DESENVOLVIMENTO DE JOGOS UTILIZANDO CONCEITOS DE TÉCNICAS ADAPTATIVAS

Leonardo Khenaifes Zaccarelli JUBRAN, Almir Rogério CAMOLESI
leonardokzj@gmail.com, camolesi@femanet.com.br

RESUMO: A adaptatividade em jogos vem sendo utilizada a vários anos. Ela permite que o jogo se adapte de acordo com a habilidade do jogador. Tal conceito vem sendo aprimorado com o tempo, tendo como objetivo uma melhor interação entre jogador e jogo. Neste trabalho foi estudado os tipos de adaptação e foi desenvolvido o jogo 21, também conhecido como *blackjack.* No jogo foi utilizado a adaptatividade de forma que alguma das cartas tenha o seu valor alterado durante o jogo, demonstrando o estudo da técnica.

PALAVRAS-CHAVE: Adaptabilidade; Técnicas Adaptativas; Jogos.

ABSTRACT:
Adaptivity in games has been used for several years in which the game adapts according to the skill of the player and has been improved over time, aiming at a better interaction between both. Thus, the types of adaptability were studied and game 21 was developed, also known as blackjack, where adaptivity was used so that some of the cards have their value changed during the game, demonstrating the study of the technique.

KEYWORDS: Adaptability; Adaptive techniques; Game.

1. Introdução

Ao se falar em jogos adaptáveis, pode parecer uma coisa distante de nossos dias atuais, mas pessoas com uma certa intimidade em jogos percebem que conforme o desenrolar da história, começam a aparecer dificuldades não existentes até o momento e conforme o jogador for melhor ou pior, o jogo se adapta, tornando-se mais fácil ou difícil.

Uma técnica utilizada para auxiliar os projetistas na modelagem de aplicações com comportamento modificável é a tecnologia adaptativa (NETO, 1993). A tecnologia adaptativa envolve um dispositivo não-adaptativo (subjacente) já existente em uma camada adaptativa que permite realizar mudanças no comportamento da aplicação definida (Pistori, 2003).

Hoje temos os *kinetics, wii* da Nintendo, entre outros onde pode usar a interatividade do corpo do jogador, fazendo com que este se sinta realmente no mundo virtual além de poder jogar com outros jogadores ao redor do mundo por meio da internet. Tal equipamento pode ser usado como uma forma de interatividade como, por exemplo, no jogo Clash Royale[1]que os jogadores duelam um contra o outro em tempo real, escolhendo suas tropas como quiser para defender e atacar o adversário, para isso o jogo faz testes dos adversários para escolher o que melhor se encaixa para entrar na batalha.

A adaptatividade pode ser utilizada para diferentes tipos de objetivos, como por exemplo para os jogadores terem maior prazer em jogar. Podem também aumentar ou ajudar a melhora motora do corpo para deficientes e/ou pessoas em algum tratamento, além de alguns jogos terem a possibilidade de jogadores mais sedentários poder praticar algum esporte e ter uma atividade física como, por exemplo, no que acontece com o PokemonGo[2]. Neste jogo o jogador deve sair pelas ruas para colher Pokebolas, capturar *pokemons* e batalhar em arenas, tendo assim uma abrangência grande na área adaptativa.

O trabalho foi organizado da seguinte forma. A primeira sessão apresentou uma ideia geral sobre o projeto. Na sequência, a sessão 2 discorre os conceitos sobre técnicas adaptativas e formas de utilizá-la. Na sessão 3 é apresentado o jogo Blackjack e um diagrama de atividades que ilustra a sequência de sua execução. Depois, na seção 4 é descrita a implementação do jogo modelado. Por fim, na seção 5 são tecidas algumas conclusões finais e trabalhos futuros.

2. Tecnologia Adaptativa

Para um jogo ser bom o jogador precisa encontrar desafio, reconhecer e dominar padrões no jogo para se divertir. Um jogo que não acompanhe o aprendizado do jogador pode se tornar frustrante, pois um jogo que não se adapta é muito previsível que após um

[1] https://play.google.com/store/apps/details?id=com.supercell.clashroyale&hl=pt_BR
[2] https://play.google.com/store/apps/details?id=com.nianticlabs.pokemongo&hl=pt_BR

tempo o jogador conheça todos os passos de seu inimigo. Para evitar isso, seria interessante que os jogos se modifiquem em um período de tempo durante o percurso do jogador.

Estes jogos podem ser adaptativos: aqueles que oferecem um mecanismo de ajuste do jogo ao jogador aumentando ou diminuindo a dificuldade das jogadas automaticamente de forma a facilitar ou dificultar o progresso do jogador conforme seu desempenho.

Um jogo parcialmente adaptativo é aquele que oferece uma forma automática de detecção das habilidades do jogador. Desta forma é preciso prever se o jogador está com habilidade acima da dificuldade atual, mas não realiza a mudança automaticamente; ou um escalamento da dificuldade em apenas uma direção, seja apenas aumentando ou diminuindo a dificuldade até atingir um patamar compatível com o jogador.

Já os jogos não-adaptativos são aqueles que não oferecem um mecanismo adaptativo de dificuldade.

Devemos tomar cuidado ao utilizar a palavra adaptável, pois a mesma pode ser interpretada de duas formas, no âmbito de jogos e também de outras áreas do conhecimento.

Quando é referido a Jogos adaptativos, refere-se àqueles que apresentem uma mudança dinâmica em sua jogabilidade, ou seja, a mudança aplicada na inteligência dos NPCs[3] ou nos desafios gerados, refletindo a interação do jogador com o jogo e oferecendo uma experiência única a cada jogador.

Porém quando é referido a Jogos adaptáveis. Estes geralmente apresentam uma evolução fixa e progressiva no desenrolar do jogo em relação aos desafios enfrentados pelo jogador, visto como um usuário ideal, com aprendizado. Permite mudar elementos estáticos dentro do jogo (cor, fonte, desenhos gráficos...).

Alguns exemplos de jogos adaptativos são **Mario Kart 64**[4] no qual ajusta a velocidade dos NPCs e probabilidade dos itens para ajudar ou atrapalhar o jogador de acordo com sua posição na corrida e **Left 4 Dead**[5].

Os jogadores selecionam a dificuldade da fase antes de começar a jogar e o sistema AIDirector realiza a distribuição de inimigos e de itens.

Para uma boa adaptação do jogo tem que ter um equilíbrio, no qual se dá de duas formas:

---

[3]non-player character (NPC), são personagens no jogo que não são controlados por nenhum jogador, no qual suas funções e ações são controladas pela máquina.

[4] http://www.emuparadise.me/Nintendo_64_ROMs/Mario_Kart_64_(USA)/39949

[5] http://www.l4d.com/blog/

- Equilíbrio estático: Relativo às regras do jogo, ou seja, como esse jogo deve ser jogado, no qual essas regras não devem ser alteradas.

- Equilíbrio dinâmico: Relativo à interação do jogador com o jogo. Neste caso o jogo pode fazer alterações durante a sua execução de acordo com a capacidade do jogador.

Sendo assim, a jogabilidade deve fornecer: Desafios compatíveis com a habilidade do jogador; uma experiência de jogo justa, o jogador não deve ser condenado desde o princípio do jogo por conta de seus "erros", ausência de estagnação (rumo do jogador), Ausência de decisão em elementos não pertencentes a regra do jogo, além de possuir níveis de dificuldade, podendo o jogador escolher a dificuldade e assim ajustar à habilidade do jogador durante o jogo.

3. Jogo Adaptativo Blackjack

Nesta seção será descrito o objetivo do jogo desenvolvido, o seu funcionamento e as funções que ele apresenta durante a jogada bem como a arquitetura do mesmo.

3.1 Descrição do Jogo

Desenvolver uma modalidade de jogo que permita ao jogador, desde o mais inapto até o mais experiente, um maior entusiasmo ao brincarem com um jogo que possui um comportamento adaptativo.

Neste contexto é proposto o jogo Blackjack. Tal jogo contem 52 cartas de um baralho que são distribuídas para os jogadores pelo *dealer* da seguinte forma. Primeiramente é distribuída uma carta para cada jogador, na sequência que irão jogar, após a primeira distribuição dar-se a segunda carta para cada um nesta mesma sequência, inclusive para ele próprio.

Por vez, cada jogador pode pedir mais cartas ou parar, de acordo com os valores das cartas de sua mão, sabendo que seus valores correspondem aos respectivos números marcados nelas exceto o *ÁS* que vale 1 ou 11 e *Q (Dama), J (Valete)* e *K (Reis)* que valem 10 pontos.

Ganha o jogo quem chagar mais próximo de 21 pontos ou fizer a maior soma próxima a 21 pontos.

3.2 Projeto do Jogo

Primeiramente foi desenvolvido um Diagrama de Atividades, com base nos conceitos da linguagem de modelagem UML (MEILIR, 2001), no programa Astah[6]. O referido diagrama tem por objetivo apresentar a sequência de atividades desempenhadas por um determinado jogador ao executar o referido programa.

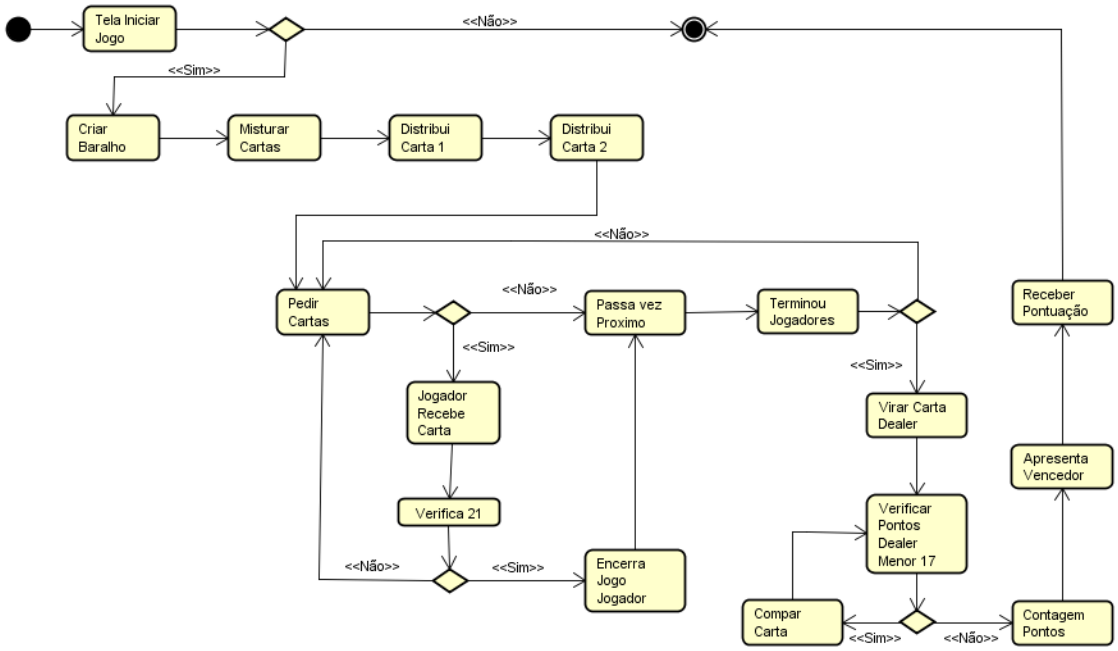

Figura 1. Diagrama atividades Jogo Blackjack.

Exemplificando a Figura 1, primeiramente pode se iniciar ou sair do jogo, se este for iniciado é criado o baralho, este é embaralhado e após distribuída uma carta para cada jogador e depois a segunda na sequência de jogada. Após a distribuição das cartas inicia-se a vez do primeiro jogador. Este por sua vez, pode pedir mais cartas ou não, se não pedir é passada a vez para o próximo se sim, este jogador recebe mais uma carta, é feita a contagem dos pontos e se não estourar os 21 pontos em sua mão, poderá continuar no jogo, ou pode simplesmente passar a vez. Após o primeiro jogador realizar a sua jogada passa-se a vez para o próximo jogador que tem as mesmas opções e por fim, na vez do *Dealer*, faz-se uma contagem de pontos, onde se tiver menos de 17 pontos ele recebera mais cartas e se tiver mais de 17 pontos será apresentado automaticamente o jogador vencedor.

[6] http://astah.net/download

Após o jogo ter sido projetado, foi definida uma linguagem de programação para sua codificação. Para tal tarefa foi escolhida e utilizada a linguagem C# (SHARP, John) e para codificar o jogo foi utilizada a plataforma Visual Studio 2015[7] como ferramenta de desenvolvimento.

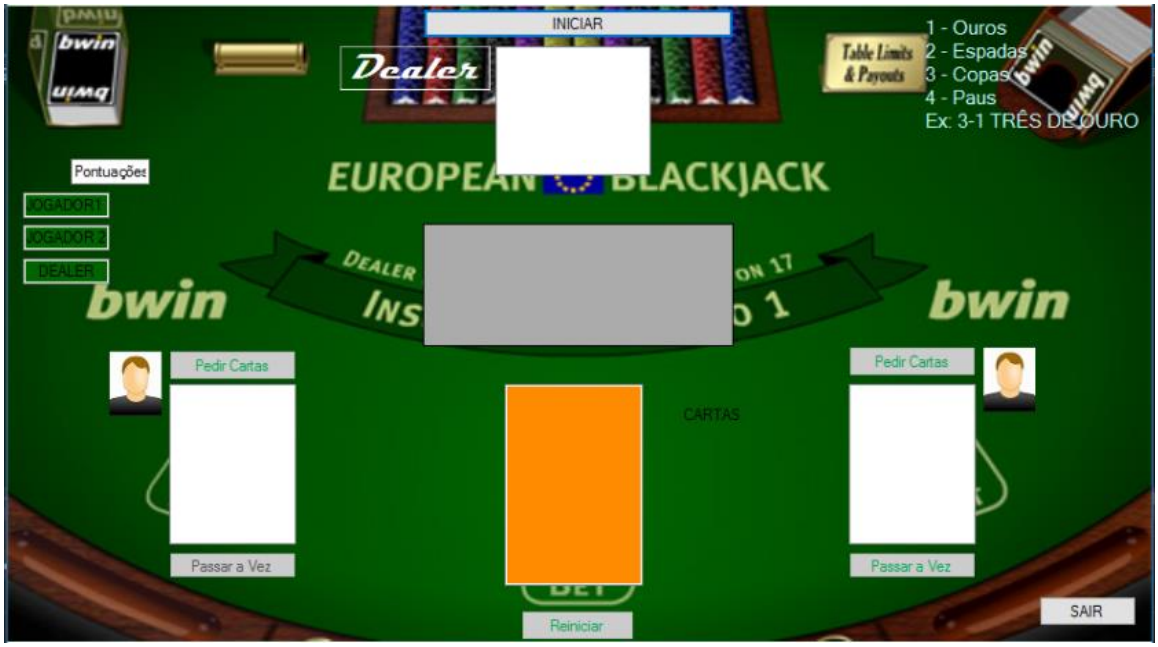

Figura 2. Interface principal do jogo Blackjack antes de inicia-lo.

Os conceitos de adaptatividade no jogo foram inseridos por meio de cartas escolhidas aleatóriamente que são marcadas com um asterisco (*) indicando que tais cartas são as adaptativas, são gerados também números aleatórios em uma lista que poderão ser utilizados para substituir os valores das cartas que podem ser adaptadas. O seu funcionamento dar-se-á ao selecionar a carta adaptativa. Após a seleção da mesma aparece uma janela sobre a janela do jogo (tipo *Pop-Up*) perguntando se o jogador deseja utilizar a carta adaptativa ou não, se a resposta for sim, o valor desta carta passa a ser o valor de um número aleatório que é gerado entre 0 e 15. Desta forma o valor da carta pode passar do valor máximo que é dez, ajudando ou mesmo tirando o jogador da jogada.

[7] https://www.microsoftstore.com/store/msbr/pt_BR/list/Visual-Studio/categoryID.66829200?s_kwcid=AL!4249!3!97331280569!e!!g!!visual%20studio%202015&WT.mc_id=pointitsem+Google+Adwords+Visual+Studio+-+PT&ef_id=V6PNiQAABAOIGx9M:20161208120610:s

Figura 3. Jogo após clicar no botão iniciar.

Na Figura 3, pode-se verificar que o jogador dois possui duas cartas oito (8) que também possuem valor 8, tendo em conta que o primeiro valor é o número da carta, seguido pelo naipe, e por último o valor da carta.

Figura 4. Mostrando e manuseando carta adaptativa.

Quando a carta é selecionada, apresenta-se a janela *Pop-Up* questionando se o jogador deseja adaptar a carta. No caso dele aceitar, a carta recebe um valor aleatório, (mostrado na figura 4.1) gerado em uma sequência indefinida.

Figura 4.1. Carta adaptada.

3.2 Execução do Jogo

A jogada começa pelo primeiro jogador que se situa do lado inferior esquerdo da tela, este pode escolher passar a vez ou se exceder os 21 pontos, a jogada é direcionada ao segundo jogador. O segundo jogador fica situado no canto inferior direito e após o mesmo é a vez do *Dealer* (situado na parte central superior) que se tiver com uma soma de cartas menor que 17 ponto recebera cartas e quando sua soma passar a ser maior que 17 pontos, permanece com o soma de seus pontos e se dá o ponto ao jogador ganhador.

Desta forma, inicia-se o jogo ao selecionar o botão “Iniciar”, conforme ilustrado no Diagrama de Atividades, Figura 1, e gera-se inicialmente o baralho. Depois do baralho misturado, sorteia-se algumas cartas do baralho e as transforma em adaptativas. Por fim, as cartas são distribuídas a todos os jogadores e ao *Dealer*. Nas figuras apresentadas anteriormente as cartas tem seus valores apresentados de forma a ilustrar o que está ocorrendo no jogo, porém na versão final do jogo, estes valores não serão apresentados para tornar o jogo realístico ao jogo real.

Observa-se na Figura 5 que na vez do jogador 1 (um), este excedeu seus 21 pontos ao pegar mais uma carta ficando com 25 pontos e automaticamente foi desclassificado.

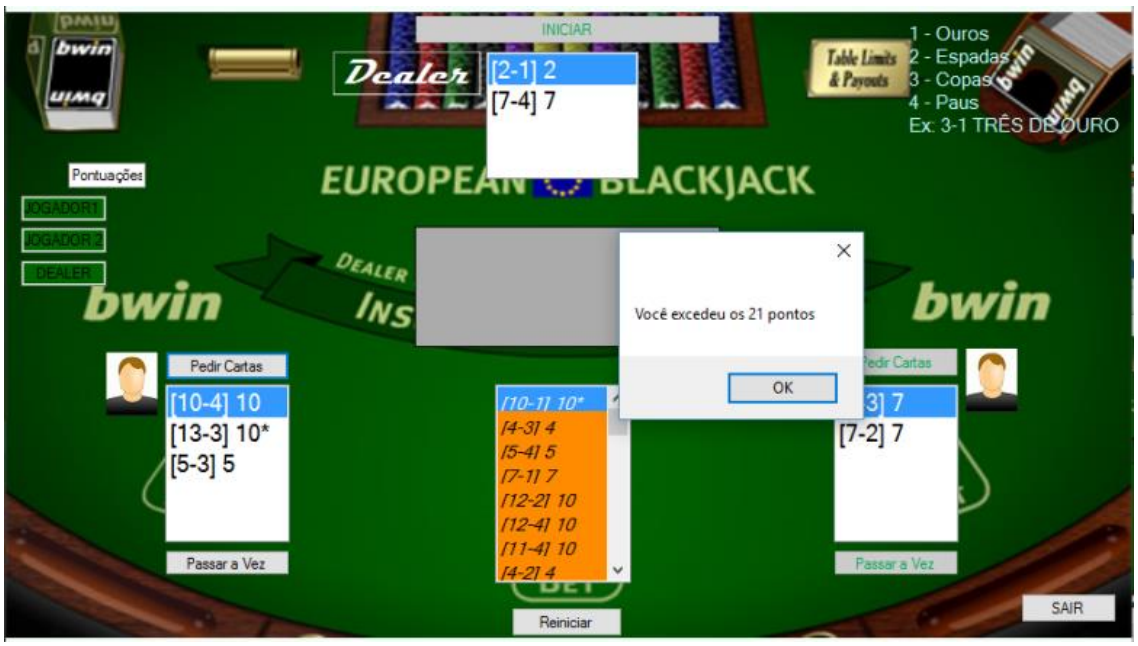

Figura 5. Pontos excedidos.

Ao chegar na vez do segundo jogador este passou a vez e ficou com 14 pontos, automaticamente o *Dealer* recebeu mais uma carta e ficou com 19 pontos chegando mais perto dos 21 pontos necessários e na contagem venceu o jogo.

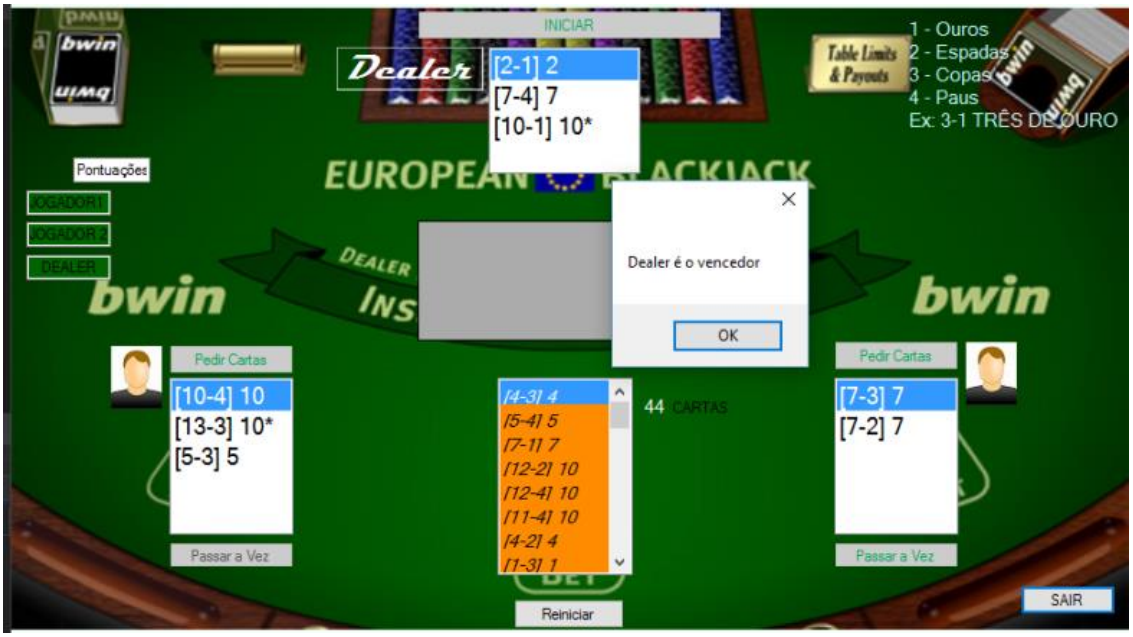

Figura 6. Apresentação do vencedor.

Assim, o jogo gerou um ponto para o *Dealer* enquanto os outros jogadores não receberam nenhum ponto. Depois da rodada ser finalizada, o jogador pode optar por reiniciar uma nova rodada e continuar com a contagem dos pontos, ou pode optar por sair. Se escolher está última ação os pontos retornam ao valor zero.

Figura 7. Pontuação do jogador vencedor.

4. Desenvolvimento do Jogo Blackjack

Para que o jogo funcione corretamente tem-se que desenvolver uma programação, a qual é responsável por realizar toda a execução das ações do jogo e suas funcionalidades. No Código 1, apresenta-se o código responsável por gerar o baralho que contem cartas de *Às* à 10, *Dama*, *Valete* e *Reis*, totalizando 13 cartas de valores diferentes, além disso temos os naipes que são 4 (*ouros*, *copas*, *espadas* e *paus*), totalizando 52 cartas.

Também vemos que são marcadas 10 cartas que farão a função adaptável deste jogo, sendo geradas aleatoriamente e contemplando todas as cartas do baralho.

Código 1. Método para criação e embaralhamento das cartas.

```
private void Criar_Baralho() {
    for (int j = 1; j <= 4; j++)
       for (int i = 1; i <= 13; i++) {
            Carta carta = new Carta();
            carta.naipe = j;
            carta.face = i;
            if (i > 10)
                     carta.valor = 10;
            else carta.valor = i;
            Baralho.Add(carta);
            topo++;
            carta.adap = false;
        }
  lstBaralho.DisplayMember = "carta"; //atributo virtual da classe Carta
  int seed = DateTime.Now.Millisecond * DateTime.Now.Year *
DateTime.Now.Second;
  Random rand = new Random(seed);
```

Para estas cartas adaptáveis serem inseridas no baralho de forma aleatória foi criada uma função “Criar_Aleatorio” com 10 cartas adaptáveis que variam do valor 0 à 15, facilitando ou dificultando o jogo para cada jogador que recebe-las. No Código 1, é apresentada tal função. Observa-se que inicialmente é executado o comando *“for”* onde o valor 4 representa os 4 naipes que as cartas possuem e no segundo *“for”* as 13 cartas do baralho do *ÀS* ao *Reis*, onde são geradas 13 cartas de *espadas*, outras 13 de paus e, assim, consecutivamente, até serem geradas as 52 cartas do baralho. Após isso é feito um teste para valores das cartas *Q (Dama)*, *J (Valete)* e *K (Reis)* na qual cada uma recebe o valor de 10 pontos. Também atribuído o valor das outras cartas que é exatamente o número marcado nelas e após tudo isto ser realizado, o baralho é embaralhado com uma função chamada *randômica*.

Código 2. Método para gerar números para a carta adaptativa e zerar o jogo ao inicia-lo.

```
private void Criar_Aleatorio()
        {
            Aleatorio.Clear();
            int qntAdap = 10;
            int valorMaximo = 15;
            int seed = DateTime.Now.Millisecond *
DateTime.Now.Year * DateTime.Now.Second;
            Random rand = new Random(seed);
            for (int i = 0; i < qntAdap; i++)
            {
                Aleatorio.Add(rand.Next() % valorMaximo);
            }
        }

        private void btnIniciar_Click(object sender, EventArgs e)
        {
            Criar_Baralho();
            Criar_Aleatorio();
            btnIniciar.Enabled = false;
            pedirCartas1.Enabled = true;
            pasVez1.Enabled = true;
            reiniciar.Enabled = true;
```

A função “btnIniciar_Click” que é chamada ao selecionar o botão “Iniciar” é responsável por distribuir as cartas para os jogadores. No início do jogo por meio da execução do comando *“if”* distribuindo 2 cartas para os jogadores em 2 rodadas de distribuição e atualiza todas as cartas em suas listas.

Código 3. Método para chamar todas as funções necessárias ao iniciar o jogo, zerar e atualizar as listas e distribuir cartas.

```
private void btnIniciar_Click(object sender, EventArgs e)
        {
            Criar_Baralho();
            Criar_Aleatorio();
            btnIniciar.Enabled = false;
            pedirCartas1.Enabled = true;
            pasVez1.Enabled = true;
            reiniciar.Enabled = true;

            for (int i = 0; i < 2; i++)
            {
                Inserir_Lista_Jogador(lista_jog1);
                Inserir_Lista_Jogador(lista_jog2);
                Inserir_Lista_Jogador(lista_D);
            }
            listJ1.DataSource = null;
            listJ1.DisplayMember = "carta";
            listJ1.DataSource = lista_jog1;
            listJ2.DataSource = null;
            listJ2.DisplayMember = "carta";
            listJ2.DataSource = lista_jog2;
            listD.DataSource = null;
            listD.DisplayMember = "carta";
            listD.DataSource = lista_D;
```

Aqui temos o método “pedirCartas_Click”. Tal método ocorre quando o jogador seleciona o botão “Pedir Cartas”, e permite ao jogador receber uma nova carta da lista do baralho. Além de verificar se este jogador excedeu seus 21 pontos ao resgatar a carta.

Código 4. Função pedir cartas.

```
        private void pedirCartas1_Click(object sender, EventArgs e)
        {

            Inserir_Lista_Jogador(lista_jog1);
            listJ1.DisplayMember = "carta";
            listJ1.DataSource = lista_jog1;

            label1.Text = Baralho.Count.ToString();
            atualizarListBox(listJ1, lista_jog1);
            atualizarListBox(lstBaralho, Baralho);

            int soma = lista_jog1.Sum(c => c.valor);
            if (soma > 21)
            {
                MessageBox.Show("Você excedeu os 21 pontos");
                pasVez1_Click(null, null);
            }
```

O Código 5, mostra que através de Cálculo Lambda soma-se novamente a numeração das cartas ao final do jogo, por meio de seus valores, obtidos nas listas de cartas que estes jogadores têm em suas mãos, para que seja apresentado ao jogador, e apresenta-se o seu devido ponto.

Código 5. Função para somar pontuação de cada jogador.

```
            soma1 =+ lista_jog1.Sum(c => c.valor);
            soma2 =+ lista_jog2.Sum(c => c.valor);
            soma =+ lista_D.Sum(c => c.valor);
```

Por fim são realizados vários testes para ver se o jogador atende as regras de especificação para ganhar o ponto e se este for o vencedor, atribui-se o ponto ao jogador.

Código 6. Testes para apresentação do vencedor.

```
if (soma1 > soma2 && soma1 > soma && soma1 <= 21 )
            {
                MessageBox.Show("Jogador 1 é o vencedor.");
                s1 += 1;
                label4.Text = s1.ToString();

            }
            else if (soma2 > soma1 && soma2 > soma && soma2 <= 21)
            {
                MessageBox.Show("Jogador 2 é o vencedor.");
                s2 += 1;
                label5.Text = s2.ToString();

            }
            else if (soma1 > soma2 && soma1 == soma && soma1 <= 21)
            {
                MessageBox.Show("Jogador 1 é o vencedor.");
                s1 += 1;
                label4.Text = s1.ToString();

            }
            else if (soma2 > soma1 && soma2 == soma && soma2 <= 21)
            {
                MessageBox.Show("Jogador 2 é o vencedor.");
                s2 += 1;
                label5.Text = s2.ToString();

            }
            else if (soma1 == soma2 && soma1 > soma && soma2 <= 21)
            {
                MessageBox.Show("Empate Jogador 1 & 2.");
                s1 += 1;
                s2 += 1;
                label4.Text = s1.ToString();
                label5.Text = s2.ToString();

            }
            else if (soma<21)
            {
                MessageBox.Show("Dealer é o vencedor");
                s3 += 1;
                label6.Text = s3.ToString();

            }
        }
```

3. Conclusão

A realização deste projeto permitiu um aprendizado de formas e técnicas de jogos adaptativos e um maior aprendizado em programação e desenvolvimento de aplicações.

A adaptatividade permitiu que o jogo fique um pouco mais interativo com o jogador saindo de seu padrão e assim trouxe uma nova forma de jogar este conhecido jogo.

A linguagem C# foi uma ferramenta de fácil utilização. Tal linguagem serviu para a construção deste jogo de forma rápida e pratica utilizando-se dos conceitos de orientação a objetos, adquiridos em sala de aula. Além disso permitiu utilizar os recursos disponíveis para desenvolver a interface (design do programa) de forma a dar uma "cara" ao jogo.

A fase de projeto serviu para conhecer e ampliar técnicas de desenvolvimento de jogos através desta plataforma e melhorar a pratica para construir códigos.

Por fim, o jogo permitiu o melhor desenvolvimento da técnica adaptativa gerando novas formas de se jogar.

Como trabalhos futuros poderão ser desenvolvidos mais funções para o jogo e uma melhor interação deste com o jogador, implementando outras formas destas técnicas, onde podem ser utilizadas para outros propósitos, inclusive para auxiliar pessoas com dificuldades especiais.

Agradecimentos

Os autores agradecem primeiramente ao Programa de Iniciação Cientifica (PIC), da Fundação Educacional do Município de Assis (FEMA), pelo apoio e incentivo financeiro, para desenvolvimento deste trabalho.

Eu Leonardo Jubran, agradeço a meu pai Fernando Jubran, pelo incentivo e pelo apoio aos meus estudos e, os autores, também o agradecem pelo auxilio prestado na correção ortográfica deste trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

CAMOLESI, A.R.; NETO, J.J. An adaptive model for specification of distributed systems. IX Congreso Argentino de Ciencias de la Computación, La Plata, Argentina, 6-10 de Outubro, 2003.

CAMOLESI, A.R.; NETO, J.J. Modelagem Adaptativa de Aplicções Com-plexas. XXX Conferencia Latinoamericana de Informática - CLEI'04. Arequipa - Peru, Setiembre 27 - Octubre 1, 2004a.

CAMOLESI, A.R.; NETO, J.J. Modelagem $AMBER_{Adp}$ de um Ambiente para Gerenciamento de Ensino a Distância. Anais do XIII Simpósio Brasileiro de Informática na Educação - SBIE 2002, pp. 401-409, São Leopoldo, RS, Novembro 12-14, 2002.

CAMOLESI, A.R.; NETO, J.J. Representação Intermediária para Dispositivos Adaptativos Dirigidos por Regras. 3rd International Information and Telecommunication Technologies Symposium, UFSCar, São Carlos, Brasil, 2004b.

CAMOLESI, A.R. Uma metodologia para o Design de Serviços de TV-Interativa. Dissertação de Mestrado, PPG-CC, UFSCar, 2000.
http://www.maxwell.vrac.puc-rio.br/21362/21362_4.PDF

MEILIR, PAGE-JONES Fundamentos do desenho orientado a objeto com UML, Makron Books, 2001.

NETO, J.J. Adaptive Rule-Driven Devices - General Formulation and Case Study. Lecture Notes in Computer Science. Watson, B.W. and Wood, D. (Eds.): Implementation and Application of Automata 6th International Conference, CIAA 2001, Springer-Verlag, Vol.2494, pp. 234-250, Pretoria, South Africa, July 23-25, 2001.

NETO, J.J.; ALMEIDA Jr.J.R.; NOVAES, J.M. Synchronized Statecharts for Reactive Systems. Proceedings of the IASTED International Conference on Applied Modelling and Simulation, pp.246-251, Honolulu, Hawai, 1998.

NETO, J.J. Contribuições à metodologia de construção de compiladores. Tese de Livre Docência, USP, São Paulo, 1993.

NETO, J.J. Cross-Assemblers para Microprocessadores - Geração Automática Através do SPD. I CONAI - Congresso Nacional de Automação Industrial, pp. 501-509, São Paulo, 1983.

NETO J.J. Introdução a Compilação. EDITORA: LTC, Rio de Janeiro, 1987

NETO, J.J.; IWAI, M.K. Adaptive Automata for Syntax Learning. CLEI 98 - XXIV Conferencia Latinoamericana de Informatica, MEMORIAS. pp. 135-149, Quito, Equador, 1998.

NETO, J.J. e MAGALHÃES, M.E.S. Um Gerador Automático de Reconhecedores Sintáticos para o SPD. VIII SEMISH - Seminário de Software e Hardware, pp. 213-228, Florianópolis, 1981.

NETO, J.J. Uma Solução Adaptativa para Reconhecedores Sintáticos. Anais EPUSP - Engenharia de Eletricidade - série B, vol. 1, pp. 645-657, São Paulo, 1988.

SHARP, John ; Microsoft Visual C# 2008 : Passo a Passo, BOOKMAN, 2008