Especificação de Requisitos

Yaniv

09/04/2024

|  |  |  |  |
| --- | --- | --- | --- |
| Versão | Autores | Data | Ação |
| 1.0 | Eduardo Gwoszdz, Micael Angelo, Brendo Alves | 13/04/2024 | Estabelecimento dos requisitos |

Conteúdo

1. Introdução

1.1. Objetivo

1.2. Definições e abreviações

1.3. Referência

2. Visão Geral

2.1 Arquitetura do Jogo

2.2 Premissas de desenvolvimento

3. Regras do Jogo

3.1 Definições

3.2 Pontuação

3.3 Pre-Round

3.4 In-Round

3.4.1 Fluxo da jogada

3.4.2 Sequências e conjuntos

3.4.3 Pontuações

3.5 Post-Round

4. Requisitos de software

4.1. Requisitos funcionais

4.2. Requisitos não-funcionais

1 Introdução

1.1 Objetivo

Desenvolvimento de um jogo que comporta 4 jogadores, cada um em sua máquina, se enfrentarem em uma partida de Yaniv, na modalidade JxJ (Jogador contra Jogador).

1.2 Definições e Abreviações

* Abreviações

MC – Monte de Compras

MD – Monte de Descarte

RF – Requisitos funcionais

RNF – Requisitos não funcionais

* Definições:

Descrito no tópico 3.1, Regras do Jogo

1.3 Referencia

Yaniv (jogo) < <https://gamerules.com/rules/jhyap-card-game/> >

2 Visão Geral

2.1 Arquitetura do Jogo

Se baseia em um sistema distribuído onde há uma divisão clara de funções entre o cliente, que solicita serviços ou recursos, e o servidor, que fornece esses serviços ou recursos, permitindo a comunicação e interação entre eles através de uma rede.

2.2 Premissas de desenvolvimento

* Implementação em Python é requerida para o programa.
* Utilização do DOG para facilitar a execução distribuída do programa.
* Além da codificação, é necessário criar uma segunda versão da especificação do projeto, baseada em UML.

3 Regras do Jogo

3.1 Definições

O jogo é realizado com um baralho com 54 cartas (conjuntos de A, 2, 3 ..., Q, K. com 4 naipes (Copas, Ouros, Paus e Espadas) mais 2 coringas.

O jogo possui rotação entre turno para cada jogador. Sendo possíveis ações de compras, descartes e Yaniv. Possuindo 2 Montes de cartas na mesa, 1 MC e 1 MD, sendo o de compras realizado com o restante das cartas após serem distribuídas aos jogadores e o MD será acessível somente em relação ao descarte do último jogador.

3.2 Pontuação

As cartas possuem valores, sendo esses, distribuídos com os seguintes valores:

Ases = 1

Números = Seus próprios valores (2 de Copas = 2 | 3 de Espadas = 3 ...)

Família Real = 10 (J de Espadas = 10, Q de Copas = 10, K de Ouros = 10 ...)

Coringas = 0

* 1. Pre-round

Antes do início da partida, é aguardado os jogadores se conectarem para se iniciar a partia, sendo essa, possível de iniciar somente com a quantidade de 4 jogadores, após todos conectados, é possível iniciar a partida.

* 1. In-round

Ao iniciar o jogo, todos os jogadores recebem cinco cartas aleatórias do baralho. Em seguida, as cartas restantes ficam disponíveis para compra no MC (cartas ficam ocultas até algum jogador realizar a compra).

3.4.1 Fluxo da partida

1. Realizar uma compra. A mesma pode ser feita no MC ou no MD, respeitando sempre a regra de sua retirada ser do topo do monte. Caso o MD esteja vazio, não é possível retirar uma carta do mesmo.
2. Realizar o descarte de uma carta ou uma combinação de cartas.
3. O jogador deve optar por chamar **YANIV** ou não. Caso YANIV seja chamado, a rodada é finalizada e a pontuação é aferida. Caso contrário, o próximo jogador deve realizar sua jogada, e assim, sucessivamente.

3.4.2 Sequências e conjuntos

* **Sequências:**

Essa combinação é possível tendo em sua mão uma sequência de pelo menos 3 cartas com valores sucessivos, contínuos, não rotativo e de mesmo naipe (Espadas, Copas, Paus ou Ouros). Exemplos:

Exemplo 1: 2 de Copas, 3 de Copas e 4 de Copas.

Exemplo 2: 10 de Espadas, J de Espadas, Q de Espadas e K de Espadas.

Exemplo 3: Q de Copas, Coringa, K de Copas.

Exemplo 4 (sequência invalida, pois é rotativa): Q de Ouros, K de Ouros, Ás de Ouros.

* **Conjuntos:**

Essa combinação é possível tendo em sua mão um conjunto de pelo menos 2 cartas de mesmo valor e consequentemente de naipes diferentes. Exemplos.

Exemplo 1: 5 de Paus e 5 de Copas.

Exemplo 2: 7 de Copas, 7 de Espadas e 7 de Ouros.

Exemplo 3: Coringa e K de Ouros.

Exemplo 4 (Conjunto inválido, pois possui naipes diferentes): 4 de Ouros e 4 de Copas.

3.4.3 Pontuação

Após um jogador chamar **YANIV**, o jogo realiza uma lógica para definir a pontuação dos demais jogadores.

* Caso o jogador tenha chamado **YANIV** com uma pontuação maior que 6 pontos, o mesmo receberá uma penalidade de +30 pontos.
* Caso contrário, é necessário verificar as outras mãos na partida.
  + Se a pontuação do jogador que chamou **YANIV** for inferior a todas na mesa, todos os outros jogadores recebem a penalidade de +10 pontos.
  + Caso contrário, ou seja, um ou mais jogadores possuírem uma pontuação inferior ou igual à do jogador, o mesmo recebe uma penalidade de +30 pontos.
  1. Post-round

Ao final da partida, após os valores das pontuações serem adicionados aos placares de cada jogador, uma nova rodada é iniciada, entretanto, com a pontuação dos placares atualizadas. Após isso, as mesmas ações serão repetidas até que um jogador alcance a pontuação igual ou superior a 100 pontos.

A pontuação máxima é o que determina o final da partida. No instante que algum jogador alcançar tal feito, será verificado qual jogador possui a menor pontuação no placar e esse será considerado o vencedor da partida.

Caso ocorra empate, as posições no ranque poderão se repetir, podendo haver múltiplas colocações.

1. Requisitos de Software

4.1 Requisitos funcionais

RF 1. **Iniciar Programa:** Ao ser executado o programa deve aparecer um menu aguardando os quatro jogadores através do Dog Server (utilizando recursos de Dog). Apenas em caso de conexão bem sucedida as demais funcionalidades estarão habilitadas. No caso de conexão mal sucedida, a alternativa é encerrar o jogo.

RF 2. **Iniciar Jogo:** O programa deve apresentar a opção de menu “start game” para o início de uma nova partida. O procedimento de início de partida consiste em enviar uma solicitação de início a Dog Server, que retornará o resultado, que será a notificação, em caso de êxito, ou a razão da impossibilidade de início de partida, caso contrário. A interface do programa deve ser atualizada com as informações recebidas e caso o jogador local seja quem inicia a partida, a interface deve estar habilitada para seu procedimento de lance. Esta funcionalidade só deve estar habilitada se o programa estiver em seu estado inicial, isto é, sem partida em andamento e com a mesa em seu estado inicial.

RF 3. **Restaurar Estado Inicial do jogo**: O programa deve apresentar a opção “restaurar estado inicial” para levar o programa ao seu estado inicial, isto é, sem partida em andamento e com a interface na fase inicial. Esta funcionalidade só deve estar habilitada se o programa estiver com o jogo finalizado.

RF 4. **Selecionar Monte de Compras**: O jogador deve selecionar o MC se desejar comprar sua carta, através do “Buy deck”. Desta forma, a carta do topo é adicionada à mão do jogador. O próximo passo é aguardar o jogador descartar sua(s) carta(s) (RF 6. Selecionar Cartas).

RF 5. **Selecionar Monte de Descarte:** Similar ao “RF 4. Selecionar Monte de Compras. Se o jogador optar por adquirir a carta do monte de descarte, a poderá ser adquirida através do “Discard deck”. Em seguida, é adicionada a mão do jogador. O próximo passo é aguardar o jogador a descartar sua(s) carta(s) (RF 6. Selecionar Cartas)

RF 6. **Selecionar Cartas:** O jogador deve selecionar uma ou mais cartas desejadas para descartar uma combinação de cartas a serem despejadas no MD. As mesmas serão descartadas através do botão “Discard”.

RF 7. **Notificação de início de partida:** O programa notificará assim que houver uma solicitação de início de partida, realizada pelo Dog Server, em função da solicitação de início de partida por parte de outro jogador conectado ao servidor. A interface irá atualizar para receber tais informações, porém, somente o jogador local poderá iniciar a partida.

RF 8. **Receber Jogada:** O programa receberá uma jogada do adversário, enviada por Dog Server, quando for a vez do adversário do jogador local (jogador remoto). A jogada recebida deve ser um lance regular e conter as informações especificadas para o envio de jogadas, como por exemplo, RF 4, RF 5 e RF 6.

RF 9: **Receber notificação de abandono:** O programa notificará assim que um jogador abandonar a partida, por parte de um adversário remoto, enviada por Dog Server. Nesse caso os jogadores serão notificados do abandono, e consequentemente, a partida se encerra, tornando necessário inicia-la novamente com o número correto de jogadores.

RF 10. **Receber Final da rodada**: O programa deve interagir com os demais usuários quando um dos jogadores pressionar o botão Yaniv, o que determina o final da rodada. Após a chamada, o programa verifica os valores de pontuação de cada jogador, determinando se está dentro das regras. Em seguida, os placares são atualizados.

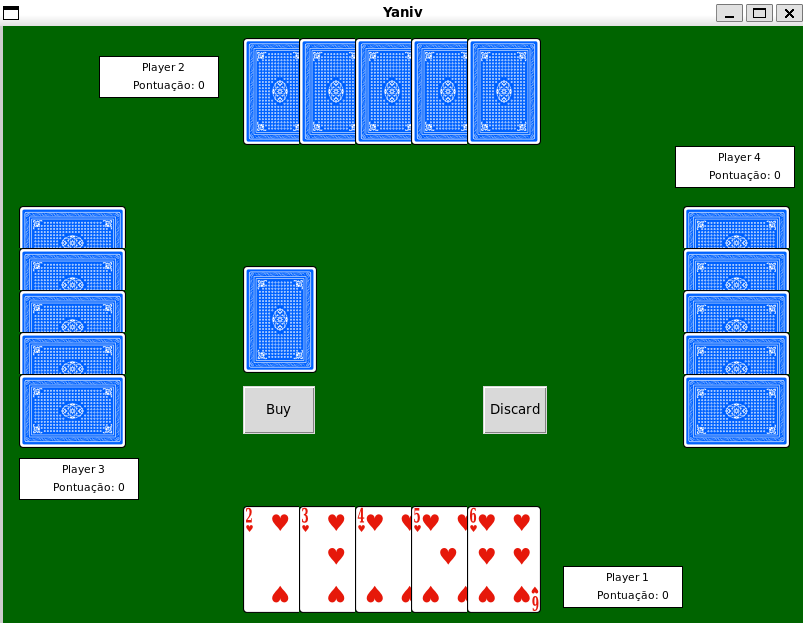
RF 11. **Verificar Final da partida:** Após o recebimento do final da rodada e atualização dos placares, o jogo deve verificar se algum jogador possui 100 pontos ou mais. Assim, a partida é encerrada e a colocação informada.

4.2 Requisitos não funcionais

RNF 1. **Especificação de Projeto**: O código de desenvolvimento será Python na versão 3.10 e juntamente com a especificação de projeto baseada em UML com a ferramenta Visual Paradigm.

RNF 2. **Especificação de Projeto**: O funcionamento distribuído é realizado pelo framework DOG.

RNF 3. **Especificação de Projeto:** A interface gráfica do programa é baseada na biblioteca em TKinter.

RNF 4. **A Interface do programa:** A interface será produzida conforme o esboço da imagem abaixo: