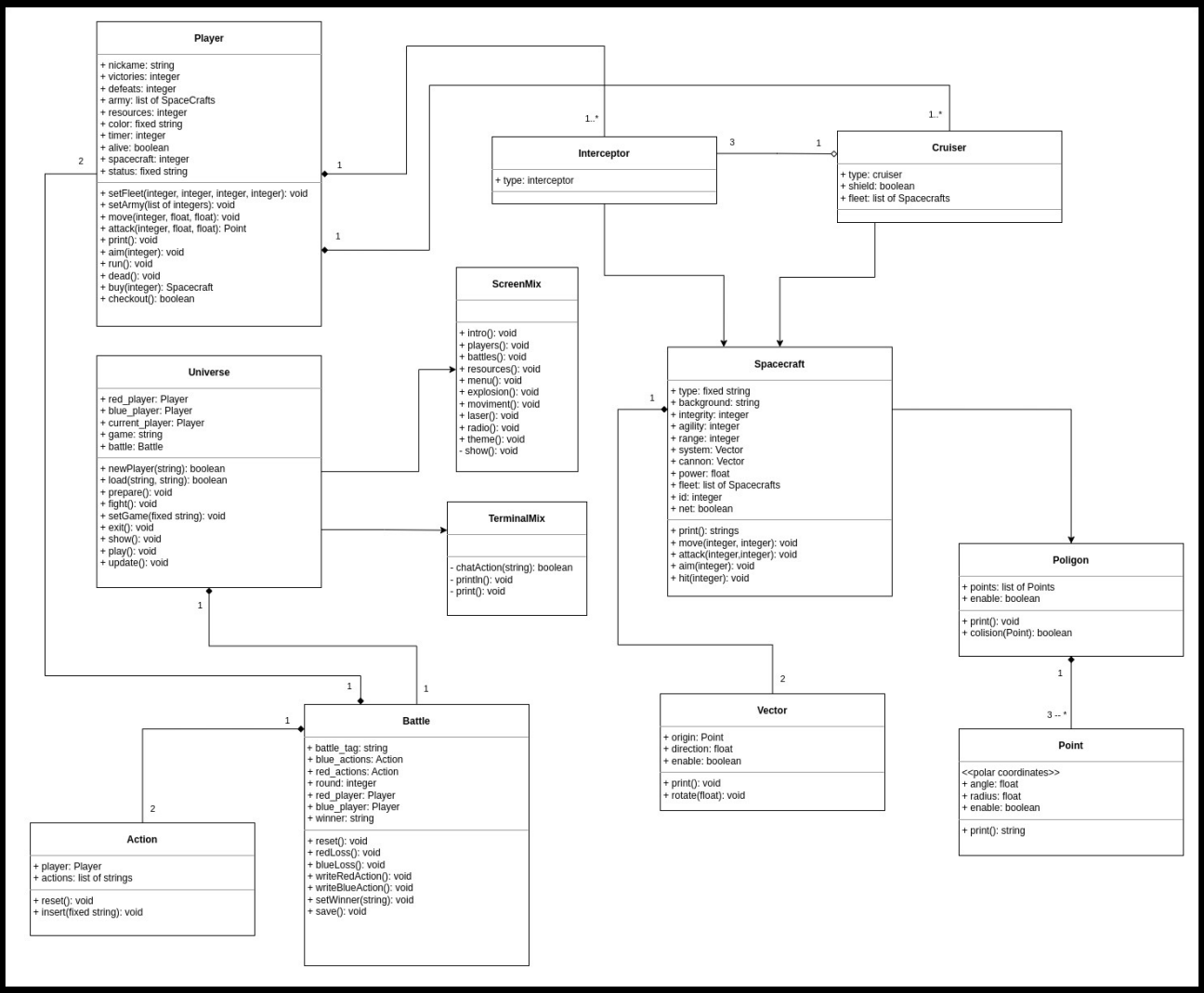

Figura 4 – Diagrama de classes.

Tabela de Construtores

Classe	Construtor
Universo	vazio
Player	(nickname, resources)
Battle	(Action, Action, Player, Player)
Action	vazio
Terminal	Mixin
Screen	Mixin
Spacecraft	(type, background, points)
Interceptor	(type)
Cruiser	(type, fleet)
Vector	(point, point)

Diagrama de Caso de Uso

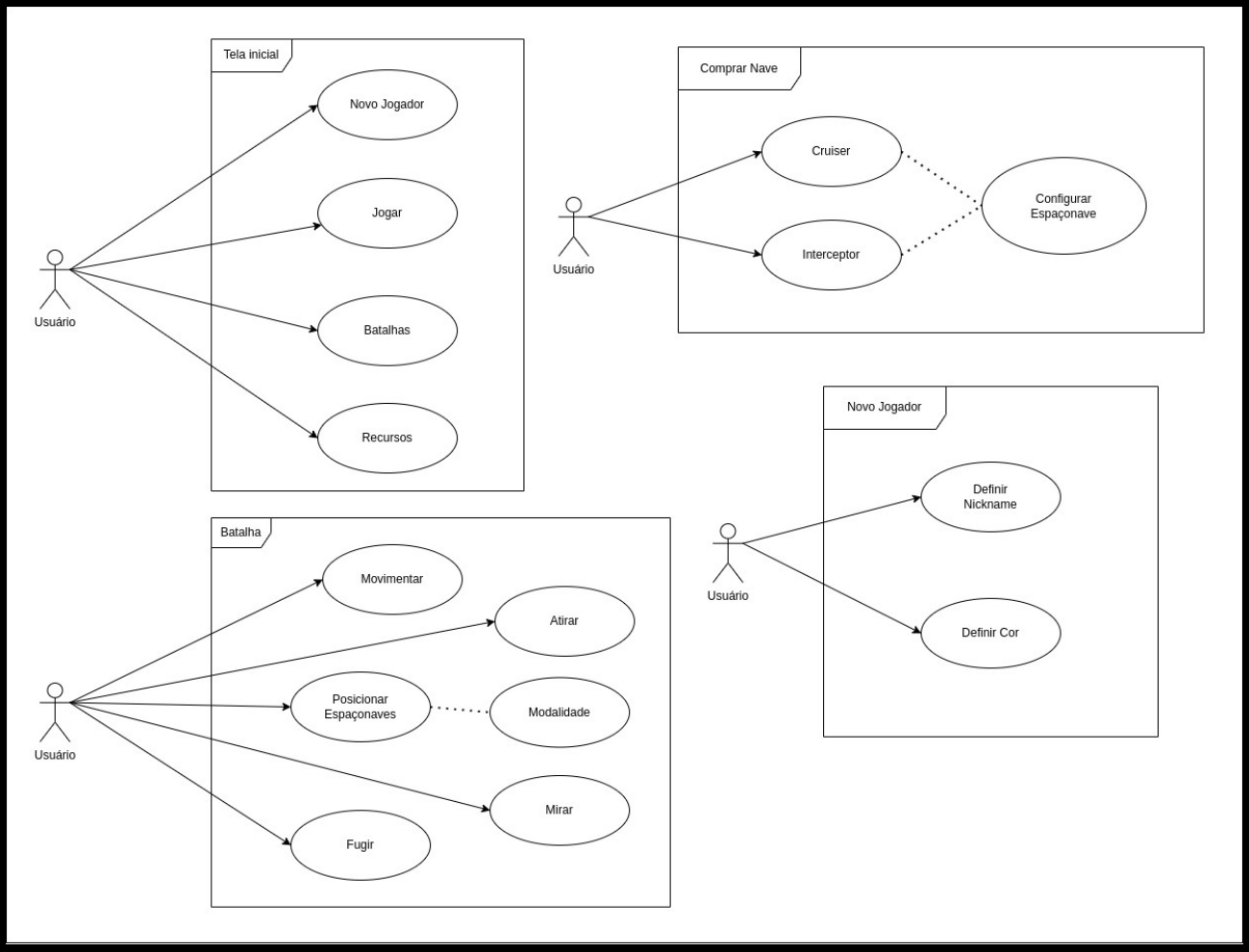

Figura 5 – Diagrama de caso de uso.

Os diagramas acima demonstram as utilidades de cada tela do jogo. Na tela inicial, temos as opções de (1) criar novo jogador, (2) jogar, (3) ver o histórico das batalhas e (4) administrar os recursos de jogadores já criados (comprar espaçonaves). Na tela da batalha, temos as ações dos jogadores, alternadamente, (1) movimentar, (2) Atirar, (3) Posicionar as espaçonaves (início), (4) mirar e (5) fugir. No ato de criar jogadores, os usuários podem definir o nome (*nickname*) e a cor do jogador (observando as cores disponíveis).

Detalhamento de requisitos de <u>número 2</u> e identificação de Classes:

Após uma correção e apresentação, em sala de aula, foram detectadas algumas correções e inserções nas classes identificadas:

1. As classes Terminal e Sreen se transformaram nas classes TerminalMix e ScreenMix, respectivamente. Desta forma, a classe Universe herdará destas duas classes seus membros.
2. A classe Interceptor foi adicionada às classes apresentadas, pois a mesma constava no diagrama de classes mas não havia sido desenhada anteriormente entre as demais classes.
3. A classe Battle recebeu um atributo novo que representa o nome da batalha.
4. As relações de composição entre as classes Interceptor e Cruiser foram estabelecidas, conforme os requisitos iniciais apresentados.
5. A classe ScreenMix recebeu novos membros, para compor seus “menus”: intro(), players(), battles() e resources(). Através destes novos membros a classe diferenciará as opções apresentadas aos usuários. Lembrando que os comandos são realizados pela classe TerminalMix.
6. A relação de composição entre a classe Player e a classe Battle foi estabelecida.
7. A relação de composição entre a calsse Player e a classe Universe foi removida.