



FACULDADE DE TECNOLOGIA DE AMERICANA
Curso Superior de Tecnologia em Jogos Digitais

André Vitor Paganini

Nicolas Silva Volpi

iMagit

Americana, SP

2017



FACULDADE DE TECNOLOGIA DE AMERICANA
Curso Superior de Tecnologia em Jogos Digitais

André Vitor Paganini
Nicolas Silva Volpi

iMagit

**Relatório técnico desenvolvido em cumprimento à exigência curricular do
Curso Superior de Tecnologia em Jogos Digitais sob a orientação do Prof. Dr.
Kleber de Oliveira Andrade**

Americana, SP.
2017

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

P147i PAGANINI, André Vitor

iMagit. / André Vitor Paganini, Nicolas Silva Volpi. – Americana: 2017.

52f.

Monografia (Curso de Tecnologia em Jogos Digitais) - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Dr. Kléber de Oliveira Andrade

1. Jogos eletrônicos 2. Jogos de mesa e tabuleiro I. VOLPI, Nicolas Silva II. ANDRADE, Kléber de Oliveira II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana

CDU: 681.6


André Vitor Paganini
Nicolas Silva Volpi


iMagit


Trabalho de graduação apresentado
como exigência parcial para obtenção do
título de Tecnólogo em Jogos Digitais,
pelo CEETEPS/Faculdade de Tecnologia
– Fatec/ Americana.
Área de concentração: Desenvolvimento
de Jogos Digitais

Americana, 30 de junho de 2017.

Banca Examinadora:


Kléber de Oliveira Andrade (Presidente)
Doutor
Fatec Americana


Daniele Junqueira Frosoni (Membro)
Especialista
Fatec Americana


Eduardo Antonio Vicentini (Membro)
Mestre
Fatec Americana

RESUMO

Com a popularização de dispositivos moveis, cada dia mais pessoas estão tendo acesso a jogos e formas de entretenimento individuais e afastando o conceito de jogar com as pessoas que estão por perto. Esse projeto visa aproveitar esses dispositivos, aliado com temáticas clássicas de jogos de tabuleiro, para trazer de volta a interação local entre jogadores. Utilizamos a ferramenta Unity para desenvolver um aplicativo para dispositivos Android que simula as regras e os componentes de um jogo de tabuleiro, porém mantendo a parte interativa fora do mundo digital. Descobrimos que a receptividade das pessoas é muito boa para esse sistema de jogos e o nível de diversão alcançado foi compatível com o dos jogos analógicos.

Palavras Chave: jogos de tabuleiro; multijogadores locais; multijogadores em rede; jogos para celular; conversão de mídia

ABSTRACT

People more and more each day have access to individual games and entertainment forgetting about playing with people around them. This has happened due to the popularization of the mobile devices such as cellphones and tablets. The present project has as a main goal to take advantage of this kind of devices also using classical board games theme to bring back the local interaction among people. We use Unity engine to develop an application to Android devices which simulates the rules and components of a board game, however keeping the interactive piece outside the digital world. We have found the people receptivity is huge and the level of entertainment compatible with analogical games.

Keywords: *board games, local multiplayer, net multiplayer, games for cellphone, media conversion*

Lista de Figuras

Figura 1 - Dados da questão: possui um dispositivo Android?.....	
9	
Figura 2 - Dados da questão: joga jogos de tabuleiro ou cartas?.....	
9	
Figura 3 - Dados da questão: jogaria esses jogos em dispositivos portáteis?.....	
10	
Figura 4 - Dados da questão: Empecilhos para adquirir estes jogos.....	
10	
Figura 5 - Dados da questão: preferência ao jogar.....	
11	
Figura 6 - Jogo Dixit.....	
12	
Figura 7 - Primeira carta de exemplo.....	
14	
Figura 8 - Segunda carta de exemplo.....	
15	
Figura 9 - Paleta de cores.....	
16	
Figura 10 - Entrada do jogo.....	
19	
Figura 11 - Sala de espera.....	
20	
Figura 12 - Início da Partida.....	
21	
Figura 13 - Primeira parte da rodada no servidor.....	22
Figura 14 - Segunda parte da rodada no servidor.....	
23	
Figura 15 - Primeira parte da rodada no cliente.....	
24	

Figura 16 - Segunda parte da rodada no cliente.....	25
Figura 17 - Menu principal.....	26
Figura 18 - Menu configurações.....	26
Figura 19 - Sala de espera.....	27
Figura 20 - Tutorial.....	28
Figura 21 - Explicação tutorial.....	28
Figura 22 - Aguardando o jogo ser iniciado.....	29
Figura 23 - Indicação do narrador.....	29
Figura 24 - Mão do jogador.....	30
Figura 25 - Seleção de carta.....	30
Figura 26 - Tela de aguardo.....	31
Figura 27 - Cartas na mesa.....	31
Figura 28 - Seleção da carta descrita.....	32
Figura 29 - Resultado.....	32
Figura 30 - Tabuleiro.....	33

SUMÁRIO

1	INTRODUÇÃO	7
2	METODOLOGIA.....	8
	2.1 PLANEJAMENTO.....	8
	2.2 SOBRE O JOGO	
IMAGIT.....		12
	2.3 ELABORAÇÃO DAS CARTAS.....	13
	2.4 PALETA DE CORES DO	
MENU.....		15
	2.5 DIVISÃO DE GRUPOS.....	16
	2.6 CRONOGRAMA.....	17
	2.7 FERRAMENTAS UTILIZADAS.....	18
3	IMPLEMENTAÇÃO.....	19
	3.1 INICIO DO JOGO.....	19
	3.2 TELA DO SERVIDOR.....	20
	3.3 TELA DO CLIENTE.....	23
4	RESULTADOS.....	26
5	CONSIDERAÇÕES	FINAIS
	34
	REFERÊNCIAS	36
	APÊNDICE A - CÓDIGO-FONTE.....	37

1 INTRODUÇÃO

O mercado de jogos vai além dos jogos digitais. Os jogos analógicos são atividades lúdicas sem necessariamente envolver o meio digital, contendo regras e objetivos a serem cumpridos. Esses são compostos de alguns componentes para a jogabilidade ser realizada, como um tabuleiro e peças no caso do xadrez, em que o objetivo de cada jogador é movimentar suas peças pelo tabuleiro e conquistar o “rei” do adversário, ou cartas, como no Uno, onde o objetivo é ser o primeiro jogador a ficar sem cartas na mão, utilizando todos os meios possíveis para impedir que os outros jogadores façam o mesmo.

Esses e diversos outros jogos analógicos já receberam uma adaptação digital. As regras são as mesmas, assim como os componentes básicos para se jogar. A diferença agora é que todos os elementos do jogo estarão em meio digital. Não é necessário transportar os componentes, estar no mesmo local, e o investimento para adquirir tal produto é baixo se comparado à versão física, ou mesmo, em alguns casos, como nos jogos citados, são possíveis encontrar versões gratuitas, o que vem se tornando uma tendência. A pesquisa Game Brasil (2017, Pág. 22), apontou que 71,3% dos entrevistados que utilizam dispositivos *smartphone* e *tablets*, adquirem somente jogos gratuitos. Entretanto, a possibilidade de as pessoas jogarem tais jogos entre si a longa distância veio com um custo: o contato e a interação presencial. Como observado pelos dados coletados por uma pesquisa realizada pelo grupo, 74,3% dos entrevistados prefere jogar com outros indivíduos presencialmente ao invés de sozinho ou em rede.

Sendo assim, o objetivo deste trabalho é desenvolver um jogo que permita manter a experiência do contato pessoal e, da melhor maneira possível, resolver todos os problemas anteriormente apresentados. Para isso, foi utilizado a Unity 5 que é uma game engine, juntamente com a linguagem C# e foi desenvolvido o jogo iMagit, onde cada participante deve tentar adivinhar qual foi a carta descrita pelo narrador, devendo ser jogado presencialmente utilizando dispositivos *Android*, com o intuito de facilitar o acesso ao jogo.

2 METODOLOGIA

Será demonstrado a coleta de dados e pontos importantes que ajudaram no desenvolvimento do projeto, exemplificando as regras do jogo, inspirações e ferramentas utilizadas para o desenvolvimento.

2.1 Planejamento

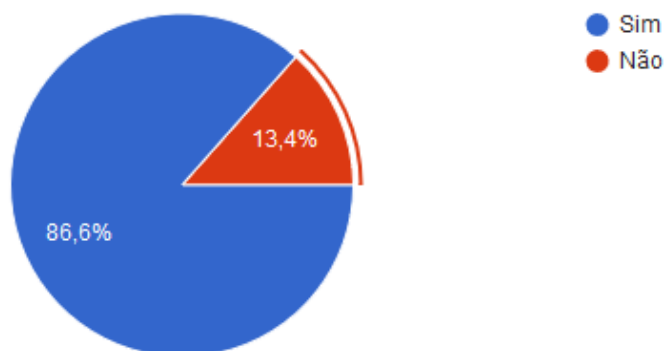
Após debates e análise de algumas ideias, foi proposto o trabalho com jogos de tabuleiros ou cartas em meio digital, mas de modo que fosse criado um meio diferenciado para a experimentação mantendo alguns fatores que são mais satisfatórios. Para cumprir a proposta, planejamos o jogo iMagit que poderia ser jogado entre um grupo de pessoas reunidas em um mesmo local, utilizando uma rede *wifi*. Cada jogador utiliza seu próprio dispositivo, chamado daqui para a frente de cliente, que serão utilizados para exibir suas cartas. Para relacionar e interligar os aparelhos, há a necessidade do que será designado como servidor, um dispositivo posicionado em local de fácil visualização para todos os jogadores, tendo como finalidade exibir todas as informações gerais necessárias e interligar os aparelhos cliente, agindo assim como um tabuleiro ou mesa.

Para descobrir melhor como o conceito seria acatado pelo público, um questionário online foi realizado utilizando o serviço “Google formulários”. Com cerca de cento e cinquenta respostas coletadas, observamos que 86,6% dos entrevistados (Figura 1) possui um dispositivo *Android*, sistema operacional no qual planejamos desenvolver o projeto, confirmando que há grande disponibilidade deste dispositivo em específico.

Figura 1 – Dados da questão: possui dispositivo Android?

Possui um dispositivo Android ?

149 respostas



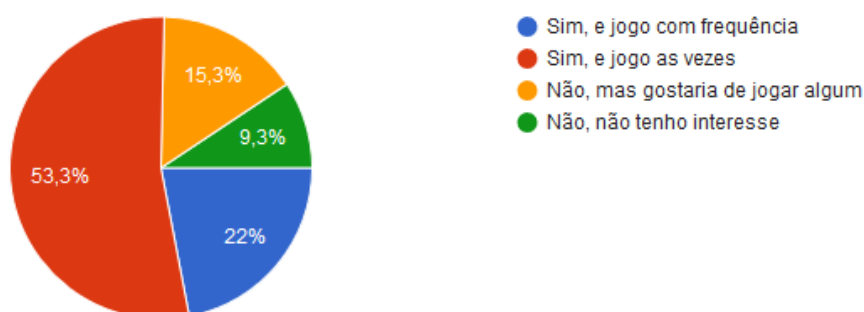
Fonte: Próprios Autores

Com relação a interesse em jogos de tabuleiro ou carta, constatamos que 75,3% dos consultados joga as vezes ou com frequência jogos de tabuleiro ou de cartas (Figura 2), apontando que existe um público não somente interessado no tipo de jogo, mas um grupo ativo de pessoas que pratica estas atividades.

Figura 2 – Dados da questão: joga jogos de tabuleiro ou cartas?

Joga algum jogo de tabuleiro ou de cartas?

150 respostas



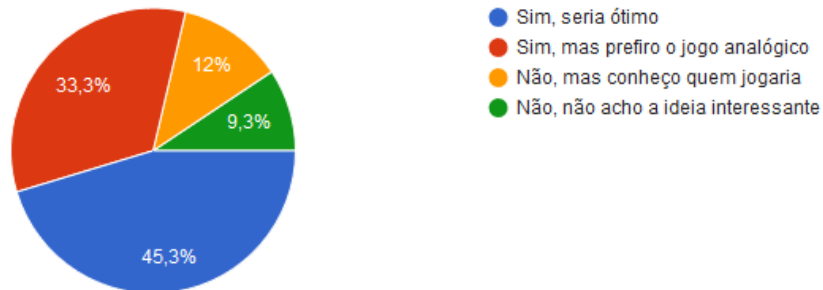
Fonte: Próprios Autores

A pesquisa também demonstrou que 78,6% dos questionados jogaria esses tipos de jogos em dispositivo portátil caso houvesse oportunidade e somente 12,3% não jogaria, mas conhece alguém que o faria (Figura 3), justificando, desta forma, a criação do modelo de jogo proposto.

Figura 3 – Dados da questão: Jogaria esses jogos em dispositivos portáteis?

Jogaria esses tipos de jogos em dispositivos portáteis presencialmente caso houvesse oportunidade?

150 respostas



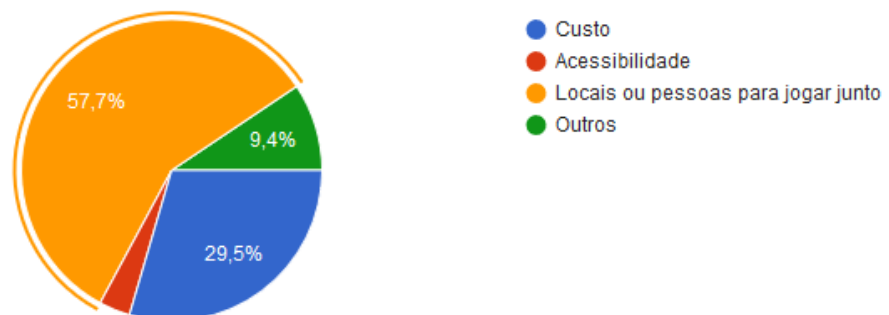
Fonte: Próprios Autores

A pesquisa ainda revelou que 57,7% (Figura 4) dos entrevistados acreditam que o maior empecilho, que os desmotiva a adquirir esse tipo de produtos, são locais ou pessoas para jogar junto, seguido pelo custo, com 29,5% dos votos. Com isso, observamos que o a falta de pessoas ou locais adequados para jogar é um problema maior do que o esperado. O projeto não encontra problemas de custo, não há materiais físicos que precisam de financiamento para produção, mas não possui ferramentas que auxiliem encontrar pessoas para jogar em conjunto, algo que talvez possa ser melhorado em futuros aprimoramentos.

Figura 4 – Dados da questão: Empecilhos para adquirir estes jogos

Quais os principais empecilhos para adquirir estes tipos de jogos?

149 respostas



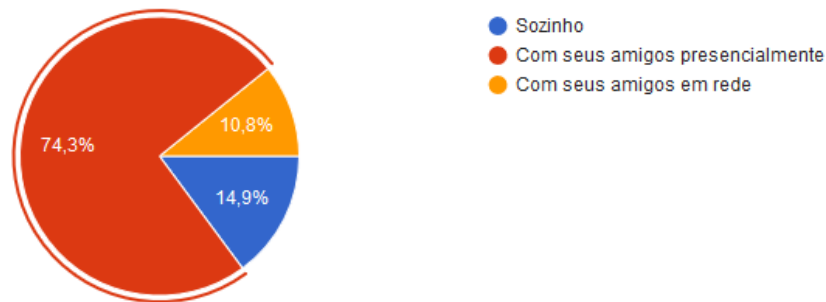
Fonte: Próprios Autores

Foi observado (Figura 5) que 74,3% dos consultados preferem jogar com os amigos presencialmente. Reforçando, novamente, a ideia da importância e desejo da interação social presencial.

Figura 5 – Dados da questão: preferência ao jogar

Você prefere jogar?

148 respostas



Fonte: Próprios Autores

Após o conceito ser concluído, analisamos alguns jogos existentes do mercado e optamos por produzir um jogo baseado em Dixit (Figura 6), já que este cumpria com as características desejadas pelo grupo: um jogo cujas regras fossem fáceis de entender e memorizar, com rápida jogabilidade e a possibilidade de ser jogado múltiplas vezes utilizando a imaginação dos jogadores. Conforme Novak (2010, p.196):

“As informações perfeitas resultam em desafios lógicos, nos quais os jogadores assimilam as informações e as utilizam para decidir qual é a melhor linha de ação[...] Com informações *imperfeitas*, os jogadores recebem apenas uma fração das informações necessárias para tomar a melhor decisão. Ao contrário do que acontece nos desafios lógicos de games com informações perfeitas, os desafios nesse games também exigem dedução (A capacidade de fazer uma suposição sobre a natureza das informações que faltam).”

Figura 6 - Jogo Dixit



Fonte: [https://en.wikipedia.org/wiki/Dixit_\(card_game\)](https://en.wikipedia.org/wiki/Dixit_(card_game))

Em Dixit os jogadores recebem cartas contendo imagens abstratas e tem que adivinhar qual é a carta descrita pelo narrador. Após a descrição o narrador recebe uma carta de cada jogador, as embaralha e as posiciona viradas para cima. Os jogadores devem então adivinhar a carta descrita pelo narrador. O objetivo do narrador é descrever a carta de modo que

O preço do jogo, entretanto, pode ser considerado elevado. Segundo o site BODOGAMI, o jogo base que contém somente algumas cartas e componentes básicos para jogar tem o custo de R\$ 159,90, cada expansão que contém um conjunto novo de cartas custa R\$ 99,90. Preços que poderiam ser facilmente reduzidos em um jogo digital, devido a desnecessidade da produção de materiais físicos.

2.2 Sobre o jogo iMagit

O jogo iMagit tem um sistema muito simples de aprender e pode ser jogado por qualquer pessoa. A cada rodada um dos jogadores é escolhido para ser o

narrador. Esse jogador então escolhe uma das cartas em sua mão e a coloca virada para baixo na mesa e ele performa algum som, palavra, frase, música ou gesto fazendo referência à carta que foi escolhida.

Após isso, o restante dos jogadores escolhe uma de suas cartas que representem a referência escolhida pelo narrador, que serão posicionadas lado a lado com a do narrador, igualmente viradas para baixo. Embaralham-se as cartas e então todas as cartas então são desviradas. O intuito dessa fase é confundir os outros jogadores, de modo que eles não consigam facilmente identificar qual foi a carta escolhida pelo narrador.

Cada jogador, com exceção do narrador, deve votar tentando adivinhar qual carta foi jogada pelo narrador para ganhar pontos. A pontuação é distribuída da seguinte maneira:

Narrador: Caso algum jogador acerte sua carta, porém não todos os jogadores, lhe será concedido três pontos.

Jogador: O jogador que votar na carta descrita pelo narrador ganhará três pontos.

Adicionais: Cada jogador, seja o narrador ou não, recebe um ponto para cada outro jogador que votou em sua carta.

Para determinar o término do jogo, um dos participantes deve atingir trinta pontos, e esse será considerado o vencedor.

De modo que todos os participantes possam manter noção fácil de suas respectivas pontuações e a de seus adversários, primeiramente, ao se conectar ao jogo, cada jogador será indicado com uma cor. No aparelho servidor será exibido um tabuleiro contendo o baralho de cartas e as peças com as cores de cada jogador. O número de casas avançadas no tabuleiro por cada peça representa a pontuação adquirida por cada jogador.

2.3 Elaboração das Cartas

Para elaboração das cartas, foi criado uma lista com palavras selecionadas ao acaso. Segundo Novak (2010; p.198) “Conhecimento extrínseco é aquele adquirido fora do mundo do game e aplicado a ele... Esses fatos amplamente conhecidos podem ser usados para solucionar diferentes enigmas ou solucionar desafios no game.” Utilizando tal conceito, foi realizada uma filtragem, de modo que

palavras de difícil entendimento ou interpretação, cujo significado ou a ideia fora considerado limitado ou não interessante com a proposta, fossem removidas da lista.

Todas as cartas, então, foram elaboradas contendo três das palavras restantes, totalizando cinquenta e seis. Para tornar melhor a consistência e frequência que cada uma delas era selecionada, foi utilizado o seguinte método:

1. Espelho
2. Zíper
3. Dragão
4. Navio
5. Vela
6. Projeto
7. Jaula

A primeira carta é composta pelos itens (1, 2, 3), seja por visual direto, deixando claramente o item visível, ou indireto, metafórico, somente por formas ou ideias que transmitisse o significado. A segunda carta (Figura 7) foi formada pelos itens (2, 3, 4), seguindo a mesma ideia no desenvolvimento. A terceira pelos itens (3, 4, 5), assim por diante.

Figura 7 – Primeira carta de exemplo



Fonte: Próprios Autores

Figura 8 – Segunda carta de exemplo

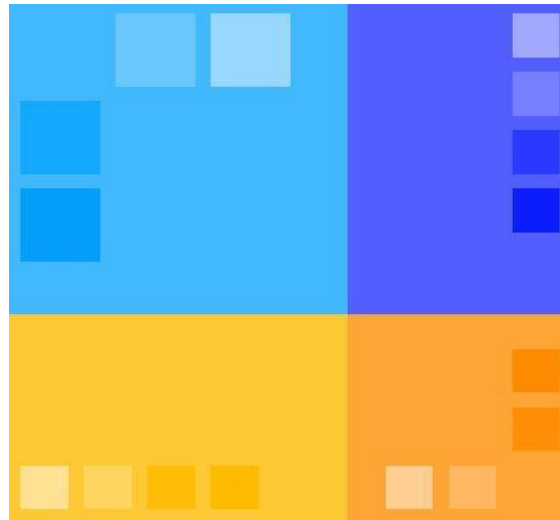


Fonte: Próprios Autores

2.4 Paleta de cores do menu

Paleta de cores é um sistema utilizado para buscar a identidade visual de um design, pode ser usada em logotipos, softwares, propagandas, sites e qualquer outro conteúdo realizado por um designer. O objetivo dela é garantir harmonia entre as cores de modo que o usuário se sinta confortável com a aplicação e receba a mensagem que o designer quis passar.

Nosso projeto partiu da cor inicial do logo do jogo, foi escolhido um azul céu pois, segundo a teoria da psicologia das cores, essa seria a cor vinculada ao infinito, aos sonhos e o imaginativo, a partir dessa cor utilizamos a ferramenta Paletton para buscar quais cores complementares fariam parte da paleta. (Figura 9)

Figura 9 – Paleta de cores

Fonte: Paletton

Definimos alguns parâmetros no site e obtemos um tom complementar de azul para criar um contraste sem perder a mesma finalidade e efeito do primeiro tom, e a cor análoga escolhida foi o laranja pois ela traz ao jogador a sensação de excitação e diversão. Essa combinação de cores escolhidas passa de uma forma direta toda a ideia do que o jogador pode esperar ao jogar esse jogo.

2.5 Divisão de Grupos

Para organização e otimização do tempo, o grupo foi dividido em duas partes:

A primeira, formando a equipe de artes, composta por Nicolas Volpi e Lucas Sousa, ficou a cargo da produção das artes das cartas e planejamento do cenário, da interface do usuário.

A segunda, a equipe de desenvolvimento, composta por André Paganini e Gabriel Andrade, ficou encarregada da programação e desenvolvimento do jogo, realizando a montagem do cenário com os componentes feitos pela outra equipe – a de artes - interligação dos aparelhos cliente e servidor, da manutenção preventiva e elaboração dos sons.

2.6 Cronograma

Como pode ser visto a seguir (Tabela 1), é mostrado como foi planejado e administrado o tempo ao longo dos meses e semanas. Cada fase do cronograma representa uma série de trabalhos e tarefas realizados pelo grupo.

Planejamento: Discussão sobre a possibilidade de realizar um jogo de tabuleiro completamente físico que foi barrada por limitações de preço e publicação. Baseado nas limitações que encontramos, surgiu todo o conceito final que foi desenvolvido.

Cartas e interface: Todo o processo de criação das cartas pela equipe de artes, bem como o tabuleiro, logo, menus e texturas.

Cliente: Utilização da Unity engine para montar o visual e programar as ações relacionadas ao dispositivo que é utilizado pelos jogadores.

Sons e animações: Esforço conjunto das duas equipes para produzir e implementar todos os efeitos, músicas e animações das cartas no jogo.

Servidor: Implementação da cena do servidor que contém o tabuleiro, transição de câmera, distribuição de cartas, comunicação de redes, sistema de votação e sistema de marcação de pontos.

Documentação: Escrita do relatório de desenvolvimento e preparação para entrega e lançamento do projeto.

Testes: Levamos o jogo para colegas e amigos para que jogassem e compartilhassem suas experiências conosco. Ajustes foram feitos baseado no resultado obtido.

Tabela 1 – Cronograma de trabalho

Tarefas	Março		Abril		Maio		Junho
Semana	1 e 2	3 e 4	1 e 2	3 e 4	1 e 2	3 e 4	1 e 2
Planejamento	x						
Cartas e interface		x	x	x	x		
Cliente			x	x			
Sons e animações				x	x		
Servidor				x	x	x	
Documentação				x	x	x	x
Testes					x	x	x
Entrega							x

Fonte: Próprios autores

2.7 Ferramentas utilizados

Para o desenvolvimento do projeto, utilizamos a licença gratuita da Engine Unity, que tem como finalidade a produção de jogos, contendo diversos recursos que facilitam na junção de diversas áreas, como artes, programação e música, combinando diversos elementos e componentes prontos para formar o jogo.

Para o desenvolvimento dos códigos, foi se usado o software *Visual Studio Community*, programa para escrever, compilar e auxiliar na remoção de erros de códigos. Com ele foi possível escrever todos o código fonte do jogo, fazendo a interligação do cliente e servidor, funcionalidade do menu e

Para a produção das cartas, logo, tabuleiro e demais partes gráficas, *GIMP (GNU Image Manipulation Program)*, um programa de edição e criação de imagens, foi a ferramenta escolhida. A escolha dessa foi feita devido sua licença gratuita.

Maya Student Version, versão gratuita para estudantes, foi utilizado para a modelagem 3D, usado para fabricação do cenário, que não foi finalizada e não está incluso.

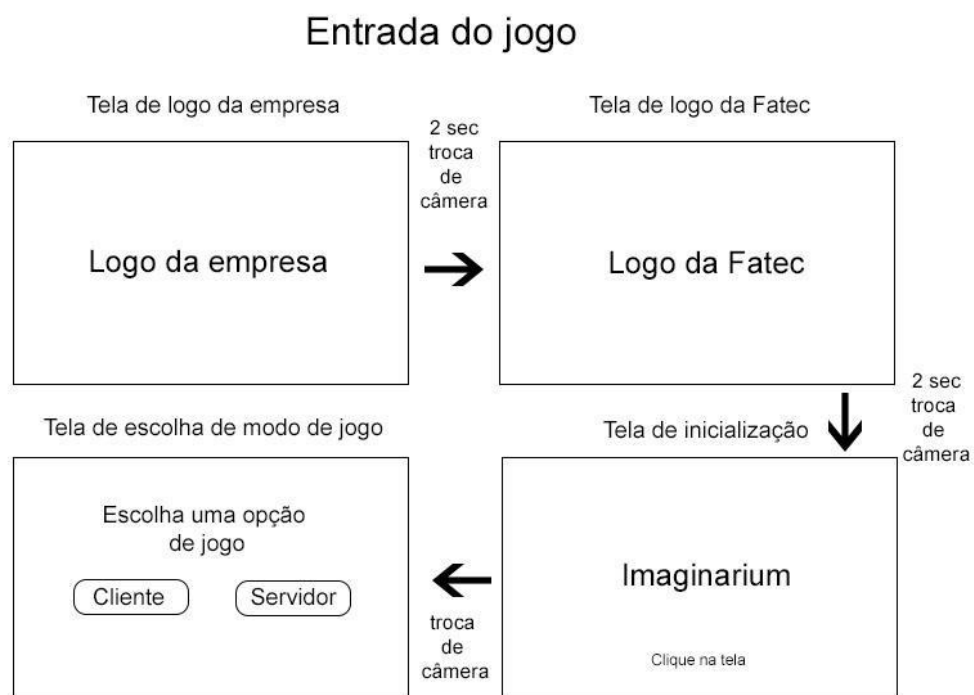
3 IMPLEMENTAÇÃO

Será demonstrado a implementação do jogo, demonstrando esquemas de como os menus, clientes e servidores devem funcionar.

3.1 Início do jogo

Como demonstrado (Figura 10), ao inicializar o jogo seria exibido o logo do grupo, em seguida, com um atraso de dois segundos, o logo da Faculdade de Americana, e novamente, com o mesmo tempo de atraso, a tela de inicialização exibindo o título do jogo, onde permanecerá até que o jogador execute o comando para inicializar o jogo. Após tal comando ser inserido, deverá ser escolhido se o aparelho operará como cliente, sendo este um dos jogadores a se conectar, ou servidor, atuando como tabuleiro e exibindo todas as informações do jogo. Devido a mudanças no projeto, o optou-se por não colocar o logo da empresa nem a tela de inicialização.

Figura 10 - Entrada do jogo



Fonte: Próprios autores

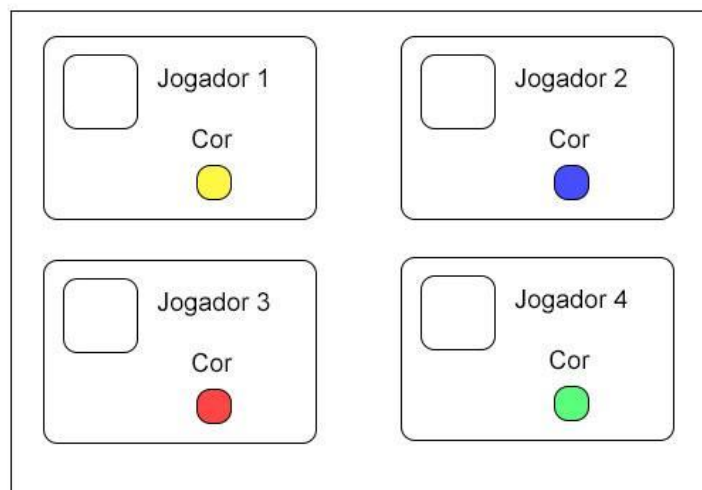
3.2 Tela do Servidor

Como demonstrado (Figura 11), é possível observar a sala de espera como foi pensada no conceito inicial, na qual são mostrados todos os jogadores que se conectarem ao jogo, bem como seus nomes e respectivas cores. Na versão final do jogo optamos por um formato de lista, mais conservador, pois esse modo ajuda o entendimento das pessoas que não estão acostumadas com jogos digitais e facilita a visualização para telas pequenas.

Figura 11 - Sala de espera

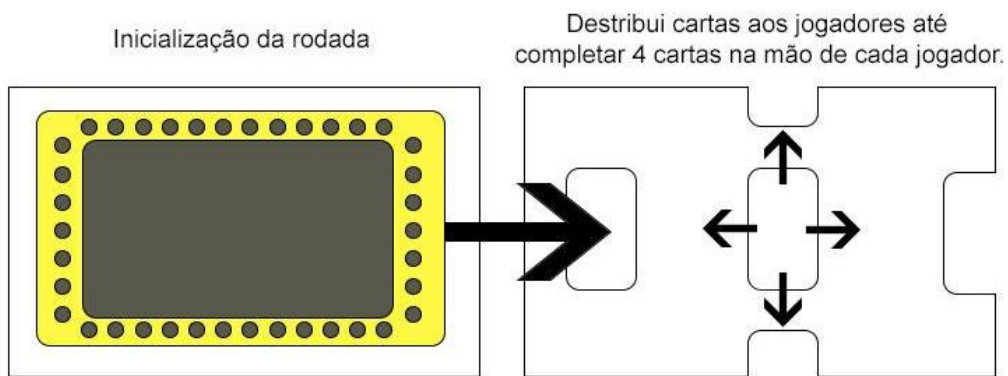
Fases do jogo Servidor

Sala de espera - Aguardar todos os jogadores se conectarem



Fonte: Próprios autores

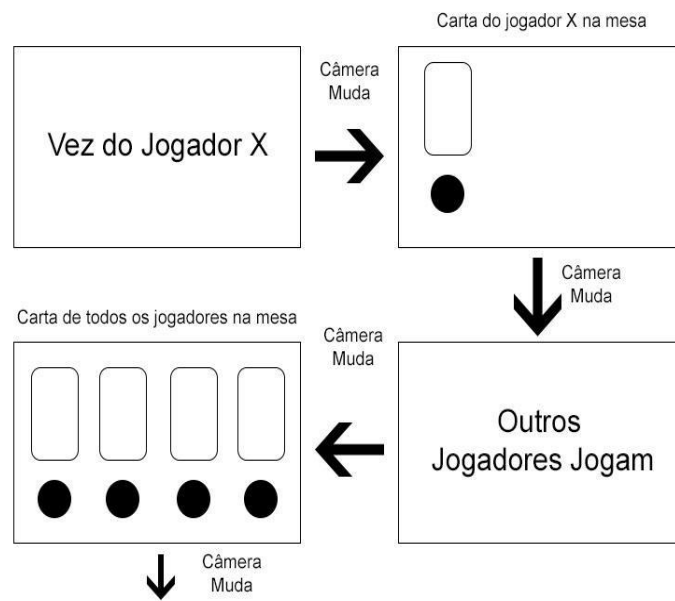
A partida será iniciada, conforme mostra a demonstrado (Figura 12), com o tabuleiro exibido ao fundo e as cartas embaralhadas e distribuídas entre os jogadores, até que cada um contenha um total de quatro.

Figura 12 – Início da partida

Fonte: Próprios Autores

Encerrando a distribuição (Figura 13), será selecionado um jogador aleatoriamente para descrever sua carta, tornando-se o narrador da rodada. Ele então realizará seu turno, descrevendo sua carta que será exibida no servidor virada para baixo.

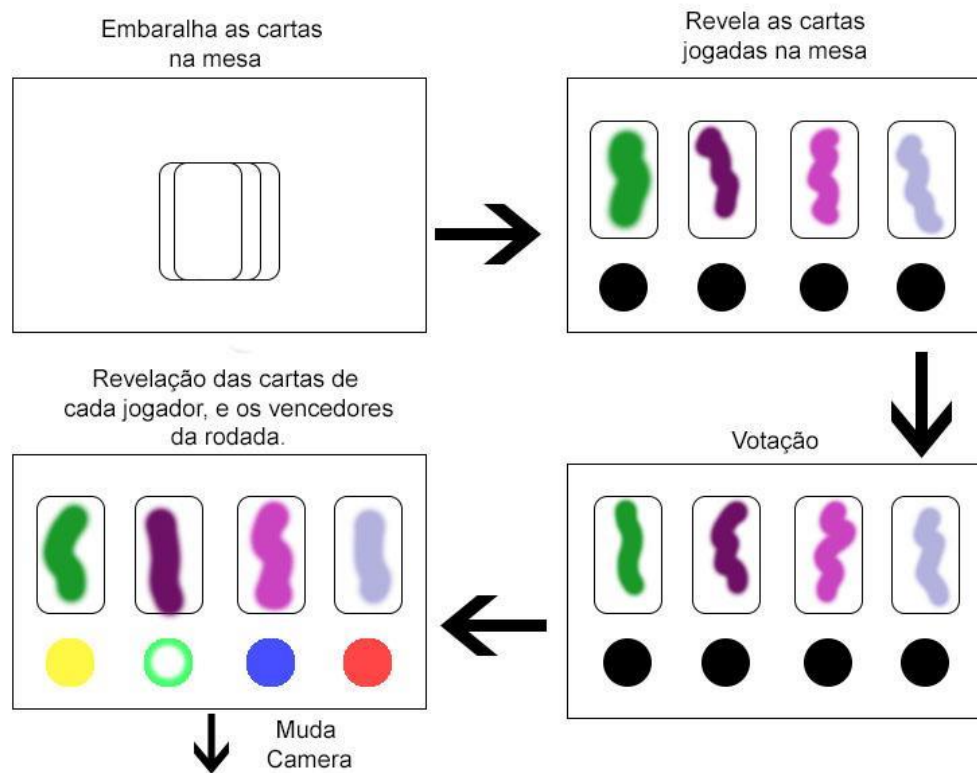
Figura 13 – Primeira parte da rodada no servidor



Fonte: Próprios Autores

Os outros jogadores então selecionaram uma de suas respectivas cartas, que serão postas lado a lado com a do narrador e embaralhadas, como exemplificado (Figura 14). Depois de todos os jogadores realizarem suas escolhas, as cartas serão reveladas, mostrando qual carta pertencia a qual jogador.

Figura 14 – Segunda parte da rodada no servidor



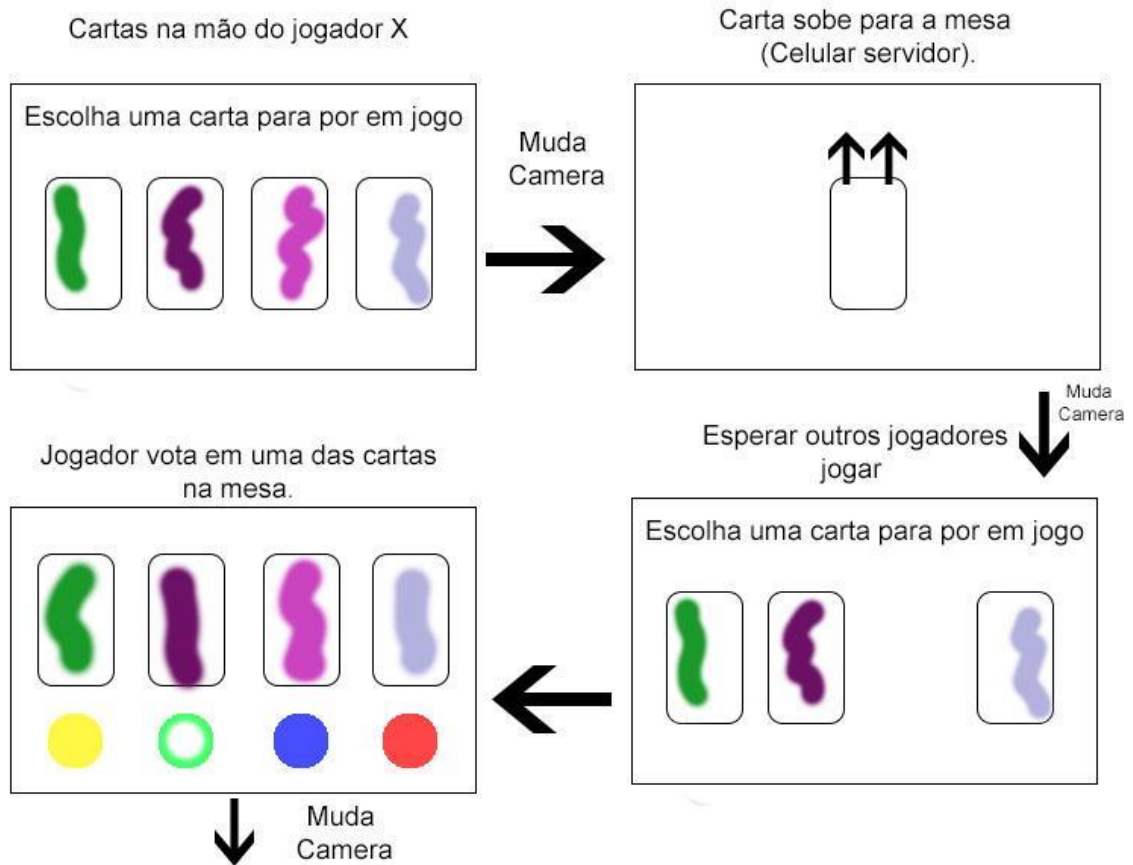
Fonte: Próprios Autores

3.3 Tela do Cliente

Após digitar o código do tabuleiro, o jogador será conectado ao servidor na sala de espera (Figura 11).

Quando a partida se iniciar (Figura 15), será exibido ao jogador todas as suas cartas. Se ele for narrador, deverá escolher a carta que quer enviar ao servidor e descrevê-la. Do contrário, deve aguardar a narrador descrever a carta e então enviar a sua. Quando todos os jogadores enviarem suas respectivas cartas, a fase de votação começa. Será exibido na tela de cada cliente as cartas enviadas ao servidor.

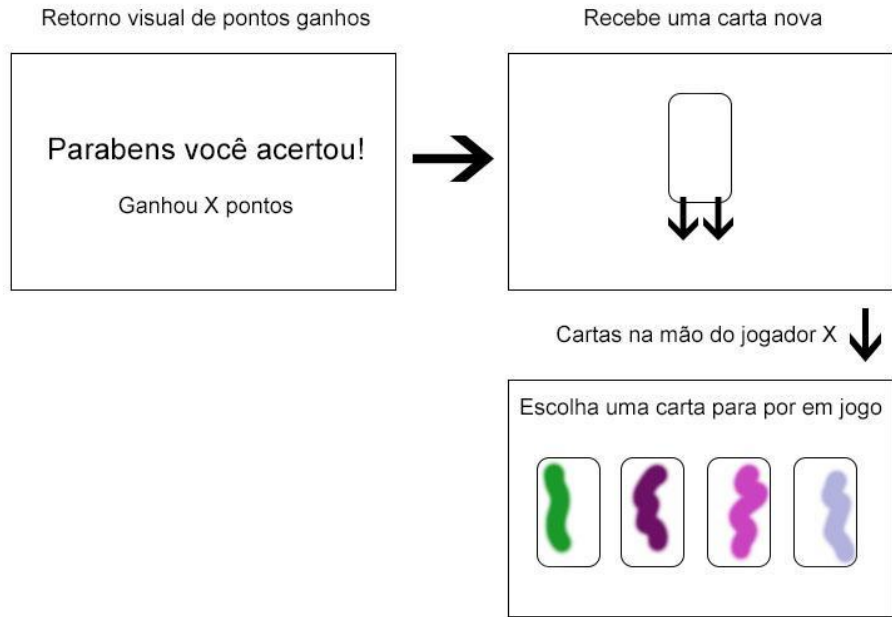
Figura 15 – Primeira parte da rodada no cliente



Fonte: Próprio Autor

Ao término da rodada (Figura 16), o jogador receberia uma resposta, dizendo se ele acertou e quantos pontos ganhou, além de ganhar uma nova carta. Isso foi alterado na versão final para fazer a atualização de pontos na tela do tabuleiro, mantendo assim a interação entre os jogadores por mais tempo enquanto acompanham a pontuação.

Figura 16 – Segunda parte da rodada no cliente



Fonte: Próprios Autores

4 RESULTADOS

Foi possível implementar um menu intuitivo (Figura 17), com opções para acessar tutorial, menu de configurações e selecionar se o dispositivo será tabuleiro ou jogador.

Figura 17 – Menu Principal



Fonte: Próprios autores

Nas configurações (Figura 18) foi colocado o essencial apenas, uma opção para ligar e desligar o som, outra para os efeitos e um local para o jogador trocar o nome. Para maior conveniência, todas essas opções são salvas ao jogador sair, de modo que estarão do modo como ele deixou ao utilizar da última vez.

Figura 18 – Menu de configurações



Fonte: Próprio autor

A sala de espera (Figura 19) foi implementada da maneira recomendada, é possível ver todos os jogadores, existe a opção de confirmar se o jogador está pronto para o jogo começar, um botão para sair, trocar a cor de cada jogador, não podendo repetir, o endereço para se conectar ao tabuleiro e o nome facilmente visível e fácil de ser trocado, bastando apenas tocar na barra com o nome.

Figura 19 – Sala de espera



Fonte: Próprio autor

O que não foi feito, devido a tempo, foi a detecção automática do servidor, removendo a necessidade do jogador de digitar o código do tabuleiro.

Para ensinar as regras básicas do jogo, foi criado um tutorial (Figura 20) que pode ser acessado pela opção "como jogar" no menu principal (Figura 17). Nele foram utilizadas fotos de aparelhos reais para exemplificar melhor como pode ser ajustado o servidor e o cliente.

Nessa mesma tela, é possível retornar para a instrução anterior, seguir para a próxima ou sair e retornar para o menu a qualquer momento. Decidimos colocar pouco texto em cada imagem de modo a não cansar nem desmotivar o jogador a ler as regras.

Existe um total de trinta e uma imagens exemplificando o funcionamento do jogo, e uma imagem inicial para mostrar demonstrar os comandos do tutorial (Figura 21).

Figura 20 - Tutorial



Fonte: Próprios autores

Figura 21 – Explicação do tutorial



Fonte: Próprios autores

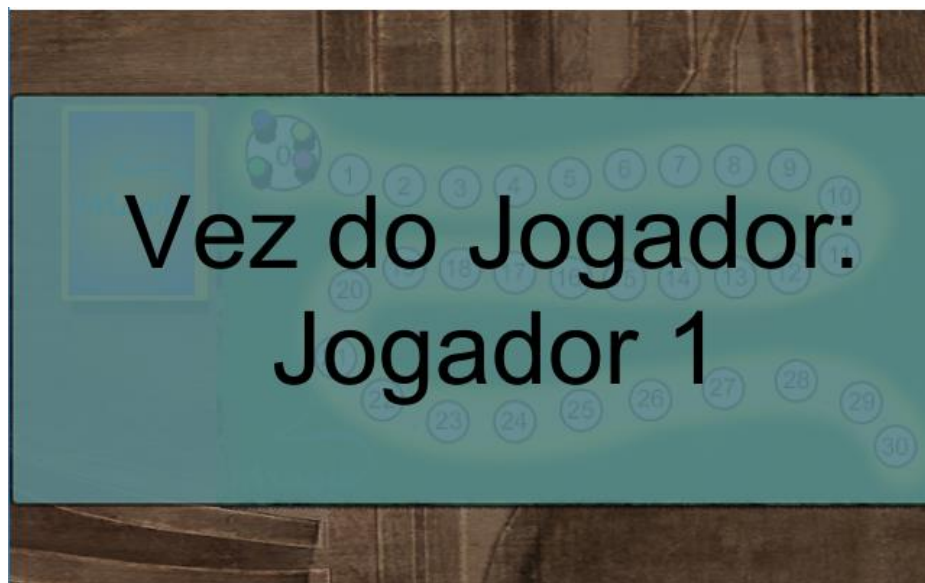
Quando todos os jogadores estão prontos para jogar, aparece uma contagem regressiva (Figura 22). Desse modo o jogador pode ter um tempo para desistir caso tenha algo errado com a configuração do seu dispositivo.

Figura 22 – Aguardando o jogo ser iniciado



Fonte: Próprios autores

Figura 23 – Indicação do narrador



Fonte: Próprios autores

No início de toda rodada, é exibido aos jogadores (Figura 23) quem será o narrador da rodada. O nome indicado é o nome que o jogador selecionou. Era planejado implementar uma entrada em uma sala preparada para o jogo. Algumas cadeiras, uma mesa, um sofá e uma prateleira foram modelados, mas foram retirados devido à necessidade de focar em outras partes do projeto.

A imagem exibida (Figura 24), demonstra o estado inicial do dispositivo do jogador com as quatro cartas. É possível, no momento adequado, selecionar a carta

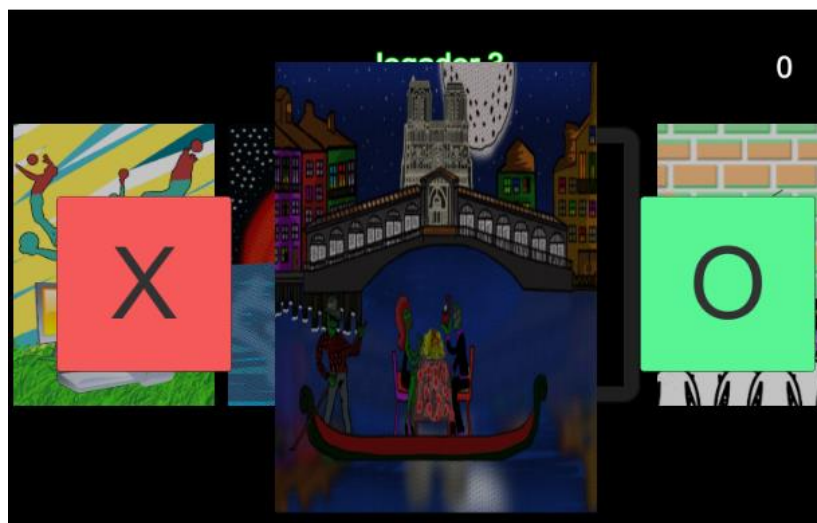
com um toque, exibindo duas opções (Figura 25), um botão para confirmar a ação de enviá-la ao tabuleiro e outra para cancelar o envio da carta.

Figura 24– Mão do jogador



Fonte: Próprios autores

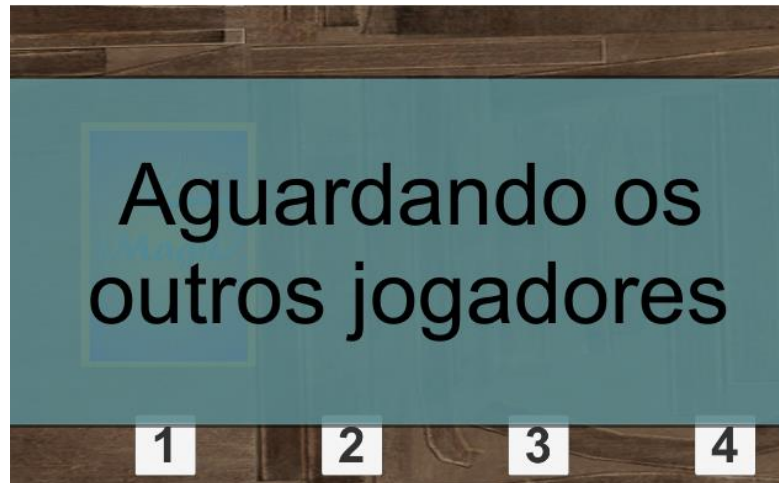
Figura 25 – Seleção da carta



Fonte: Próprios autores

A seleção da carta só é permitida quando se é o narrador, nesse caso, selecionar a carta a ser adivinhada, deixando o jogo em um estado de espera (Figura 26).

Figura 26 – Tela de aguardo



Fonte: Próprios autores

Quando o servidor receber a carta, ela será exibida na mesa virada para baixo (Figura 26), e, após o restante do jogador enviar suas respectivas cartas, estas serão embaralhadas e viradas para cima (Figura 27).

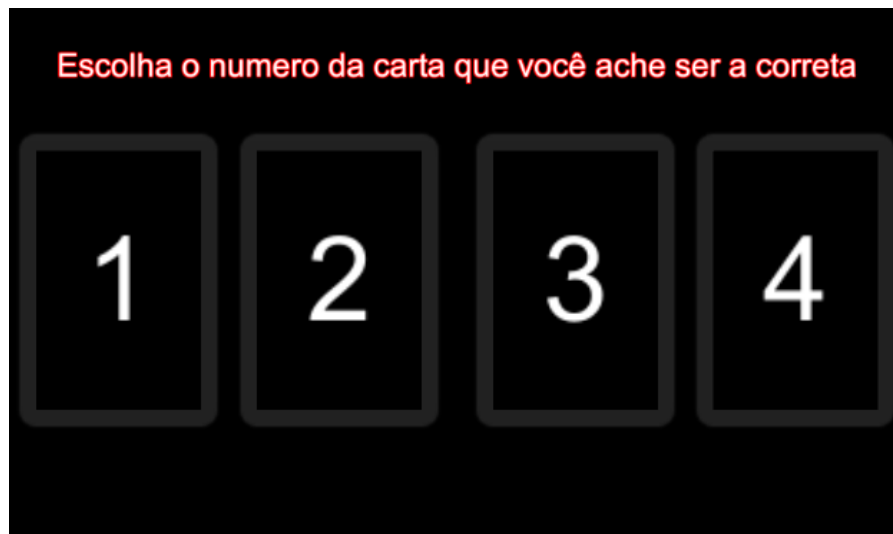
Figura 27 – Cartas na mesa



Fonte: Próprios autores

Após isso, cada jogador, com exceção do narrador da rodada, deve selecionar qual é a carta descrita (Figura 28). Foi optado por um sistema de seleção simples, que não deixasse dúvida sobre a ação a se tomar.

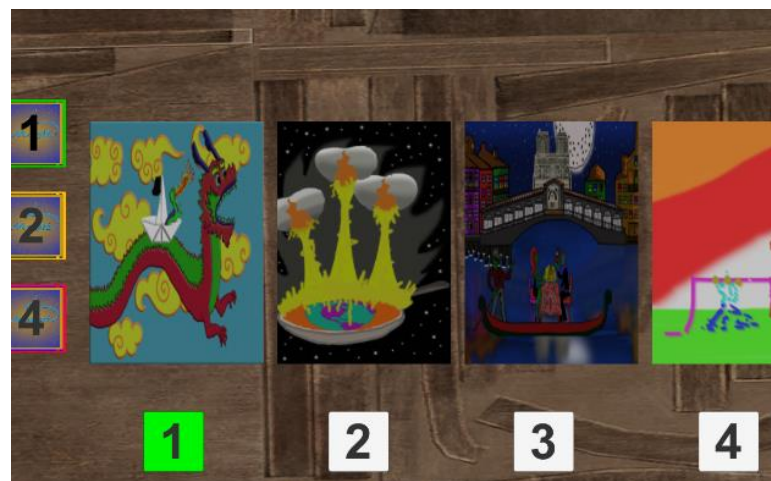
Figura 28 – Seleção da carta descrita



Fonte: Próprios autores

Por fim, é revelado a resposta correta e a opção que cada jogador selecionou é exibida a esquerda (Figura 29).

Figura 29 - Resultados



Fonte: Próprios autores

Os pontos então são somados, se o narrador conseguiu com sua descrição que todos pelo menos um dos jogadores, mas não todos, acertassem sua carta, ele ganhará três pontos. O jogador que acertar a carta descrita, também ganhar três pontos. Além disso, independente se o jogador for narrador ou não, receberá um ponto toda vez que a carta for selecionada.

No final da rodada, as peças de cada jogador serão movimentadas de acordo com sua pontuação, para cada um ponto, a peça moverá uma casa. Cada jogador possui sua peça correspondendo a sua cor (Figura 30). E uma nova rodada será iniciada

Figura 30 - Tabuleiro



Fonte: Próprios autores

5 CONSIDERAÇÕES FINAIS

O propósito inicial do projeto foi alcançado através do jogo, que pode ser baixado em desenvolvido pelo grupo, mesmo com o número reduzido de testes realizados foi possível perceber que iMagit foi uma experiência divertida para todos os jogadores e aproximou pessoas que sequer tinham curiosidade de conhecer sobre jogos de tabuleiro.

Algumas alterações foram realizadas para melhorar a experiência do usuário, algumas removeram funcionalidades, enquanto outras adicionaram boas características ao jogo que visaram torna-lo mais acessível e divertido.

Foi planejado que o jogador, que estivesse usando seu aparelho móvel para jogar, pudesse utilizar a câmera de seu aparelho para que a imagem captada por ela fosse utilizada como plano de fundo em tempo real durante o jogo. Esse sistema foi pensado para aprimorar a percepção do jogador de que o jogo não é unicamente digital e foi feito para jogar com o mundo e as pessoas a sua volta.

Porém utilizar a câmera do dispositivo mostrou-se um efeito pesado em conjunto com o processamento do jogo e as chamadas de rede, tornando o jogo um pouco lento mesmo em dispositivos considerados com alta capacidade de processamento. Outra limitação decorrente do uso dessa função é a necessidade do aplicativo pedir autorização para uso de câmera no dispositivo, isso faz com que, durante a publicação do jogo, seja requisitado uma política de privacidade, a qual deve estar disponível online para ser lida por qualquer usuário que queira instalar o aplicativo.

Visto a inexperiência do grupo em assuntos burocráticos como esse apresentado, somado à ineficiência do processamento consumido pela utilização da câmera, foi optado pela completa remoção dessa característica do jogo.

No planejamento do projeto havia sido decidido que *WiFi Direct* seria o modo de comunicação utilizado entre dispositivos para que o jogo fosse realizado, porém descobrimos mais futuramente que o jogo poderia ser implementado usando sistemas de comunicação de rede pelo sistema integrado da plataforma Unity, chamado UNET.

A utilização desse sistema permitiu que o jogo pudesse também ser jogado em computadores pessoais, utilizando os sistemas operacionais Windows, Linux ou MacOS. Desse modo os jogadores poderiam utilizar um computador como tabuleiro

conectado à uma TV, criando um tabuleiro gigante para todos poderem acompanhar o jogo e tornando a experiência muito mais convidativa e interativa.

O sistema como um todo foi muito moldado se baseando no sistema específico de jogo do iMagit, um avanço natural dele seria torna-lo adaptável à novos tipos de peças, tabuleiros dinâmicos, tabuleiros adicionais, maior quantidade de jogadores, de modo que possa se tornar a plataforma definitiva para vários tipos de jogos do segmento.

Com o avanço da plataforma seria inevitável a implementação do sistema de loja para que o jogador pudesse comprar os jogos que deseja jogar na plataforma, bem como pacotes de expansão para os jogos que já tem acesso. Pacotes de expansão são peças, cartas, tabuleiros ou qualquer outro tipo de recurso adicional que pode ser integrado a um jogo já existente, introduzindo novas mecânicas e/ou modalidades de jogo.

Com o sistema de loja implantado, seria possível buscar parcerias com produtoras e editoras que publicam jogos de tabuleiro oferecendo a possibilidade de fazer o porte do jogo para a versão digital, aproveitando todas as características de interação pessoal que cada jogo oferece e apenas utilizando os dispositivos o mínimo possível para que o jogo seja jogado.

Após a experiência de produzir o projeto durante esse semestre, como Trabalho de Conclusão de Curso, planejamos para um próximo projeto para que o jogo esteja disponível em mais plataformas.

O jogo pode ser baixado em:

<https://play.google.com/store/apps/details?id=com.swartgames.imaginarium>

REFERÊNCIAS

BODOGAMI. Venda e distribuição do jogo dixit. Disponível em:

<https://www.bodogami.com.br/l/31893132945>

Acesso em: 14 abr. 2017

GALÁPAGO JOGOS. Dixit regras do jogo. Disponível em:

<http://galapagosjogos1.hospedagemdesites.ws/PDFs/Jogo-Dixit-regras.pdf>

Acesso em: 12 abr. 2017

NOVAK, Jeannie - Desenvolvimento de Games. 2 ed. São Paulo. Cengage Learning. 2010.

PALLETON. Palheta de cores. Disponível em:

<http://paletton.com/#uid=73u0u0kpg++d-+KjO+Yvd+WQ4ZA>

Acesso em: 19 abr. 2017

SIOUX, BLEND, ESPM. Pesquisa Game Brasil 2017. Disponível em:

<https://www.pesquisagamebrasil.com.br/pesquisagamebrasilgratis>

Acesso em: 15 mar. 2017.

APÊNDICE A – CÓDIGO FONTE

BaralhoScript.cs - Contém criação e manipulação de baralho, como sacar cartas e embaralhar

```
public class BaralhoScript : MonoBehaviour {

    [SerializeField]
    int _totalCartas;
    public int TotalCartas
    {
        get
        {
            return _totalCartas;
        }
        set
        {
            _totalCartas = value;
        }
    }

    [SerializeField]
    GameObject CartaPrefab;

    [SerializeField]
    DescarteScript Descarte;

    List<GameObject> Cartas;
    GameObject CartaAberta;

    public DirectorScript Director;

    // Use this for initialization
    void Start () {
        NovoBaralho();
    }

    public void NovoBaralho()
    {
        Cartas = new List<GameObject>();
        for (int x = 0; x < TotalCartas; x++)
        {
            GameObject novaCarta = Instantiate(CartaPrefab, gameObject.transform,
false);
            novaCarta.GetComponentInChildren<CartaScript>().Index = x;

novaCarta.GetComponentInChildren<CartaScript>().TrocaFrente(Director.PegaCarta());
            novaCarta.transform.localPosition = Vector3.up * 0.005f * x;
            Cartas.Add(novaCarta);
        }
        Embaralhar();
    }

    public void Embaralhar()
    {
        Cartas.Shuffle();
        // Reorganizando no visual
        for (int x = 0; x < Cartas.Count; x++)
        {
```



```

        Cartas[x].transform.localPosition = Vector3.up * 0.005f * x;
    }
}

public void EmbaralharComAnimacao()
{
    // TODO: Animacao embaralhar...
    Cartas.Shuffle();

    // Reorganizando no visual
    for (int x = 0; x < Cartas.Count; x++)
    {
        Cartas[x].transform.localPosition = Vector3.up * 0.005f * x;
        Cartas[x].GetComponentInChildren<CartaScript>().Embaralhar();
    }
}

public void ComprarCartaAberta()
{
    if (CartaAberta != null)
    {
        SaidaCartaAberta();
    }
    else if (Cartas != null && Cartas.Count > 0)
    {
        CartaAberta = Cartas[Cartas.Count - 1];
        Cartas.RemoveAt(Cartas.Count - 1);
        CartaAberta.GetComponentInChildren<Animator>().SetTrigger("DrawAberto");
    }
}

public void SaidaCartaAberta()
{
    if (CartaAberta != null)
    {
        CartaAberta.GetComponentInChildren<Animator>().SetTrigger("SaidaDrawAberto");
        CartaAberta = null;
    }
}

public Texture2D ComprarCartaFechada()
{
    if (CartaAberta != null)
    {
        SaidaCartaAberta();
    }
    else if (Cartas != null && Cartas.Count > 0)
    {
        GameObject carta = Cartas[Cartas.Count - 1];
        Cartas.RemoveAt(Cartas.Count - 1);
        carta.GetComponentInChildren<Animator>().SetTrigger("DrawFechado");
        return carta.GetComponentInChildren<CartaScript>().Frente;
    }
    return null;
}

IEnumerator DoSomethingInAWhile()
{
    yield return new WaitForSeconds(1);
}

```

```

public List<Texture2D> SacarCartas(int quantidade)
{
    var ret = new List<Texture2D>();
    if (quantidade < 1 || quantidade > 4)
    {
        return ret;
    }
    while (quantidade-- > 0)
    {
        var aux = ComprarCartaFechada();
        if (aux != null)
        {
            ret.Add(aux);
        }
        else
        {
            return null;
        }
    }
    return ret;
}

public void Reembarelhar(List<GameObject> cartas)
{
    Cartas.AddRange(cartas);
    Embaralhar();
}

}

static class Extensions
{
    //Extensão
    public static void Shuffle<T>(this IList<T> list)
    {
        int n = list.Count;
        while (n > 1)
        {
            n--;
            int k = UnityEngine.Random.Range(0, n + 1);
            T value = list[k];
            list[k] = list[n];
            list[n] = value;
        }
    }
}

```

VotacaoScript.cs - Coordena as cartas que entram para votar e os tokens

```

public class Voto
{
    public int _jogador;
    public int _voto;
    public Color _cor;

    public Voto(int jogador, int voto, Color cor)
    {
        _jogador = jogador;
        _voto = voto;
    }
}

```

```

        _cor = cor;
    }
}

public class CartaVotacao
{
    public int _jogador;
    public Texture2D _carta;

    public CartaVotacao(int jogador, Texture2D carta)
    {
        _jogador = jogador;
        _carta = carta;
    }
}

public class VotacaoScript : MonoBehaviour {

    public List<CartaScript> CartasObjetos;

    public List<TokenVotacaoTabuleiroScript> Tokens;

    public List<Image> Indicadores;

    List<CartaVotacao> CartasColocadas;
    List<Voto> votos;
    int cartaCerta;

    // Use this for initialization
    void Start () {
        CartasColocadas = new List<CartaVotacao>();
        votos = new List<Voto>();
    }

    public void InicializarVotacao(int Jogadores)
    {
        while (Indicadores.Count > Jogadores)
        {
            Indicadores[Jogadores].gameObject.SetActive(false);
            Jogadores++;
        }
    }

    public void EmbaralhaAbreCartasComDelay(float delay)
    {
        Invoke("EmbaralhaAbreCartas", delay);
    }

    public void EmbaralhaAbreCartas()
    {
        CartasColocadas.Shuffle();
        int index = 0;
        foreach(CartaVotacao cartaColocada in CartasColocadas)
        {
            CartasObjetos[index].TrocaFrente(cartaColocada._carta);
            CartasObjetos[index].AnimacaoVirarVotacao();
            index++;
        }
    }

    public int EntraCarta(Texture2D carta, int jogador)
    {

```

```

    int indexCarta = CartasColocadas.Count;
    if (indexCarta == 0)
    {
        cartaCerta = jogador;
    }
    if (indexCarta < 4)
    {
        CartasColocadas.Add(new CartaVotacao(jogador, carta));
        CartasObjetos[indexCarta].TrocaFrente(carta);
        CartasObjetos[indexCarta].AnimacaoEntraVotacao();
    }
    return indexCarta + 1;
}

public int RecebeVoto(int jogador, int voto, Color cor)
{
    int indexVotos = votos.Count;
    if (jogador < 4 && votos.Count(x => x._jogador == jogador) == 0)
    {
        votos.Add(new Voto(jogador, voto, cor));
        Tokens[indexVotos].EntraToken(cor);
    }
    return votos.Count + 1;
}

public void RevelaVotosComDelay(float delay)
{
    Invoke("RevelaVotos", delay);
}

public void RevelaVotos()
{
    Indicadores[CartasColocadas.FindIndex(x => x._jogador == cartaCerta)].color =
Color.green;
    for(int i = 0; i < votos.Count; i++)
    {
        Tokens[i].RevelarToken("" + votos[i]._voto);
    }
}

public int[] ContarPontos()
{
    int[] resultado = new int[4];
    bool todosCertos = true;
    bool todosErrados = true;
    foreach(Voto voto in votos)
    {
        if (voto._jogador != CartasColocadas[voto._voto - 1]._jogador)
        {
            resultado[CartasColocadas[voto._voto - 1]._jogador] += 1;
        }
        if (CartasColocadas[voto._voto - 1]._jogador == cartaCerta) {
            resultado[voto._jogador] += 3;
            todosErrados = false;
        }
        else
        {
            todosCertos = false;
        }
    }
    if (!todosCertos && !todosErrados)
    {

```

```

        resultado[cartaCerta] += 3;
    }
    return resultado;
}

public List<Texture2D> ResetVotacao()
{
    List<Texture2D> ret = new List<Texture2D>();
    // Reset
    foreach (Image Indicador in Indicadores)
    {
        Indicador.color = Color.white;
    }
    for (int i = 0; i < votos.Count; i++)
    {
        Tokens[i].ResetarToken();
    }
    foreach (CartaScript carta in CartasObjetos)
    {
        if(carta.Frente != null)
            ret.Add(carta.Frente);
        carta.ResetaVotacao();
    }
    CartasColocadas = new List<CartaVotacao>();
    votos = new List<Voto>();
    return ret;
}
}

public class GamePlayerScript : NetworkBehaviour
{
    public delegate void ZoomFade();

    List<CapsulaCartaScript> CartasMao;

    public DirectorScript director;
    DirectorScript Director
    {
        get
        {
            if (director == null)
                director = GameObject.Find("Director").GetComponent<DirectorScript>();
            return director;
        }
    }
}

```

GamePlayerScript.cs - Objeto de rede que representa o jogador, tem as funções de comunicação de rede e manipulação da interface do jogador.

```

[SyncVar]
public bool isTabuleiro;
[SyncVar]
public bool isLocal;
[SyncVar]
int zoomIndex;
[SyncVar]
public bool PodeJogar;

[SyncVar]

```

```

public string Name;
[SyncVar]
public Color Color;
[SyncVar(hook = "OnScore")]
public int Score;
[SyncVar]
public int ID;

public int[] Mao;
[SyncVar]
public int QtdCartas;

// Helper
int votoMaximo;

public GamePlayerScript()
{
    Mao = new int[4] { -1, -1, -1, -1 };
    QtdCartas = 0;
    votoMaximo = 0;
    CartasMao = new List<CapsulaCartaScript>();
    zoomIndex = -1;
    PodeJogar = false;
}

public override void OnStartLocalPlayer()
{
    base.OnStartLocalPlayer();
    if (!isTabuleiro)
    {
        Director.CameraJogador();
        Director.VincularUI(this);
    }
    if (!isTabuleiro)
    {
        CartasMao.Add(GameObject.Find("CapsulaCarta1").GetComponent<CapsulaCartaScript>());
        CartasMao.Add(GameObject.Find("CapsulaCarta2").GetComponent<CapsulaCartaScript>());
        CartasMao.Add(GameObject.Find("CapsulaCarta3").GetComponent<CapsulaCartaScript>());
        CartasMao.Add(GameObject.Find("CapsulaCarta4").GetComponent<CapsulaCartaScript>());
    }
}

public override void OnStartServer()
{
    base.OnStartServer();
    if (!isTabuleiro)
        Director.ConectaJogador(this);
}

public override void OnStartClient()
{
    base.OnStartClient();
}

public void OnScore(int newScore)
{
    if (isLocalPlayer)

```

```

    {
        Score = newScore;
        Director.AtualizaPontuacao(newScore);
    }
}

[ClientRpc]
public void RpcAdicionarCartas(int indiceCarta)
{
    if (isLocalPlayer)
    {
        for (var i = 0; i < 4; i++)
        {
            if (Mao[i] < 0)
            {
                Texture2D carta = Director.PegaCarta(indiceCarta);
                Mao[i] = indiceCarta;

                CartasMao[i].GetComponentInChildren<CartaScript>().TrocaFrente(carta);

                CartasMao[i].GetComponentInChildren<CartaScript>().AnimacaoEntraMao(2f + (i / 3f));
                break;
            }
        }
    }
}

[ClientRpc]
public void RpcComecaVotacao(int qtdPlayers)
{
    if (isLocalPlayer)
    {
        votoMaximo = qtdPlayers;
        Director.MoverCameraParaTokens();
    }
}

public void ZoomCarta(int index)
{
    if (isLocalPlayer)
    {
        if (Mao[index] >= 0)
        {
            CartasMao[index].ZoomIn();
            zoomIndex = index;
        }
    }
}

public void SairZoom()
{
    if (isLocalPlayer)
    {
        CartasMao[zoomIndex].ZoomOut();
        zoomIndex = -1;
    }
}

public void EnviaCartaZoom(ZoomFade callback)
{
    if (isLocalPlayer && zoomIndex >= 0 && PodeJogar)
    {

```

```

CartasMao[zoomIndex].GetComponentInChildren<CartaScript>().AnimacaoSairMao();

CmdEnviarCarta(Director.PegaIndice(CartasMao[zoomIndex].GetComponentInChildren<CartaSc
ript>().Frente));
    Mao[zoomIndex] = -1;
    zoomIndex = -1;
    if (callback != null)
        callback();
}
}

public void EnviarVoto(int voto)
{
    if (PodeJogar && voto <= votoMaximo)
    {
        CmdEnviarVoto(voto);
        PodeJogar = false;
        Director.MoverCameraParaMao();
    }
}

[Command]
public void CmdEnviarCarta(int cartaIndice)
{
    if (PodeJogar)
    {
        Director.RecebeCartaJogador(cartaIndice, this);
    }
}

[Command]
public void CmdEnviarVoto(int voto)
{
    Director.RecebeTokenVotacao(voto, this);
}
}

static class ExtensionsJogador
{
    //Extensão
    public static int CardCount(this int[] mao)
    {
        return mao.Where(x => x >= 0).Count();
    }
}
}

```

DiretorScript.cs - Responsável por dirigir o andamento do jogo, tanto do lado do cliente como no servidor, contém a máquina de estado do jogo e controla a utilização de todas as outras áreas do jogo.

```

enum GAMESTATE
{
    LOBBY,
    PLAYING,
    VEZJOGADOR,
    CARTASVOTACAO,
    TOKENVOTACAO,
    PONTUACAO,
}

```



```

    FIM
}

public class DirectorScript : MonoBehaviour {

    [Header("Camera")]
    [SerializeField]
    CameraScript Camera;
    [Space(10)]

    [Header("PopUp Informativo")]
    [SerializeField]
    PopUpScript PopUp;
    [Space(10)]

    [Header("Baralho")]
    [SerializeField]
    List<Texture2D> CartasParaMontarBaralho;
    [SerializeField]
    BaralhoScript BaralhoDoJogo;
    [SerializeField]
    DescarteScript Descarte;
    [Space(10)]

    [Header("Elementos do Jogador")]
    [SerializeField]
    List<GameObject> PlayerCards;
    [SerializeField]
    List<GameObject> PlayerPecas;
    [Space(10)]
    [SerializeField]
    Text NomeJogador;
    [SerializeField]
    Text PontosJogador;
    [SerializeField]
    Outline OutlineJogador;
    [Space(10)]

    [Header("Texturas")]
    [SerializeField]
    Texture2D PlayerCardTexture;
    [Space(10)]

    [Header("Areas de Jogo")]
    [SerializeField]
    GameObject Lobby;
    [SerializeField]
    Text LobbyInfoText;
    [Space(10)]
    [SerializeField]
    GameObject Tabuleiro;
    [SerializeField]
    GameObject Votacao;
    [SerializeField]
    GameObject Mao;

    List<Texture2D> CartasUsadasParaMontarBaralho;
    List<Texture2D> LISTAFIXADECARTAS;
    List<GamePlayerScript> Jogadores;

    GamePlayerScript JogadorDaVez;

```

```

//LobbyManager Rede;
bool isServer;

GAMESTATE GameState;

// TESTE DA CAMERA
public RawImage CameraBackground;

void Awake()
{
    Jogadores = new List<GamePlayerScript>();
    // Rede = LobbyManager.s_Singleton;
    isServer = false;

    CartasUsadasParaMontarBaralho = new List<Texture2D>();
    LISTAFIXADECARTAS = new List<Texture2D>(CartasParaMontarBaralho);

    GameState = GAMESTATE.LOBBY;
    PopUp.ShowText("Aguardando os jogadores");
    Reset();
    Invoke("StartFromLobby", 10f);
}

public void ComecarServer()
{
    isServer = true;
    NomeJogador.text = "TABULEIRO";
    PontosJogador.text = "";
}

public Texture2D PegaCarta()
{
    if (CartasParaMontarBaralho.Count == 0)
    {
        Reset();
    }
    Texture2D carta = CartasParaMontarBaralho[0];
    CartasUsadasParaMontarBaralho.Add(carta);
    CartasParaMontarBaralho.RemoveAt(0);
    return carta;
}

public Texture2D PegaCarta(int indice)
{
    return LISTAFIXADECARTAS[indice];
}

public int PegaIndice(Texture2D carta)
{
    return LISTAFIXADECARTAS.IndexOf(carta);
}

public void Reset()
{
    CartasParaMontarBaralho.AddRange(CartasUsadasParaMontarBaralho);
    CartasUsadasParaMontarBaralho.Clear();
}

public void ConectaJogador(GamePlayerScript jogador)

```

```

{
    if (GameState == GAMESTATE.LOBBY)
    {
        int JogadorAtual = Jogadores.Count;
        if (JogadorAtual < 4)
        {
            Jogadores.Add(jogador);
            PlayerPecas[jogador.ID].GetComponent<Renderer>().material.color =
Jogadores[JogadorAtual].Color;
            PlayerPecas[jogador.ID].GetComponent<Renderer>().enabled = true;
        }
        Jogadores = Jogadores.OrderBy(x => x.ID).ToList();
    }
}

IEnumerator DoSomethingInAWhile(int seconds)
{
    yield return new WaitForSeconds(seconds);
}

public void StartFromLobby()
{
    PopUp.Hide();
    if (isServer)
    {
        Votacao.GetComponent<VotacaoScript>().InicializarVotacao(Jogadores.Count);
        GameState = GAMESTATE.PLAYING;

        PopUp.ShowTextForSeconds("Distribuindo as Cartas", 3);
        foreach (var jogador in Jogadores)
        {
            jogador.PodeJogar = false;
        }
        JogadorDaVez = Jogadores[0];
        JogadorDaVez.PodeJogar = true;
        Invoke("DistribuirCartas", 4f);
    }
}

public void AguardaJogador()
{
    GameState = GAMESTATE.VEZJOGADOR;
    PopUp.ShowText("Vez do Jogador: " + JogadorDaVez.Name);
    foreach (var jogador in Jogadores)
    {
        jogador.PodeJogar = false;
    }
    JogadorDaVez.PodeJogar = true;
}

public void RecebeCartaJogador(int cartaIndex, GamePlayerScript jogador)
{
    Texture2D carta = LISTAFIXADECARTAS[cartaIndex];
    // Recebe a Carta de um jogador
    if (GameState == GAMESTATE.VEZJOGADOR && JogadorDaVez.ID == jogador.ID)
    {
        GameState = GAMESTATE.CARTASVOTACAO;
        Camera.MoverParaVotacao();
        Votacao.GetComponent<VotacaoScript>().EntraCarta(carta, jogador.ID);
        PopUp.ShowText("Aguardando os outros jogadores");
        foreach (var jogadorNaoAtual in Jogadores)

```

```

        {
            jogadorNaoAtual.PodeJogar = true;
        }
        jogador.PodeJogar = false;
        jogador.QtdCartas--;
        // TODO: Avisar Jogadores para jogar cartas
    } else if (GameState == GAMESTATE.CARTASVOTACAO && JogadorDaVez.ID !=
jogador.ID)
    {
        jogador.PodeJogar = false;
        jogador.QtdCartas--;
        if (Votacao.GetComponent<VotacaoScript>().EntraCarta(cartas, jogador.ID) >=
Jogadores.Count)
        {
            PopUp.Hide();
            GameState = GAMESTATE.TOKENVOTACAO;
            Votacao.GetComponent<VotacaoScript>().EmbaralhaAbreCartasComDelay(3f);
            foreach (GamePlayerScript jog in Jogadores.Where(x => x.ID !=
JogadorDaVez.ID))
            {
                jog.PodeJogar = true;
                jog.RpcComecaVotacao(Jogadores.Count);
            }
        }
    }
}

public void RecebeTokenVotacao(int voto, GamePlayerScript jogador)
{
    if (GameState == GAMESTATE.TOKENVOTACAO)
    {
        int totalVotos =
Votacao.GetComponent<VotacaoScript>().RecebeVoto(jogador.ID, voto, jogador.Color);
        if (totalVotos >= Jogadores.Count)
        {
            GameState = GAMESTATE.PONTUACAO;
            Votacao.GetComponent<VotacaoScript>().RevelaVotosComDelay(2f);
            Camera.MoverParaTabuleiroComDelay(ContarPontos, 12f);
        }
    }
}

public void ContarPontos()
{
    int[] totalPontos = Votacao.GetComponent<VotacaoScript>().ContarPontos();
    foreach (GamePlayerScript jogador in Jogadores)
    {
        jogador.Score += totalPontos[jogador.ID];
        if (jogador.Score > 30)
        {
            jogador.Score = 30;
            EncerraJogo();
            return;
        }
    }
}

PlayerPecas[jogador.ID].GetComponent<PecaTabuleiroScript>().IrParaCasa(jogador.Score);
}

List<Texture2D> CartasParaDescarte =
Votacao.GetComponent<VotacaoScript>().ResetVotacao();
float delay = 0f;
foreach (Texture2D cartaParaDescarte in CartasParaDescarte)

```

```

    {
        if (cartaParaDescarte != null)
            Descarte.Adiciona(cartaParaDescarte, delay);
        delay += 0.3f;
    }
    JogadorDaVez = Jogadores[(JogadorDaVez.ID + 1) % Jogadores.Count];
    GameState = GAMESTATE.PLAYING;
    DistribuirCartas();
}

public void DistribuirCartas()
{
    foreach(GamePlayerScript jogador in Jogadores)
    {
        int qtd = 4 - jogador.QtdCartas;
        var carts = BaralhoDoJogo.SacarCartas(qtd);
        if (carts != null)
        {
            foreach (Texture2D card in carts)
            {
                int indice = LISTAFIXADECARTAS.IndexOf(card);
                jogador.RpcAdicionarCartas(indice);
                jogador.QtdCartas++;
            }
        }
        else
        {
            EncerraJogoSemCartas();
            return;
        }
    }
    Invoke("AguardaJogador", 3f);
}

public void EncerraJogoSemCartas()
{
    GameState = GAMESTATE.FIM;
    PopUp.ShowTextForSeconds("Não há cartas suficientes", 5);
    Invoke("EncerraJogo", 5f);
}

public void EncerraJogo()
{
    GameState = GAMESTATE.FIM;
    GamePlayerScript vencedor = Jogadores.OrderByDescending(x => x.Score).First();
    PopUp.ShowText(vencedor.Name + " venceu com " + vencedor.Score + " Pontos");
    Invoke("ReturnToLobbyWarning", 5f);
}

public void ReturnToLobbyWarning()
{
    PopUp.ShowText("Saindo do jogo em 5 segundos");
    Invoke("ReturnToLobby", 5f);
}

public void ReturnToLobby()
{
    LobbyManager.s_Singleton.ServerReturnToLobby();
}
}

```