

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

**Estendendo o Jogo Funny Places para
Identificação de Funções Executivas**

Eduardo Junqueira Wichrowski

PROJETO FINAL DE GRADUAÇÃO

CENTRO TÉCNICO CIENTÍFICO - CTC

DEPARTAMENTO DE INFORMÁTICA

Curso de Graduação em Ciência da Computação

Rio de Janeiro, julho de 2022



Eduardo Junqueira Wichrowski

**Estendendo o Jogo Funny Places para Identificação de
Funções Executivas**

Relatório de Projeto Final, apresentado ao programa
Ciência da Computação da PUC-Rio como requisito
parcial para a obtenção do título de Bacharel em Ciência
da Computação.

Orientador: Bruno Feijó

Rio de Janeiro
julho de 2022.

Agradecimentos

À minha mãe que me incentivou desde pequeno a aprender, estudar e questionar o mundo, que me ajudou nos meus estudos e ensinou muita coisa a vida toda e que ficou por horas ouvindo eu tentando explicar o que estava fazendo para tentar organizar meus pensamentos. Ao meu pai e minha mãe que estiveram comigo ao longo dessa trajetória. Aos meus diversos amigos que não nomeio, pois a lista é enorme, que me apoiaram e estiveram comigo mantendo a minha sanidade mental. Ao Bruno Tassara que me ajudou quando precisei entender o projeto, sanou minhas dúvidas e me deu confiança. Ao meu orientador Bruno Feijó, que me incentivou especialmente nessa reta final e me deu apoio para conseguir chegar ao final desse processo.

Resumo

Wichrowski, Eduardo Junqueira. Feijó, Bruno. Estendendo o jogo Funny Places para identificação de Funções Executivas. Rio de Janeiro, 2022. 38 p. Relatório Final de Estágio Supervisionado II – Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro.

Funny Places é um jogo sério que, de forma lúdica e divertida, ajuda na identificação da perda de eficiência no uso das Funções Executivas e, possivelmente, no treinamento das mesmas. O objetivo deste projeto é estender o desenvolvimento do Funny Places, um jogo multijogador online. A principal extensão, aqui proposta, é a coleta das variáveis de analítica de jogo e quantificar certos comportamentos do paciente sendo analisado ou assistido pelo psicólogo. O trabalho também tem por objetivo ajustar e melhorar algumas funcionalidades relacionadas que ou estavam apenas conceitualizadas ou não estavam totalmente implementadas.

Palavras Chave

Funções Executivas; Jogo Multijogador Online; Analíticas de jogos; Dados.

Abstract

Wichrowski, Eduardo Junqueira. Feijó, Bruno. Extending the game Funny Places for identification of Executive Functions. Rio de Janeiro, 2022. 38 p. Relatório Final de Estágio Supervisionado II – Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro.

This project aims to extend the development of Funny Places, an online multiplayer game. Funny Places is a serious game that, in a playful and fun way, helps to identify the loss of efficiency in using Executive Functions and possibly train them. The main extension proposed here is to collect the game analytics variables and quantify certain behaviors of the patient being analyzed or assisted by the psychologist. The work also aims to adjust and improve some related features that were either just conceptualized or not fully implemented.

Key Words

Executive Functions; Online Multiplayer Game; Game Analytics; Data.

Sumário

1 Introdução	1
2 Situação Atual	1
3 Objetivos	2
4 Funções Executivas	3
5 O Sistema	4
5.1 Motor de Jogo e Assets	4
5.2 Rede	5
5.3 Mirror Networking	6
5.4 Diagrama de Classes	8
5.5 Resumo	9
5.6 Especificações	10
5.7 Atividades de desenvolvimento	11
5.8 Padrão de desenvolvimento	12
5.9 Testes	13
6 Game Analytics	13
6.1 Aplicação das Game Analytics no jogo	13
6.2 Adicionando uma nova métrica	15
6.3 Interpretador de Python para Game Analytics e questionário	16
7 O Jogo	18
7.1 O objetivo principal do jogo	18
7.2 Os Controles	19
7.3 Barra de energia	20
7.4 Carteira	22
7.5 Seta	22
7.6 Do Lado do Psicólogo	23
7.7 Questionário	24
7.8 Tela do psicólogo no jogo	25
7.9 NPCs	26
7.10 Cenas de Movimento e Tutorial	29
7.11 Final do Jogo	32
7.12 Manutenção	34
7.13 Controle de Versão	34
8 Considerações finais e Trabalhos futuros	35
9 Referências	37

Lista de Figuras

Fig 1: Modelo de conexão do Mirror [1]	6
Fig. 2 Interação das ações remotas entre cliente e servidor[7]	7
Fig. 3 ilustração do funcionamento da tag command	8
Fig. 4 ilustração das Tags de ClientRpc e TargetRpc	8
Fig. 5 Diagrama de classes atualizado, com destaque em vermelho para as classes mais alteradas.	9
Fig. 6 Visualizações dos dados de analytics e questionários	16
Fig. 7 Visualização gerada pela função visualizeOneMetricBar com a métrica de distância total percorrida	17
Fig. 8 Visualização gerada pela função visualizeOneUser com os dados das MatchAnalytics do primeiro jogador registrado.	18
Fig. 9 Lista de Tarefas com a explicação do objetivo para o jogador.	19
Fig. 10 Tela de seleção de personagem do jogador.	20
Fig. 11 Ao topo a barra de energia parcialmente esgotada e a frente o carrinho de suco.	21
Fig. 12 Figura com a seta funcionando apontando para outro jogador a distância	23
Fig. 13 Telas de dois jogadores. A tela da esquerda é o jogador visto ao longe na direita e sua seta não é visível para o jogador da direita.	23
Fig. 14 Tela de configurações no Lobby do jogo.	24
Fig. 15 Questionário que o jogador responde a pedido do psicólogo.	25
Fig. 16 Tela do servidor durante o jogo principal com o painel de controles.	26
Fig. 17 Controlador de animações genérico dos NPCs	27
Fig. 18 NPC de Dicas no seu estado normal, com um indicador acima de sua cabeça para distingui-lo	28
Fig. 19 NPC de dica após a interação com a sua dica sendo exibida na tela do jogador	28
Fig. 20 Como é organizado e definido quais são os targets do NPC que caminha.	29
Fig. 21 Mensagem do tutorial de movimentação.	30
Fig. 22 Mensagem do tutorial de reconhecimento	31

Fig. 23 Disposição dos NPCs no tutorial de reconhecimento provisório.	32
Fig. 24 Mensagem de fim de jogo para o primeiro a completar as tarefas.	32
Fig. 25 Mensagem de fim de jogo para os outros jogadores.	33
Fig. 26 Passeio de Avião	34

Lista de Tabelas

Tabela 1. Funções Executivas	4
Tabela 2. Lista de Tarefas propostas	10
Tabela 3. Cronograma de Tarefas Desenvolvidas	12
Tabela 4. Métricas de coleta do Jogo	14

1 Introdução

Com o aumento da expectativa de vida vem também a necessidade de entender melhor a terceira idade e como os efeitos do tempo afetam o ser humano. Entre as várias questões que afetam esse grupo, o declínio cognitivo, apesar de não exclusivo, é percebido de forma mais acentuada, sendo causado, em parte, pelo envelhecimento natural do cérebro.

Para se falar sobre a saúde mental de um indivíduo é importante trabalhar com um aspecto fundamental da saúde das pessoas, um conceito da neuropsicologia conhecido como Funções Executivas. Tais habilidades envolvem o controle cognitivo necessário para realizar um objetivo específico, permitindo que o indivíduo defina metas, inicie ações, iniba outras, planeje, alterne e monitore ações. Esse desempenho cognitivo pode ser observado através de outros aspectos como: funcionamento cognitivo global, aprendizagem, memória de trabalho, reconhecimento, evocação tardia, atividades da vida diária instrumentais e básicas. Entre os idosos estas funções se deterioram com facilidade, especialmente em casos como o de portadores da doença de Alzheimer.

Existem atualmente jogos e outros aplicativos que prometem exercitar o cérebro, melhorar desempenho e até aumentar seu QI. A maioria desses aplicativos focam em pequenos jogos que alegam exercitar a memória e a rapidez de raciocínio. Apesar de superficiais em sua maioria, estudos apontam que jogos de fato podem ser utilizados para beneficiar a mitigação do declínio cognitivo.

Também deve ser considerada a importância da interação social, apesar de não haver muitos artigos que diretamente relacionem a prevenção do declínio cognitivo e a interação social, na área de psicologia e sociologia é uma noção bastante aceita que as relações sociais são importantes para manter a saúde mental não apenas em idosos, mas em todas as faixas etárias.

Tendo esses conceitos em mente e com o objetivo de preencher as lacunas dos trabalhos semelhantes, o jogo "Funny Places" [1] foi desenvolvido com o foco no uso das Funções Executivas. Porém, este jogo está longe de ser terminado. O objetivo do presente projeto é dar continuidade ao jogo "Funny Places", implementando algumas funcionalidades que não foram concluídas e incrementando o que for possível para melhorar a coleta dos dados (com vistas à analítica de jogo) e a usabilidade do jogo.

2 Situação Atual

Dentre os trabalhos encontrados na literatura, dois se destacam dos demais, principalmente por fugirem da mera implementação digital de testes da

psicologia. O primeiro é o de Valladares et al. [2] que se mostra como o de maior impacto e precisão na detecção de déficit cognitivo, valendo-se de aprendizado de máquina para prever sua ocorrência. Este trabalho usa métricas consagradas na área médica e coleta variáveis que são diretamente relacionadas aos testes aplicados (tais como o número de evocações corretas ou o número de omissões em memória de curto e longo prazo).

No jogo Funny Places [1], espera-se coletar muitas outras variáveis, boa parte delas consideradas indiretas, ou seja, relacionadas a um comportamento do jogador durante a partida, mas não diretamente relacionada a algum teste. Além disso, espera-se produzir uma métrica exclusiva para cada função executiva, embora uma abordagem menos granular, com uma detecção do tipo “comprometimento cognitivo vs condição saudável” possa também ser implementada em versões futuras. O jogo Funny Places também se destaca por permitir o psicólogo configurar o jogo e interferir no ambiente durante a simulação.

O segundo trabalho a se destacar é o de Krawczyk [3], que utiliza um jogo com visual realista, excelente qualidade gráfica e atividades do dia a dia para exercitar e melhorar o desempenho dos jogadores em funções executivas. No Funny Places, espera-se também aproveitar-se do contexto. Além disso, se espera entregar os diferenciais da configuração por parte do psicólogo e da dimensão multijogador online, que pode trazer novas possibilidades de identificação e treinamento de funções executivas. Quanto à qualidade gráfica, no Funny Places, optou-se por um visual mais lúdico, pois acredita-se que o mesmo possa tornar a experiência mais agradável.

Uma maneira de comparar a abordagem proposta no Funny Places com a dos outros trabalhos é observar que aqui se está evitando a mera aplicação digital de testes classicamente feitos em consultórios, visto que a simples tradução da técnica tradicional para um meio digital não representa inovação.

Espera-se ir além do estado da arte (especialmente [2] e [3]). Para tanto, além de uma abordagem repleta de contexto, que aborda atividades relacionadas ao dia a dia, o jogo se vale de um grande número de variáveis coletadas, métricas para avaliar funções executivas individuais, possibilidade de configuração e adaptação dos testes por parte do controlador ou psicólogo e da socialização, pois trata-se de um jogo multijogador online.

3 Objetivos

O objetivo do presente trabalho é dar continuidade ao desenvolvimento do jogo Funny Places [1], que disponibiliza um ambiente lúdico para a identificação

das Funções Executivas, oferecendo o diferencial da experiência multijogador online. A principal extensão, aqui proposta, é a coleta das variáveis de analítica de jogo. Os outros desenvolvimentos visam a customização do jogo através da generalização do painel de configurações (a ser usado pelo psicólogo) e a introdução de NPCs (Non-Player Characters). O desafio do projeto é manter o controle multijogador com a introdução das extensões propostas.

4 Funções Executivas

Embora não haja um consenso quanto à conceituação das Funções Executivas (FE), elas geralmente são definidas como um conjunto de habilidades ou processos do cérebro, essenciais ao controle voluntário para realizar um objetivo específico [4]. Estas habilidades permitem que o indivíduo pense antes de agir, controle suas ações, planeje e execute uma sequência de passos para realizar uma tarefa e mantenha a atenção e o foco. São essenciais para a realização de tarefas do dia a dia [5] e o déficit delas pode tornar difícil se concentrar, seguir instruções ou até mesmo controlar as emoções.

As funções executivas podem ser divididas em Funções Frias e Funções Quentes [6]. As frias, correspondem ao funcionamento do lobo pré-frontal do córtex cerebral e correspondem aos componentes cognitivos, estando envolvidas nos processos controlados e nas etapas necessárias para execução de um comportamento novo ou dirigido a uma meta específica. Elas englobam o planejamento, iniciativa, monitorização, memória de trabalho e atenção seletiva dividida e alternada, além da fluência verbal. Já as funções quentes respondem pela regulação emocional do comportamento e o controle inibitório, controle da impulsividade e da reatividade emocional. A empatia social, que envolve entender os comportamentos e emoções do outro também faz parte das funções executivas quentes.

Entre 5 e 13 anos, há um salto no desenvolvimento destas habilidades, que em geral apresentam um declínio perceptível acima de 60 anos.

As funções executivas são responsáveis pela iniciativa e pelo controle de uma atividade nova. Após a repetição, as funções executivas são necessárias apenas para iniciar a atividade. No processo de envelhecimento, com seu declínio, os idosos tentam realizar mais processos automáticos do que controlados, preferindo se ater a uma rotina superaprendida ao longo dos anos, por isso são tão avessos a mudanças.[6]

As funções executivas são apresentadas na Tabela 1.

Tabela 1. Funções Executivas

FE	Descrição
Memória de Trabalho	A memória de trabalho é um estoque transitório de informações na memória de curto prazo, essencial enquanto estamos realizando uma determinada tarefa e é importante para qualquer tarefa que se desenrole com o tempo.
Flexibilidade Cognitiva	A flexibilidade cognitiva é a habilidade de alternar entre ações e permite se adaptar a mudanças em um ambiente. Esta função executiva é importante para se realizar tarefas múltiplas
Autocontrole Emocional	Esta função diz respeito a capacidade de gerenciar e controlar as emoções durante uma atividade ou para atingir um objetivo. Também inclui a capacidade de se manter calmo ante um imprevisto ou contratempo. Pessoas com esta função menos desenvolvida podem experimentar mudanças repentinas de humor e sentir-se derrotadas ante um empecilho.
Atenção Seletiva	Esta função executiva, engloba a habilidade de focar a atenção em uma atividade, dispensando estímulos de distração e outras atividades.
Planejamento	Esta função responde pela capacidade de planejar e organizar uma tarefa de forma organizada e sequencial. Dito de outra forma, pode-se pensar nesta habilidade com a capacidade de mentalmente antecipar a melhor maneira de realizar uma tarefa.
Monitoração	Capacidade de auto monitorar os passos de uma tarefa para que esta seja concluída com êxito. Esta habilidade está relacionada à capacidade de avaliar se os passos tomados para a conclusão de uma tarefa são os corretos e à capacidade de detectar que alguma etapa não foi tomada corretamente

5 O Sistema

5.1 Motor de Jogo e Assets

O desenvolvimento de Lugares Divertidos foi feito no *game engine* Unity versão 2019.4.15.f1, que foi selecionado por ser uma opção de baixo custo e que permite a entrega de produtos com nível comercial, além de ter uma enorme comunidade muito ativa no desenvolvimento de games, o que facilita seu aprendizado. Outra razão para o uso do Unity é a facilidade de geração para várias plataformas (PC, celular, tablet).

Outra vantagem do Unity é a Unit Asset Store™ (<https://assetstore.unity.com/>), onde diversos componentes desenvolvidos por estúdios ou pela comunidade ficam disponíveis, alguns gratuitos e alguns pagos. Estes conteúdos são chamados de Assets. Os scripts gerados para o jogo foram feitos com C# e os

assets usados no desenvolvimento foram:

- Simple Shops (Synty Studios)
- Polygon Town (Synty Studios)
- Polygon City Characters (Synty Studios)
- Simple UI (Unruly Games)
- 500 Resource Icons (Poneti)
- Mirror (Vis2K)
- Simple Airport (Synty Studios)
- Bird Cute Series (Meshtint Studio)
- Arrow (Bruno Tassara)
- Limão Carrinho (Giovanna Camocardi)

Para a movimentação dos personagens foram usadas animações disponíveis no site <https://www.mixamo.com>. Estas animações estão sob uma licença que permite o livre uso e até mesmo comercialização de jogos que as utilizem, havendo restrição apenas para a revenda das próprias animações.

5.2 Rede

O Unity apresenta uma solução nativa de API de alto nível para desenvolvimento de aplicações multijogador: a Unet. Entretanto, a solução fornecida pelo Unity está marcada para ser descontinuada a partir de 2021. A nova proposta da empresa é fornecer as funcionalidades online por meio de um serviço de servidores dedicados, a qual promete ser muito completa, mas com um custo associado. Embora esta solução pareça uma boa opção para grandes jogos, com grande público, para os pequenos desenvolvedores e desenvolvimentos acadêmicos pode não ser adequada. Somado a isso, durante o desenvolvimento inicial do protótipo do jogo, esta nova solução nativa do Unity ainda não estava pronta, sendo apenas disponibilizada sua versão alpha. Pensando no desenvolvimento contínuo do jogo e para não ficar dependente de uma versão desatualizada do engine, foi buscada uma alternativa.

A alternativa escolhida foi o Mirror (<https://mirror-networking.com/>), que é praticamente idêntico ao Unet, sendo basicamente uma reescrita dele (com mais de 950 bug fixes, segundo os desenvolvedores). A possibilidade de implementação de um modelo gratuito, sem limite de jogadores simultâneos, viabilizado pelo Mirror, pesou muito na escolha. Espera-se que versões futuras do jogo Funny Places, orientadas ao treinamento de funções executivas, possam ser distribuídas para uma quantidade maior de usuários, caso no qual talvez fosse interessante ter salas de jogo dedicadas.

5.3 Mirror Networking

O Mirror Networking [10], opção selecionada para implementar as funcionalidades de rede do jogo Funny Places, usa um modelo Cliente-Servidor, com um cliente atuando como host, como pode ser visto na Fig.1.

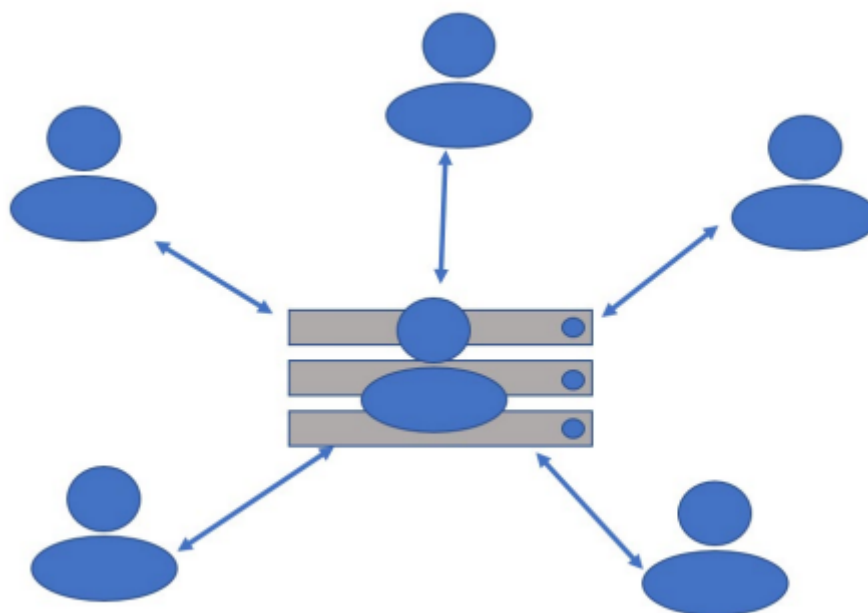


Fig. 1 Modelo de conexão do Mirror[1]

Este tipo de arquitetura tem algumas características que, numa primeira análise pode parecer desaconselhável, pois coloca um pouco mais de encargos no cliente que atua como host, além de apresentar uma ligeira vantagem para o host que jogaria sem latência, acrescido da indesejável facilidade de cheats (trapaças ou vantagens obtidas pela manipulação de variáveis no servidor).

Entretanto, para o uso esperado do Funny Places, em seu estágio atual, é este o layout perfeito – já que o host atua apenas como servidor e é ocupado pelo controlador ou psicólogo a quem cabe o controle sobre o jogo (sem interesses de realizar trapaças e sem ter objetivos de competir com os jogadores).

Caso seja necessário ter ganho de escala, o Mirror [10] pode ser usado com servidores dedicados, bem como combinado com soluções de matchmaking, permitindo que jogadores em celulares possam se conectar a seus amigos, por exemplo.

Com o uso do Mirror [10], não há necessidade de códigos diferentes para cliente e servidor – eles podem ser o mesmo projeto, razão do nome Mirror.

O Mirror [10] oferece o uso de tags no código que identificam o tipo de comunicação (Fig. 2), são elas:

- [Server] / [Client]: usadas para limitar trechos de código a rodar apenas no cliente ou apenas no servidor
- [Command]: usadas para comunicação cliente servidor
- [ClientRpc] / [TargetRpc]: usadas para comunicação servidor cliente
- [SyncVar]: usadas para sincronizar dados

Neste contexto, por exemplo, [Command] pode ser usado para que um cliente sinalize para o servidor que um objeto foi coletado. Desta forma o servidor irá informar aos clientes que aquele objeto não existe mais. [ClientRpc] pode ser usado para que o servidor informe ao cliente que deve criar algum objeto e [Syncvar] pode ser usado para sincronizar a barra de energia de diferentes jogadores, de forma que todos vejam valores atualizados de energia.

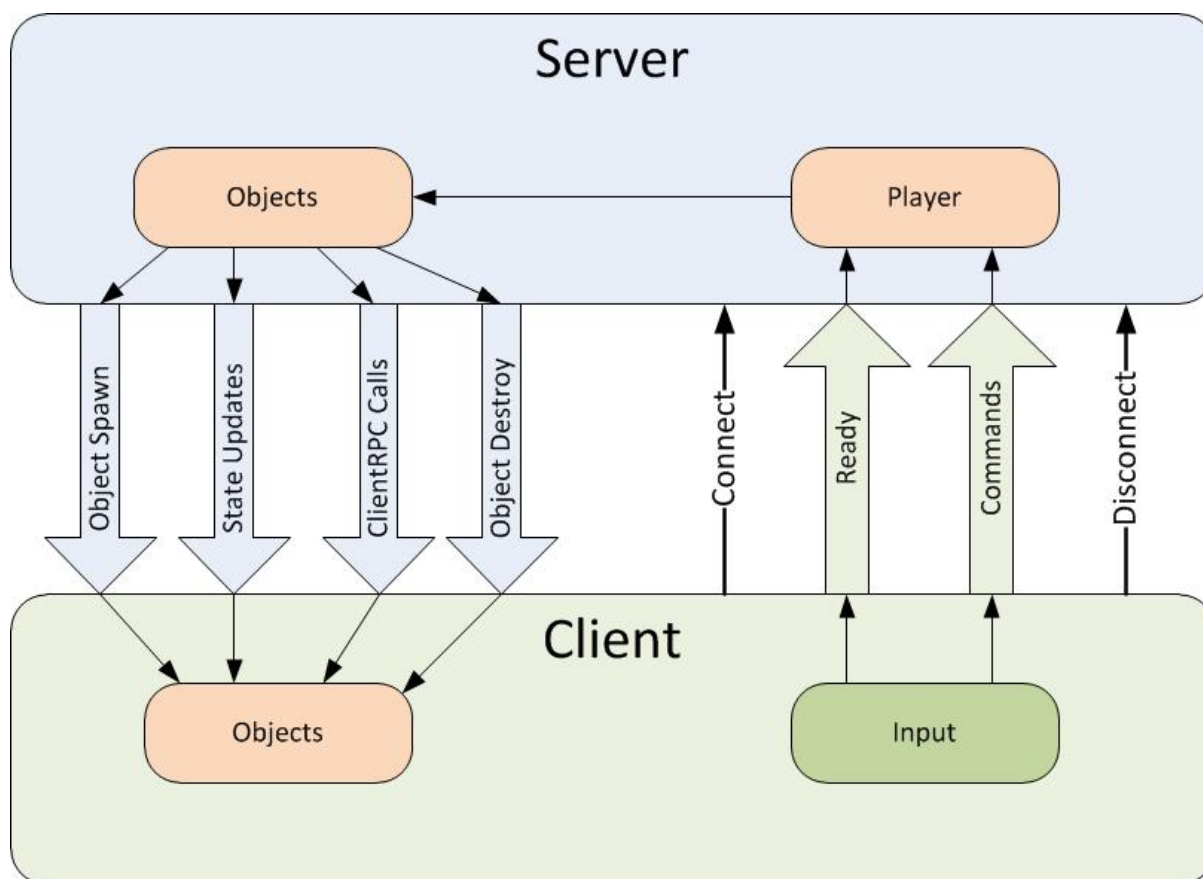


Fig. 2 Interação das ações remotas entre cliente e servidor[7]

Dentre essas tags há algumas peculiaridades necessárias de se ilustrar. Na tag de command que define um **comando** o objeto com o qual se interage precisa existir e ser visível tanto ao servidor quanto ao jogador, assim o **comando** lida com as “cópias” dos objetos que existem no servidor e não nas que existem no cliente (Fig. 3).

De forma similar as tags de ClientRpc e TargetRpc, lidam com a cópia dos objetos nos clientes, tanto os clientes que chamaram algum comando ou outros cliente que foi especificamente mencionado com o TargetRpc. Novamente por interagir com a cópia dos objetos entre quem chama o Rpc e quem recebe o Rpc é necessário que o objeto referido exista em ambos (Fig. 4).

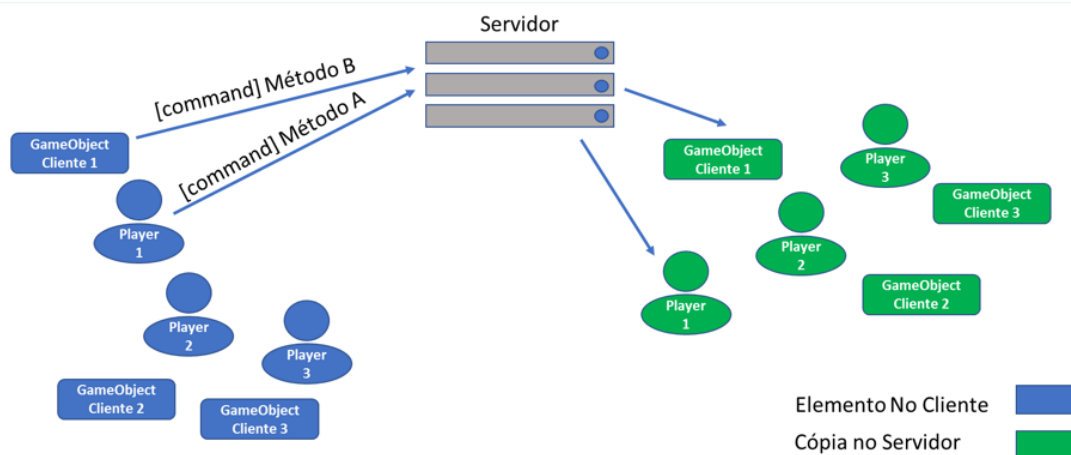


Fig. 3 ilustração do funcionamento da tag `command`[1]

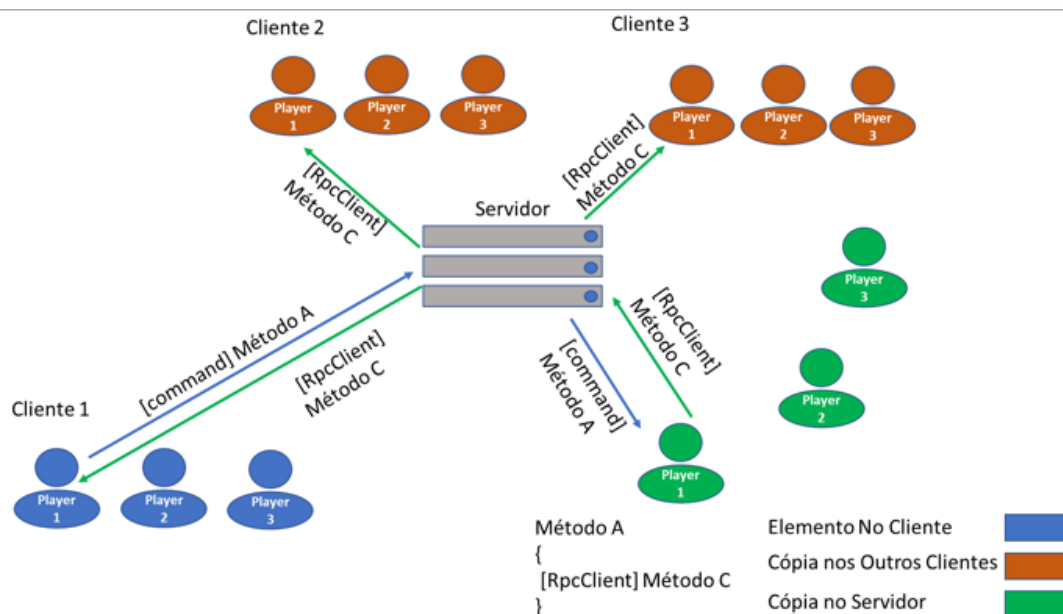


Fig. 4 ilustração das Tags de ClientRpc e TargetRpc[1]

5.4 Diagrama de Classes

O diagrama de classes (Fig. 5), com destaque em vermelho para as classes que foram mais significativamente alteradas, mostra a disposição das principais classes do jogo, que em sua maioria herdam da classe **NetworkBehaviour**, do Mirror [1], e que fornecem muitos dos comportamentos necessários para a comunicação online e a viabilização do jogo multijogador.



Fig. 5 Diagrama de classes do *Funny Place* atualizado, com destaque em vermelho para as classes mais alteradas.

5.5 Resumo

O projeto é uma extensão ao jogo original, trazendo uma ferramenta de analítica de jogo que coleta as métricas ao longo do jogo e outra ferramenta que permite a visualização dos dados gerados por essas métricas e do questionário que o

jogador responde. Além disso implementa funções e sistemas que foram idealizados ou começados, mas que não foram concluídos no jogo original.

5.6 Especificações

Quanto às especificações funcionais do sistema, as principais já foram mencionadas antes, o jogo deve ser multijogador, online e utilizando o sistema do Mirror [10] para tal. Além disso, montou-se uma tabela listando alguns dos desenvolvimentos planejados para o jogo (Tabela 2). Esta tabela foi a base para guiar os desenvolvimentos do presente projeto.

Tabela 2 Lista de tarefas propostas

Item	Descrição	Complexidade	Trabalhoso
1	Seta que aponta para jogador ficar móvel e seguir o jogador mais próximo	Média	Pouco
2	Popup da mensagem do psicólogo deve aparecer na tela por 5 s	Baixa	Pouco
3	linkar máquina de cartão de crédito com as opções setadas pelo psicólogo	Média	Médio
4	Criar os presets de de configuração do jogo	Baixa	Pouco
5	habilitar salvamento e carregamento de configurações	Baixa	Pouco
6	preparar a cena do tutorial de movimentação	Alta	Muito
7	preparar a cena do tutorial de reconhecimento	Alta	Muito
8	Popular o mapa com npcs	Baixa	Muito
9	Encontrar um modelo de gato compatível e uma animação para pulo e uma para corrida	Baixa	Médio
10	Aumentar as animações dos NPCs e flexibilizar os animator controllers para trocar a animação por meio do editor	Alta	Muito
11	Criar Npcs que se movem (sugestão é se mover acompanhando a calçada)	Alta	Muito
12	Implementar o decaimento dos itens (sorvete, refrigerante e batata frita)	Média	Médio
13	Aumentar as opções do psicólogo na fase inicial, colocando a opção de quantos itens o jogador deve buscar	Média	Pouco
14	Deixar pronta uma cena para questionários (vai entrar na mesma área onde se seleciona os tutoriais)	Média	Médio
15	Fazer o pássaro ser spawnado nas árvores (já tem modelo e as animações)	Baixa	Pouco
16	Fazer a cena do sobrevoo (3 percursos diferentes e aleatórios para o voo)	Média	Médio
17	Coletar todas as variáveis de analytics já levantadas (mesmo que não vá usar todas)	Alta	Muito
18	Colocar alguns NPCs com algum tipo de indicação (primeira sugestão é um ícone de interrogação acima da cabeça, para indicar que ele dá dicas)	Alta	Muito
19	Corrigir o script da carteira e enviar mensagens em caso de dinheiro insuficiente	Média	Médio
20	Substituir a lista de itens por sprites dos itens	Baixa	Médio
21	Viabilizar o salvamento dos dados de analytics ao final da partida (json)	Baixa	Pouco
22	Fazer o interior das casas (decidir se todas as casas precisam ser "entráveis" ou não)	Baixa	Muito
23	Substituir os Mesh Colliders	Baixa	Muito
24	Transformar a carrocinha de cachorro quente em bancada de limonada	Baixa	Pouco
25	Pensar na possibilidade de um visualizador de resultados independente (python já pensando num machine learning talvez)	Média	Muito

O desenvolvimento em sua maior parte foi baseado nesta tabela, tendo em alguns pontos se desviado para viabilizar algumas tarefas ou métricas que eram necessárias para o jogo.

Outro requerimento do jogo era que se mantivesse uma taxa de quadros de 30 quadros por segundo, o que se confirmou nos testes que é possível, mas cujos resultados diminuem conforme múltiplas instâncias do jogo são abertas na mesma máquina, porém dado que essa não é a intenção de uso não foi considerado um problema.

As métricas estabelecidas para esse projeto eram as relacionadas às funções já implementadas e as que deveriam ser implementadas, elas são: total andado no jogo, total de dinheiro gasto, total de vezes que ativou o botão de jogador mais próximo, se levou os amigos no avião no final da partida, total andado no tutorial, total de dinheiro ao final da partida, total de vezes que tentou se hidratar, total de vezes que se hidratou, uma lista contendo os valores da barra de energia quando se hidratou, quantidade de vezes que a energia se esgotou, quantidade de vezes que acionou o botão da lista de itens, quantidade de vezes que interagiu com outros jogadores.

Todas essas métricas estão atualmente sendo coletadas e guardadas.

Os principais requisitos do sistema de analítica de jogo é que os valores finais das métricas coletadas ao longo do jogo devem ser salvos na máquina do servidor, no formato JSON para possibilitar sua leitura pelo programa de análise e ter um programa de análise associado que possa ler esses dados e apresentar de forma visual.

Um requisito que o projeto não estabelece é de que o jogo precisa estar completamente implementado, não é o objetivo final desse projeto e nem o foco que o *gameplay* principal esteja concluído, foram implementadas algumas coisas nessa linha, mas apenas com o objetivo de auxiliar a coleta de métricas.

Dentre os requisitos não funcionais, o jogo deveria ter um foco em fazer a edição de opções mais fáceis e acessíveis para alguém que não é especialista, além de manter o jogo divertido e lúdico. O jogo precisa operar no windows e não perder as funcionalidades online.

5.7 Atividades de desenvolvimento

Algumas tarefas de menor prioridade, ou cujo foco era mais visual do que funcional, infelizmente não puderam ser feitas por algumas dificuldades de implementação ou tempo.

O foco do desenvolvimento foi sempre na área de coletar as variáveis de analítica dos sistemas ao entorno. Também foi possível focar nas tarefas relacionadas aos NPCs e a questão dos modos opcionais.

Não foi possível fazer o trabalho de criar os cenários interiores e resolver o problema dos mesh colliders, que teriam um trabalho muito intenso em tempo de desenvolvimento. Outras tarefas que teriam menor trabalho em si como mudar os sprites dos itens, na verdade tinham problemas mais fundos que não foram identificados nem pela lista nem na primeira parte do projeto que lidam com o sistema maior com toda a parte da coleta de itens, inventário e como os itens são tratados dentro do jogo.

Tabela 3 Cronograma de Tarefas Desenvolvidas

Cronograma	Janeiro	Fevereiro	Março	Abril	Maio	Junho	Julho
Início do entendimento do Mirror							
Projeto de teste do Mirror							
Início da implementação das Game Analytics							
Carteira							
Barra de Energia							
Mudança na implementação das Game Analytics							
Implementando Presets							
NPCs Prefab Genérico							
NPCs Andantes							
NPCs Dicas							
Salvar Game Analytics e Presets como Json							
Mudança de Cenas							
Resolvendo Conexão do Mirror no jogo							
Questionario Operante							
Visualizador de resultados							
Cena de Tutorial de Movimentação							
Cena de Tutorial de Reconhecimento							

5.8 Padrão de desenvolvimento

O motor de jogo, Unity, usa principalmente a linguagem de programação C#, linguagem essa fortemente tipada e voltada à programação orientada a objetos. O próprio Unity opera bastante em volta dos conceitos dos *game objects*, onde cada objeto recebe uma série de componentes como forma, demarcadores de colisão,

scripts que ficam associados ao objeto, um *transformer* que é o que define a posição do objeto na cena. Os conceitos da orientação a objetos guiaram bastante do padrão seguido para desenvolver o jogo.

5.9 Testes

A maior parte dos testes realizados foram testes unitários, criando um protótipo com a nova função ou alteração e testando com uma das instâncias sendo rodadas no editor do unity permitindo que os valores e logs fossem vistos em tempo real.

No caso de modificações que lidavam com objetos do jogo aos quais o aspecto multiplayer não afetava era possível até testar na cena do editor sem que houvessem jogadores, não precisando criar um protótipo, mas sempre ao fim para validar se a mudança estava consistente era gerado um protótipo e então testado num ambiente como se fosse jogado por três jogadores.

Devido a ausência de múltiplos computadores na minha rede foi necessário testar o jogo na mesma máquina utilizando múltiplas instâncias e conectado por um servidor local.

Afim de garantir que as mudanças por exemplo que geram arquivos estavam sendo apropriadamente criadas na “máquina” do servidor, foi utilizado um controle do Mirror que verifica se quem está executando o código é um cliente ou servidor, assim podendo confirmar que os arquivos eram salvos na máquina e endereço do servidor e não do cliente.

6 Game Analytics

6.1 Aplicação de Game Analytics no jogo

Uma das principais funções do projeto é a aplicação de *Game Analytics*, ou analítica de jogo, para ajudar o entendimento do psicólogo sobre os comportamentos dos jogadores. A ideia das Game Analytics é que ao se quantificar e coletar certas métricas, como o número de passos ou a quantidade de vezes que um jogador tentou executar uma ação, juntar essas informações sobre o jogador e enviar ao psicólogo para que ele possa analisar o comportamento do jogador sobre cada uma dessas métricas.

Parte do trabalho também envolve uma segunda ferramenta além do jogo em si, que recebe esses dados passados pelo psicólogo e trata eles para serem usados para visualizações como tabelas e gráficos que facilitem a interpretação dos dados pelo psicólogo.

Para que esses sistemas funcionem no jogo foi preciso criar um Script que permitisse o jogador enviar **comandos** para o servidor registrar as suas métricas

individuais e ao final da partida o servidor envia um pedido aos jogadores para que eles enviem suas métricas finais e que o servidor possa salvar a versão final das métricas individuais.

O Salvamento é feito através de arquivos Json para que possam ser lidos pela segunda ferramenta desenvolvida em Python.

Uma facilidade que esse sistema provê é que é relativamente fácil adicionar uma nova métrica. Como no exemplo da figura abaixo, temos a base da classe **Analytics** que é usada pelo script de Game Analytics. Cada um dos valores da classe, apresentada na Tabela 3, representa uma métrica do jogo, vale notar que nem todas as métricas apresentadas aqui estão operando no jogo, mas já existem visando futuras implementações.

Tabela 4. Métricas de coleta do Jogo *Funny Places*

Métrica	Descrição
M1	Distância total percorrida no jogo
M2	Total de dinheiro gasto
M3	Tempo para digitar a senha
M4	Tempo para encontrar jogadores no tutorial
M5	Distância percorrida durante o tutorial
M6	Dificuldade dos itens listados
M7	Quantidade de vezes que tentou se hidratar
M8	Quantidade de vezes que se hidratou
M9	Valor da barra de energia quando se hidratou
M10	Acionamentos do botão de interação social
M11	Acionamentos do botão de doação de cupom
M12	Acionamentos do botão de jogador mais próximo
M13	Recuperação de um item perdido
M14	Tamanho da senha digitada
M15	Razão entre o valor da compra e valor dos itens
M16	Quantidade de vezes que a energia se esgotou
M17	Ordem de aquisição do item com prioridade
M18	Obteve item de combinação
M19	Quantidade de vezes que cada botão de interação ficou disponível
M20	Quantidade de animações acionadas
M21	Créditos ao final da partida
M22	Opção de levar os amigos para o sobrevoo
M23	Quantidade de vezes que acionou a lista de itens
M24	Quantidade de vezes que acionou a dica de itens

M25	Quantidade de vezes que interagiu
-----	-----------------------------------

O script de GameAnalytics é associado a cada prefab de RoomPlayer, que cria uma série de variáveis correspondentes a cada um dos elementos da classe Analytics. O motivo de estar associado ao RoomPlayer e não ao prefab de Player em si é porque entre as cenas de gameplay, a MainTown, Tutorial e Reconhecimento, o Player é criado pelo servidor nas cenas, portanto era necessário um objeto que fosse persistente através das mudanças e que fosse associado ao player e sua conexão. Esses valores são atualizados no servidor através de comandos chamados no código de controlador do Player porém executados no RoomPlayer.

Lembrando que ao usar a Tag [Command] a função se torna um **comando** que executa do cliente para o servidor, sendo assim possível que isso seja atualizado não no cliente mas no servidor em si.

O **comando** CmdUpdateDinheiroTotalGasto é chamado em outra parte do código quando o jogador faz uma ação que gaste seu dinheiro, então ele envia as informações atualizadas para o servidor.

6.2 Adicionando uma nova métrica.

Para implementar uma nova métrica a ser coletada é necessário fazer algumas adições ao código no script específico e em outros pontos do jogo.

Primeiro é preciso no script **GameAnalytics**, que gerencia a maior parte do processo, é preciso ir na classe **Analytics**, essa classe tem a definição de todas as métricas, pela necessidade de como é salvo e pelo padrão do C# é necessário definir ali não só o nome para a nova métrica, mas o seu tipo, sendo um tipo que é aceito pelo C#. Para casos onde a métrica não um valor só, mas vários valores coletados em pontos diferentes é preciso definir essa métrica como uma Lista do tipo desejado.

Segundo, também no mesmo script, fora da classe há a criação de uma variável que será usada pelo jogador localmente e pelo servidor para guardar o valor atual que a determinada métrica tem para o jogador ao longo do jogo, é necessário instanciar essa variável, por padrão tendo o mesmo nome que a definição que existe na classe.

Em terceiro lugar, que é a parte mais difícil, é preciso identificar no resto do programa onde que essa métrica acontece. Se for algum comportamento relacionado a como o jogador interage é mais provável que seja no próprio script **PlayerController** que controla a maior parte de suas ações. Uma vez encontrado o ponto do código desejado, seja no clique com algum objeto, seja relacionado ao caminhar ou olhar. é necessário adicionar um **comando**.

Fig. 6 Visualizações dos dados de analytics e questionários.

Uma das visualizações mais básicas que o programa oferece é a de barra para uma métrica com a função `visualizeOneMetricBar` (Fig. 7). O psicólogo fornece o `DataFrame` desejado, no caso o `AllData` que lida com as `GameAnalytics` de todos os jogadores e a métrica desejada, e o programa irá retornar um gráfico de barras que mostra a distância total de cada usuário.

```
visualizeOneMetricBar(AllData, "M1_DistânciaTotalPercorrida")
```

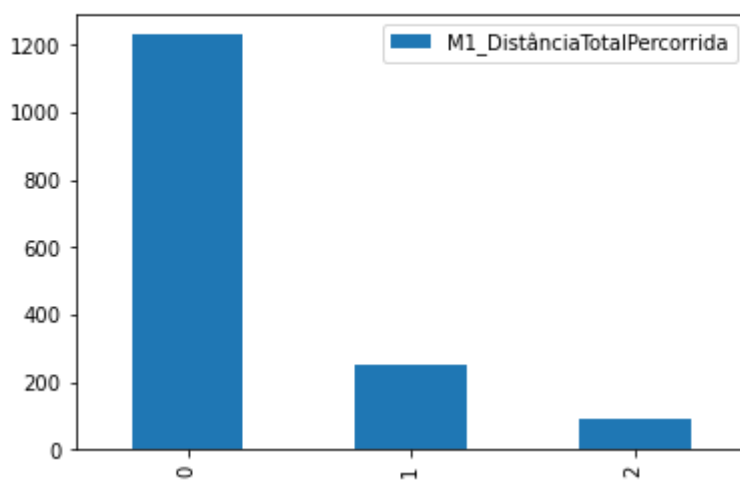


Fig. 7 Visualização gerada pela função `visualizeOneMetricBar` com a métrica de distância total percorrida

Para ver todas as métricas de um usuário específico, há a função `visualizeOneUser` (Fig. 8). Essa função tem como parâmetros o `DataFrame` desejado e qual o id do usuário que é associado ao número do arquivo.


```
visualizeOneUser(AllData,0)
```

M1_DistânciaTotalPercorrida	1230
M2_TotalDeDinheiroGasto	0
M3_TempoParaDigitarASenha	[]
M4_TempoParaEncontrarJogadoresNoTutorial	[]
M5_DistânciaPercorridaDuranteOTutorial	0
M6_DificuldadeDosItensDaLista	[]
M7_QuantidadeDeVezezQueTentouSeHidratar	0
M8_QuantidadeDeVezezQueSeHidratou	0
M9_ValorDaBarraDeEnergiaQuandoSeHidratou	[]
M10_AcionamentosDoBotaoDeInteracaoSocial	0
M11_AcionamentosDoBotaoDeDoacaoDeCupom	0
M12_AcionamentosDoBotaoDeJogadorMaisProximo	0
M13_RecuperacaoDeUmItemPerdido	[]
M14_TamanhoDaSenhaDigitada	[]
M15_RazaoEntreValorDaCompraEValorDosItens	[]
M16_QuantidadeDeVezezQueAEnergiaSeEsgotou	0
M17_OrdemEmQueAdquiriuOItemDePrioridade	0
M18_ObteveItemDeCombinacao	False
M19_QuantidadeDeVezezQueOBotaoDeInteracaoSocialFicouDisponivel	0
M20_QuantidadeDeAnimacoesAcionadas	0
M21_CreditosAoFinalDaPartida	2000
M22_OpcaoDeLevarOsAmigosAoSobrevoo	False
M23_QuantidadeDeVezezQueAcionouOBotaoDeListaDeItens	0
M24_QuantidadeDeVezezQueAcionouOBotaoDeDicaDeItem	[]
M25_QuantidadeDeVezezQueInteragiu	0

Name: 0, dtype: object

Fig. 8 Visualização gerada pela função visualizeOneUser com os dados das MatchAnalytics do primeiro jogador registrado.

7 O Jogo

7.1 O objetivo principal do jogo

Do ponto de vista do jogador, o objetivo em Funny Places é ajudar a preparar a festa de aniversário de 80 anos do seu amigo. Para ajudar a organizar a festa, o jogador deve coletar alguns itens de sua lista, que ele recebe no momento que entra no jogo junto a uma explicação do objetivo (Fig. 9). A explicação, além de contextualizar a festa, indica os itens que o jogador deve procurar pela cidade e também no final diz o nome do hotel no qual o jogador começa e ao qual ele deve retornar para entregar os itens após ter terminado de coletar todos.

Por fim, para auxiliar com o reconhecimento de qual é o hotel, no final da mensagem há também uma dica sobre este, isto é, o quadro que está no lobby do hotel. Desse modo o jogador pode confirmar que está de fato no hotel correto, sendo também algo que deve ser memorizado, incentivando o propósito do jogo.

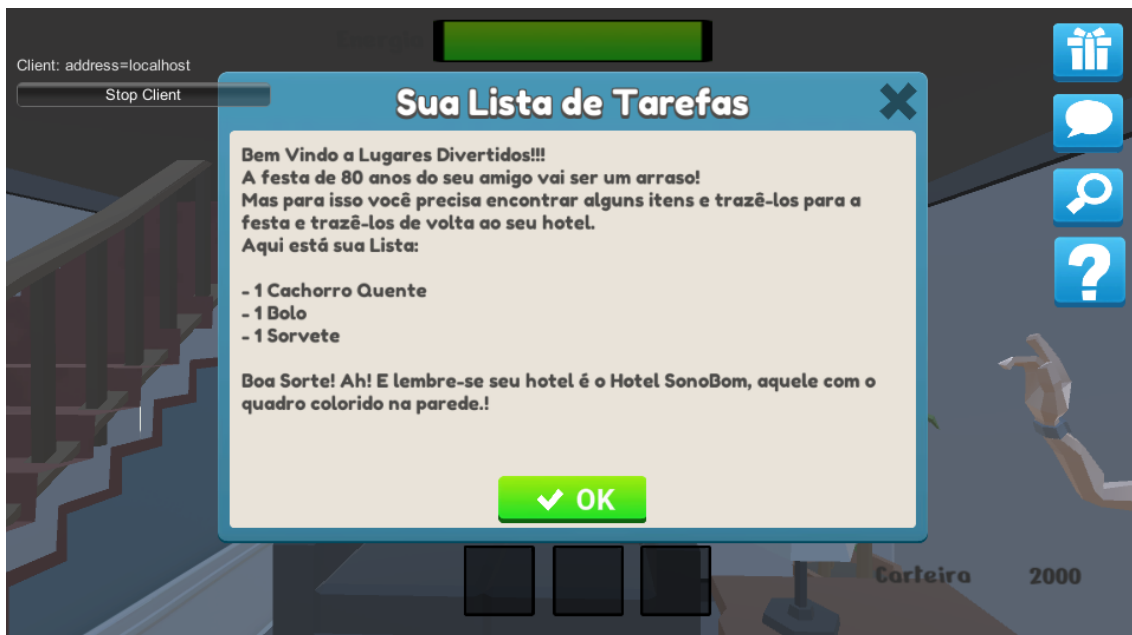


Fig. 9 Lista de Tarefas com a explicação do objetivo para o jogador.

Todos os itens na lista do jogador, assim como o hotel associado a ele, são decididos pelo psicólogo na fase de preparação enquanto o jogador está escolhendo seu personagem. Essas informações são salvas num objeto comum a todos os jogadores que recebe: os hotéis, os itens das listas, se o jogador tem que calcular as compras e quantos dígitos tem a senha do cartão. No momento que o jogador é instanciado ele lê as opções associadas a ele desse objeto e guarda em si próprio localmente essas informações.

7.2 Os Controles

Os controles do jogo por parte do jogador envolvem a movimentação e a interação. A movimentação é feita através do teclado. O jogador se move avançando ou retrocedendo em um eixo e pode-se virar a câmera para a esquerda ou para a direita. O controle desses movimentos pode ser feito pelas setas, com o avanço e retrocesso sendo ditados pela seta superior e pela seta inferior respectivamente e a movimentação da câmera para os lados é controlado pelas setas esquerda e direita.

Alternativamente, o jogador também pode usar um padrão bem conhecido no mundo dos jogos o WASD, onde as teclas de letras passam a corresponder a mesma orientação cardinal que as teclas de seta, com o 'W' como seta superior, o 'S' como seta inferior, o 'A' como seta esquerda e o 'D' como seta direita. Essa posição é mais confortável ergonomicamente para quem já tem o costume de jogar jogos de computador que usam o teclado dessa maneira, mas para pessoas que não estão

acostumadas é recomendado o uso das setas por ser visualmente é mais fácil de entender intuitivamente a que posição corresponde cada ação.

A interação é feita principalmente através do mouse com o clique em objetos do jogo ou com elementos da interface de usuário. Quase toda ação que o jogador fizer, seja selecionar o personagem no Lobby (Fig. 10), escolher o alvo certo no tutorial de reconhecimento, interagir com as barracas que recarregam a barra de energia, interagir com outros jogadores e interagir com os itens que ele precisa coletar, todos usam o clique do mouse para selecionar o alvo que você deseja interagir. Para as interações com a interface de usuário não há restrições, exceto restrições de tempo em alguns casos, mas para as interações com objetos do jogo é necessário estar dentro de uma distância mínima do objeto que é calculada pela diferença de distância do objeto que representa o jogador e objeto alvo.



Fig. 10 Tela de seleção de personagem do jogador.

7.3 Barra de energia

Um dos conceitos importantes do jogo é a barra de energia: ela começa cheia e é consumida ao caminhar no modo de jogo normal, e é mostrada ao jogador pela barra de energia que fica no topo da tela e vai lentamente diminuindo conforme o valor dela vai decrescendo.

A energia pode ser recarregada de duas maneiras, a primeira é em pontos de refresco que podem ser encontrados espalhados pelo mapa. Para comprar um refresco basta interagir com os carrinhos de suco (Fig. 11), cada vez que a ação de refresco for tomada ela irá custar ao jogador 100 unidades de dinheiro da sua carteira,

sendo assim é um uso limitado pois o jogador também precisa desse dinheiro para comprar os itens necessários para concluir suas tarefas. Cada uso do refresco recupera 30% do total da barra de energia, não podendo recuperar mais do que o máximo.



Fig. 11 Ao topo a barra de energia parcialmente esgotada e a frente o carrinho de suco.

Outro modo de recuperar energia é interagindo com outros jogadores. Ao encontrar outro jogador é possível interagir como ele clicando no outro, essa ação é unilateral e individual do jogador. A interação com outro jogador recupera a barra de energia em 50% e assim como o refresco não pode recuperar mais do que o máximo. Não há limites para a interação com outros jogadores para recuperar energia, o que incentiva a cooperação e a socialização entre os jogadores.

Caso o jogador fique completamente sem energia ele não poderá se mover até que recupere a energia e pela natureza estática dos pontos de refresco se faz necessário que um outro jogador vá até o colega imóvel e se aproxime para que ele interaja recuperando sua energia.

Dentre as métricas coletadas sobre a barra de energia temos a quantidade de vezes que o jogador tentou se refrescar, a quantidade de vezes que ele conseguiu se refrescar, o valor que a barra de energia tinha no momento que ele se refrescou, a quantidade de vezes que a barra de energia acabou e a quantidade de vezes que interagiu com outro jogador.

7.4 Carteira

Uma mudança menor foi corrigir e manter consistente o valor da carteira do jogador. Anteriormente, uma dificuldade do projeto era como e onde manter esse valor para que ele ficasse consistente entre o jogador e o servidor e fosse exibido na tela do jogador corretamente, além de manter o quanto foi gasto e o dinheiro que sobrou sendo observado pelo Game Analytics. Como as interações entre o jogador e o dinheiro são totalmente independentes do resto dos jogadores atualmente, o controle da carteira fica a cargo do objeto do Player, atualizando os valores no momento da transação para si e para o servidor por meio de comandos. Nota-se que a informação relevante para o Servidor é apenas a questão dos Analytics.

7.5 Seta

No projeto original havia uma funcionalidade de procurar o jogador mais próximo, o que seria feito ao clicar no botão associado a essa função que iria gerar uma seta por alguns segundos que aponta para o jogador mais próximo. Essa função estava um pouco incompleta e com alguns problemas como aparecer para outros jogadores, além do que acionou a seta, e alterar a câmera do jogador para uma versão em terceira pessoa que focava na seta e era desorientante. Atualmente a seta é gerada na frente do jogador e fica fixa na frente da câmera independente de onde o jogador estiver olhando (Fig. 12), ela gira em torno de seu próprio eixo e aponta para um jogador que esteja dentro da distância máxima dela, além de não mais aparecer para outros jogadores que não o que ativou a seta (Fig 13). O efeito da seta dura alguns segundos e após esse período a ação fica indisponível de ser acionada novamente por alguns segundos.



Fig. 12 Figura com a seta funcionando apontando para outro jogador a distância



Fig. 13 Telas de dois jogadores. A tela da esquerda é o jogador visto ao longe na direita e sua seta não é visível para o jogador da direita.

7.6 Do Lado do Psicólogo

O psicólogo controla a instância do Servidor, o Host, sendo ele quem controla as configurações do jogo e o fluxo do jogo, podendo influenciar também o jogo uma vez que ele está em progresso.

A primeira tela que o psicólogo tem acesso após começar a hospedar uma partida como “Host” é a tela de configuração de servidor do Lobby (Fig. 14). Aqui é possível definir as opções para cada jogador durante o jogo, essas incluem em que hotel ele irá começar, quais itens ele terá de adquirir no jogo, se ele deverá calcular ou não suas compras e a quantidade de dígitos que sua senha terá. É possível marcar

uma opção para que algum modo opcional seja jogado antes do principal, esses modos são o de tutorial de movimentação e o tutorial de reconhecimento. Também é possível salvar as configurações atuais dos jogadores em pre-sets, que guardam de forma persistente entre sessões do jogo essas configurações e o psicólogo pode carregar esses pre-sets que ele tenha salvo para alterar as configurações. e enviar ao jogador um questionário que ele deve preencher que será salvo no servidor.



Fig. 14 Tela de configurações no Lobby do jogo.






7.7 Questionário

Os questionários(Fig. 15) contém perguntas que variam em escalas de 1 a 5 e atualmente são compostos por 15 perguntas, porém mais perguntas podem ser adicionadas. Até 5 perguntas ficam visíveis por página e para prosseguir para a próxima página é preciso ter respondido todas as perguntas. Ao completar a última página, o jogo utiliza um **comando** para enviar as respostas para o servidor que então salva para o psicólogo e fica marcado para o psicólogo que aquele jogador terminou o

questionário.

Client: address=localhost
Stop Client

Por Favor Responda Pontuando de 1 a 5.

	Eu sempre lido com tecnologia e procuro os novos lançamentos e tendências.	<input type="text"/>
	Eu gosto de ter os ultimos lançamentos de computadores e consoles.	<input type="text"/>
	Eu estou disposto a pagar qualquer preço por um jogo.	<input type="text"/>
	Eu prefiro jogos de ação ou violentos.	<input type="text"/>
	Eu prefiro jogos com complexidade e profundidade	<input type="text"/>

Próximo →

Fig. 15 Questionário que o jogador responde a pedido do psicólogo.

7.8 Tela do psicólogo no jogo

O painel de controle do psicólogo (Fig. 16) é por onde ele vê e interage com o jogo durante o decorrer da partida. Nesse modo é possível se deslocar pelo mapa com o mesmo esquema de controle que o jogador usa para andar, mas ao invés de controlar um personagem o psicólogo observa a cidade de cima podendo se aproximar ou se afastar através dos botões de câmera que ficam no lado esquerdo de sua tela.

Uma interação importante do psicólogo com o jogador no decorrer do jogo é que é possível mandar mensagens para jogadores específicos. A intenção principal dessa função é poder enviar dicas ao jogador, mas não existe uma restrição de que tipo de mensagem de texto pode ser enviada. A mensagem aparece por alguns segundos no topo da tela do jogador e depois some.

Foi adicionada uma funcionalidade ao painel de controle do psicólogo durante as cenas de tutorial, permitindo que o psicólogo não precise encerrar a sessão e começar de novo para transitar entre as cenas de tutorial e do jogo normal. Mesmo transitando entre essas cenas, os dados a serem salvos pelo sistema de *Game Analytics* persistem entre cenas e não são zerados ou sobrescritos ao entrar em uma nova cena. Isso permite começar uma partida no tutorial de movimentação e quando mudar para a cena de jogo normal as métricas relevantes do tutorial ainda estarão guardadas pelo servidor para que possam ser salvas junto com as métricas coletadas na cena de jogo.

Além disso, também foi adicionado um botão para permitir que o psicólogo salve para seu computador as *Game Analytics* de todos os jogadores.



Fig. 16 Tela do servidor durante o jogo principal com o painel de controles.

7.9 NPCs

No projeto original do Funny Places, os NPCs (Non player character), que são todos os personagens que não são jogadores, eram manualmente colocados no mapa com sua forma e animação já definidas e não tinham a capacidade de se movimentar, eram essencialmente objetos estáticos que não mudavam. Agora com a criação de um prefab mais genérico de NPC e com um controlador novo e mais robusto para as animações, os NPCs agora podem ser instanciados na cena e alterados livremente pelo editor, inclusive sendo possível mudar a animação atual no meio da execução. Apesar de viável, não há atualmente uma ferramenta para o psicólogo poder editar o NPC enquanto o jogo está em execução, mas o sistema para permitir tal função existe.

O controlador de animações (Fig. 17) dos NPCs tem por padrão a animação idle quando ele é instanciado, mas através das variáveis associadas a ele que por padrão tem como nome "is", precedendo o nome do estado associado, é possível alternar entre as animações existentes. Para os estados onde o NPC se encontra parado a transição de animações precisa passar pelo idle, o que tem como função diminuir o número de erros e evitar transições estranhas. Dessa forma, uma animação precisa ser encerrada, retornando ao idle, para que ela possa ser trocada por outra animação. As animações de caminhada, por outro lado, para manter um fluxo mais suave entre andar para frente e para trás, se necessário podem alternar entre si sem

precisar retornar ao idle, só retornando quando o NPC não mais estiver andando nem para frente nem para trás.

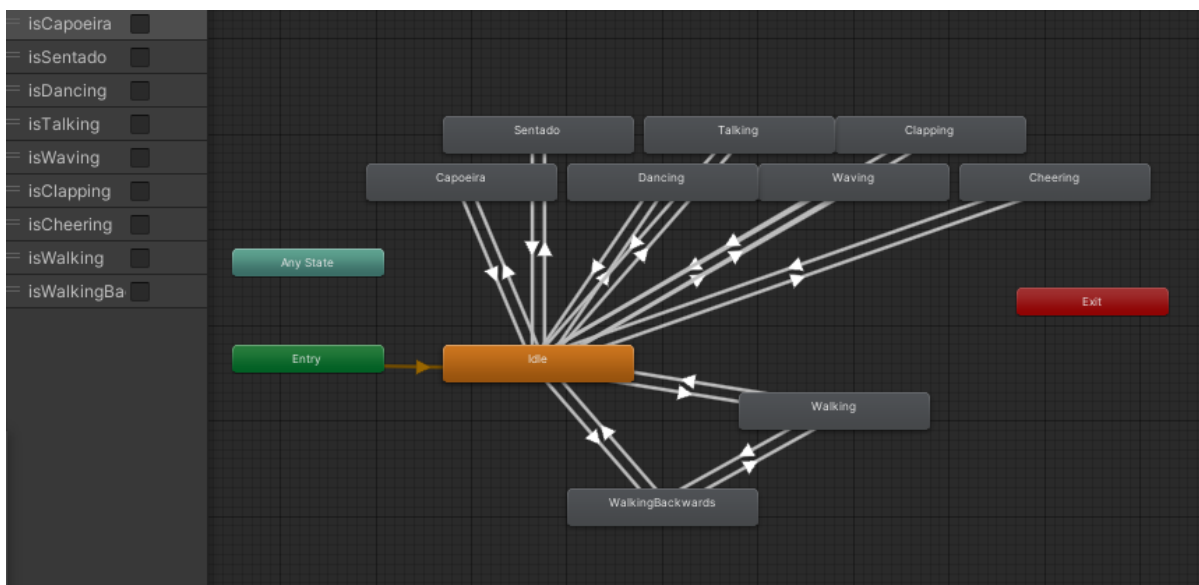


Fig. 17 Controlador de animações genérico dos NPCs

Nessa versão do jogo há dois tipos de NPC diferentes dos NPCs estáticos normais. Há a possibilidade de ter NPCs que caminham entre dois pontos, podendo aumentar esse número se desejado, e os NPCs que dão dicas são sinalizados por uma indicação amarela em suas cabeças (Fig. 18). Ao interagir com eles, clicando no NPC, eles dão uma dica sobre os itens para o jogador (Fig. 19). A dica se baseia na instância do NPC, e todo NPC tem um script NPC_Dicas que guarda dentro dele um valor de “dica”. Se o NPC tiver um dica associada a ele, essa dica deve ser escrita no Editor após criar uma instância do NPC de Dicas no mapa na variável de dica que é visível no Inspector, se não a dica não poderá ser carregada pelo jogador.

Vale ressaltar que o NPC de dicas precisa ter a tag “NPC_Helper” associada a ele, pois é assim que o jogador sabe que esse NPC é interativo. A tag permite que o jogador acione a função que lê a dica dele. Além disso, para diferenciar o NPC de dicas do resto dos NPCs ele tem um prefab próprio para facilitar, que inclui não só o NPC base onde se ajusta as dicas, mas também um objeto amarelo acima de sua cabeça. Esse objeto não tem propósito mecânico e serve apenas para sinalizar ao jogador que aquele NPC tem algo diferente do resto.



Fig. 18 NPC de Dicas no seu estado normal, com um indicador acima de sua cabeça para distingui-lo



Fig. 19 NPC de dica após a interação com a sua dica sendo exibida na tela do jogador

O script de movimento dos NPCs que se movem se baseia em ter dois objetos que não possuem modelo para agirem como objetivos para o NPC, que segue entre esses dois pontos. Apesar dos NPCs implementados usarem apenas dois pontos de movimento, do modo que o código foi feito é possível adicionar mais pontos para a

rota, no entanto seria necessário aumentar igualmente o número de objetos de objetivo relacionados ao NPC.

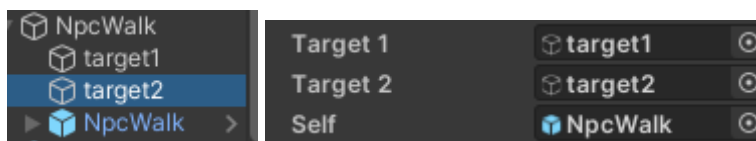


Fig. 20 Como é organizado e definido quais são os targets do NPC que caminha.

Como forma de organização os alvos ficam como parte de um objeto para o NPC (Fig. 20) enquanto o prefab do NPC em si também fica nesse agrupamento e recebe os targets e a si mesmo como variáveis. Isso ajuda com as possibilidades de expansão de rotas no futuro, podendo alterar ou duplicar o Script “NPC_Walking” para ter versões com mais targets.

7.10 Cenas de Movimento e Tutorial

Além da cena de gameplay normal há mais duas cenas para o jogador agora, a cena de tutorial de movimentação e a cena de tutorial de reconhecimento. Ambas as cenas podem ser acessadas durante a etapa de definir as configurações do jogo por parte do psicólogo, marcando se ao invés de ir para a cena de gameplay normal direto ele quer que vá para a cena de tutorial de movimentação ou a de reconhecimento. Quando essas opções são marcadas, o botão de iniciar o jogo ao invés de levar para o gameplay normal, leva para uma dessas cenas.

A cena de tutorial de movimentação é uma cópia da cena de gameplay normal, onde o jogador não interage com nada nem gasta energia e apenas passeia pelo mapa (Fig. 21). O objetivo é que o jogador que potencialmente não está acostumado com jogos, seus esquemas de controle e a sensação de movimento e câmera, possa ir se acostumando.

O modo essencialmente serve apenas para se acostumar, e cabe ao psicólogo decidir quando é um momento bom para encerrar o modo e seguir para uma das outras cenas. Entre as métricas coletadas pelo Analytics, uma delas é o quanto um jogador andou durante esse modo.

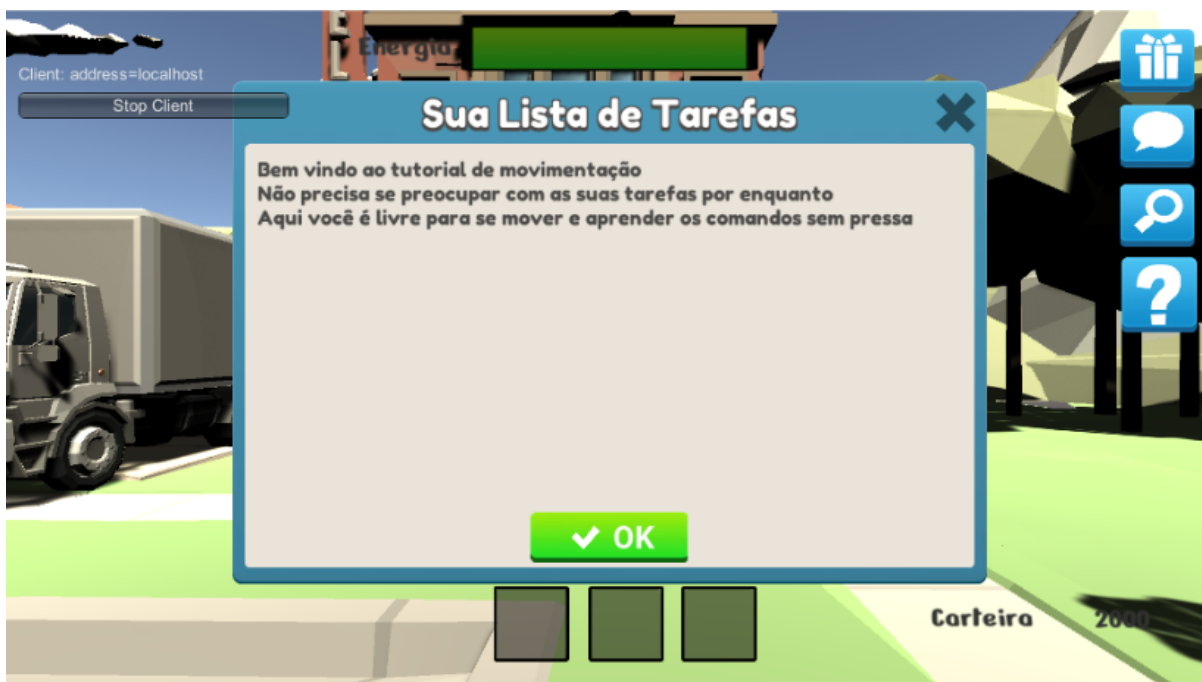


Fig. 21 Mensagem do tutorial de movimentação.

No tutorial de reconhecimento (Fig. 22) o jogador encontra alguns NPCs, entre estes 3 deles usam o mesmo visual do modelo dos avatares que foram escolhidos pelo jogador e pelos seus outros 2 colegas. O jogador deve então clicar para interagir com os NPCs que correspondem a seus outros colegas. A atividade tem como objetivo ajudar o jogador a diferenciar e reconhecer quem são os colegas que estarão com ele durante o jogo, dado que esse reconhecimento é importante por ser um jogo multijogador em que você pode interagir e cooperar com seus colegas. Entre as métricas coletadas pelo Analytics, uma delas é o tempo que o jogador precisou até acertar os dois colegas. Quando os jogadores tiverem encontrado seus colegas o psicólogo então pode avançar para a fase de Gameplay.

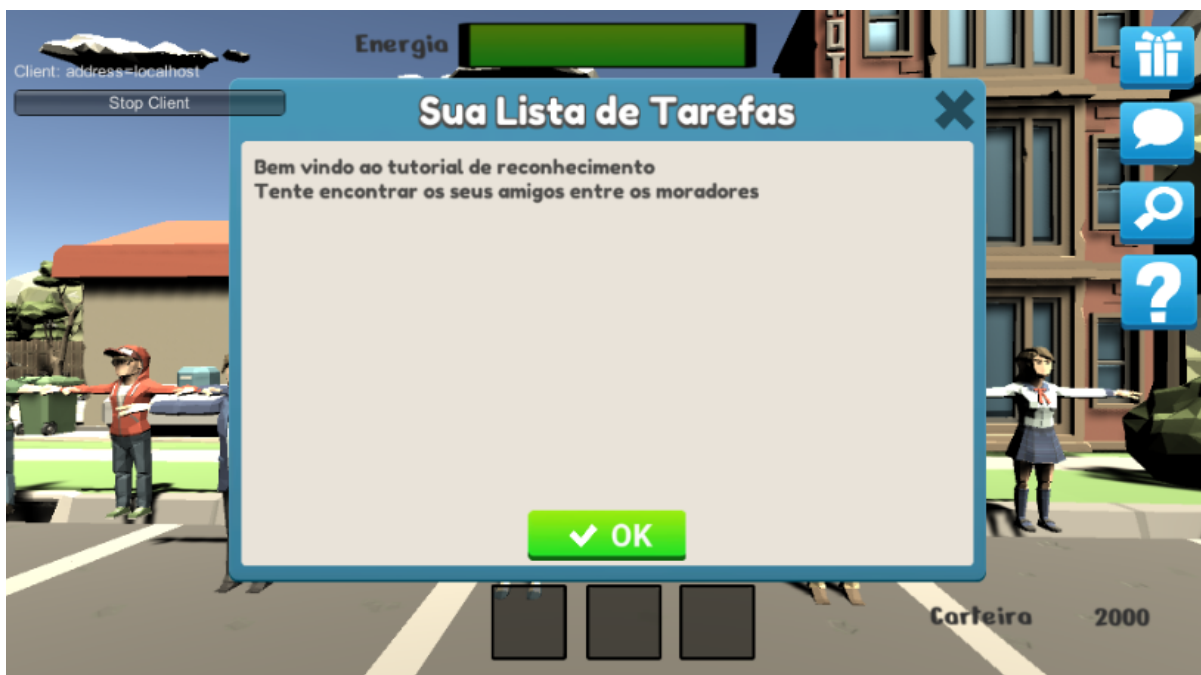


Fig. 22 Mensagem do tutorial de reconhecimento

Na cena do tutorial de reconhecimento (Fig. 23) estão dispostos lado a lado NPCs com as possibilidades de modelo de jogador e à frente deles os 3 NPCs que recebem a aparência dos jogadores atuais quando carregados. Atualmente a cena ainda não possui uma maneira de rearranjar as posições para poder misturar os verdadeiros e os falsos, mas a ideia final desse modo de jogo é para que não fique óbvio pela posição e que os jogadores de fato tenham que se lembrar de como era o modelo. Outro problema que pode vir a ocorrer é caso dois jogadores escolham o mesmo modelo, o que pode gerar confusões quando eles não estiverem em posições determinadas.



Fig. 23 Disposição dos NPCs no tutorial de reconhecimento provisório.

7.11 Final do Jogo

Ao final do jogo, ou seja, quando todos os jogadores tiverem completado suas tarefas, o primeiro jogador a terminar recebe como recompensa um voo pela cidade em um avião como uma comemoração por ter completado o jogo e ele recebe a opção de levar os outros colegas consigo (Fig. 24). A decisão do jogador de levar ou não os colegas é registrada como uma das métricas coletadas pelo jogo.

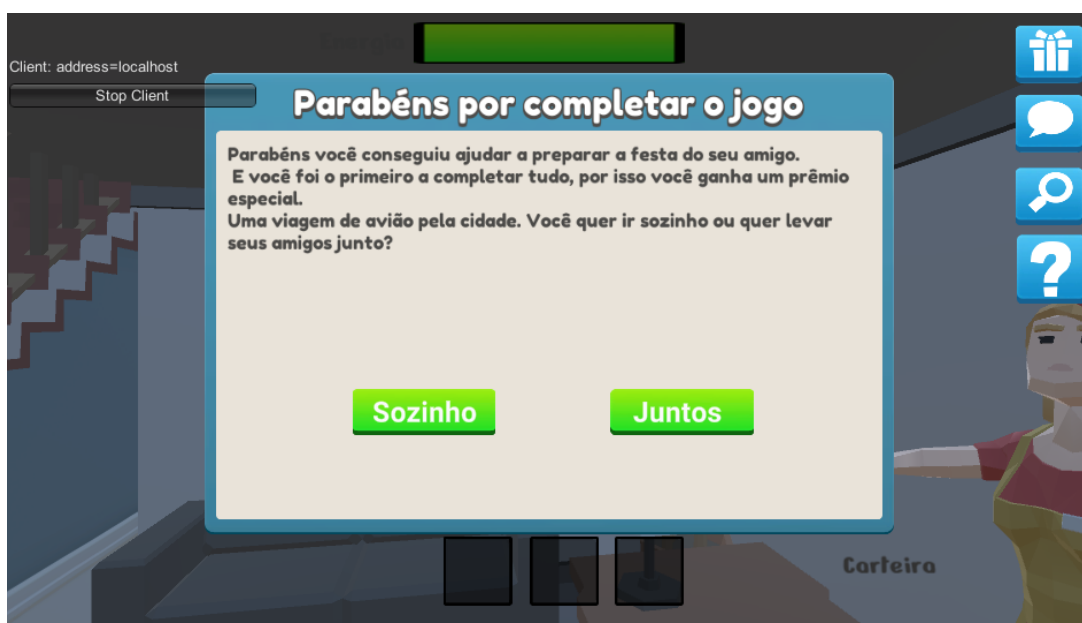


Fig. 24 Mensagem de fim de jogo para o primeiro a completar as tarefas.

O fim do jogo também é um ponto onde a sincronização entre clientes e servidor é importante, uma vez que não só é preciso registrar a ordem de quem completou as tarefas, mas que no momento que todos tiverem terminado apenas o primeiro que terminou receba a tela especial de fim de jogo, o resto recebe uma outra tela de final de jogo que não contém as opções, podendo apenas esperar pela decisão do primeiro (Fig. 25).

Além disso, com a decisão dele é preciso, caso ele vá sozinho, permitir a viagem de avião apenas para ele e caso ele escolha ir com os colegas é preciso ainda transmitir a decisão de volta ao servidor e informar aos outros clientes que eles também vão fazer esse passeio. O processo todo de marcar quem terminou, a ordem e a decisão faz uso de uma série de comandos que fazem requisições do servidor nos usuários usando **TargetRpc**, que fazem essencialmente o mesmo que um **ClientRpc**, só que direcionado a alvos específicos, no caso para cada jogador e não só quem fez o pedido.

Na questão técnica o passeio de avião é definido de uma forma similar, mas mais robusta do comportamento de movimentação de um NPC, o avião tem uma série de múltiplos pontos de alvo para o qual ele se direciona e de ponto em ponto ele irá fazer sua trajetória pelo céu, por fim retornando ao ponto de partida. Durante esse passeio a câmera do(s) jogador(es) que estiverem no passeio será substituída por uma câmera especial do avião (Fig. 26) e perderão temporariamente o controle de interação e movimento.

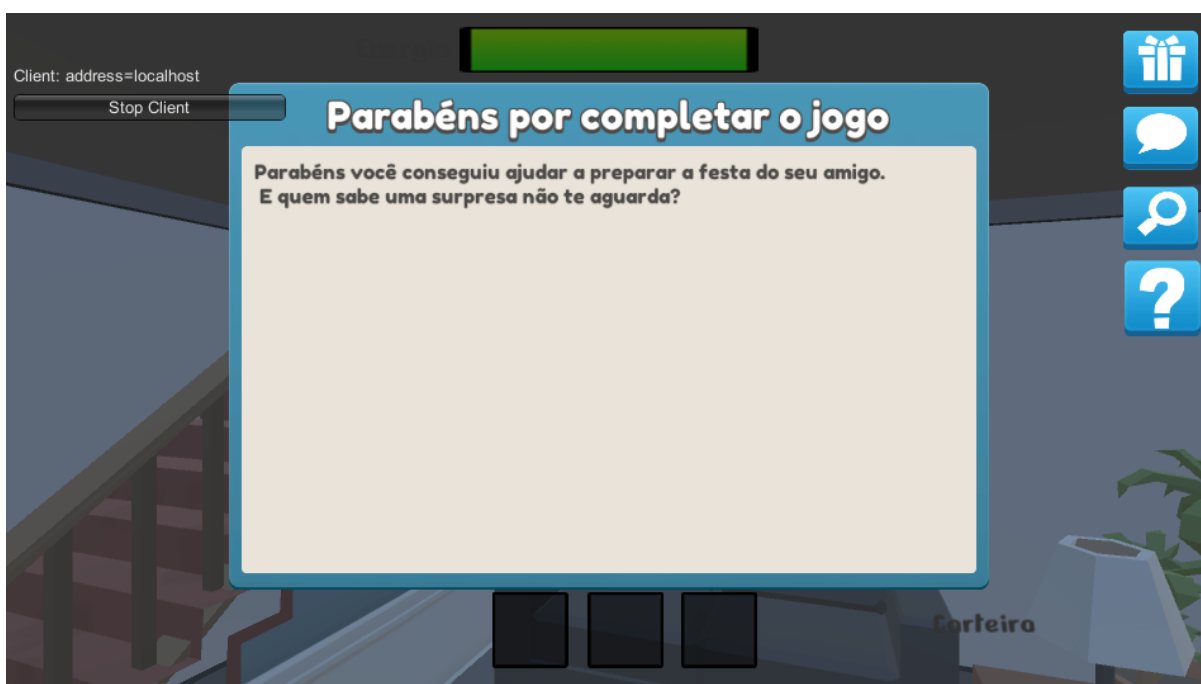


Fig. 25 Mensagem de fim de jogo para os outros jogadores.



Fig. 26 Passeio de Avião

7.12 Manutenção

Vale a pena ressaltar que como a base do jogo não está totalmente completa alguns pontos podem trazer dificuldade e confusão na manutenção do código. Há algumas funções que estão implementadas como *placeholders*, especialmente na área de interação com itens e inventário.

Há alguns arquivos que não estão sendo utilizados no jogo atualmente por serem alguns experimentos e protótipos que foram deixados de lado, mas que estão lá para serem estendidos futuramente.

Os nomes dos *scripts* em sua maioria explica a função deles, mas há certa redundância com relação a alguns mais antigos.

7.13 Controle de Versão

O controle de versão do projeto ao longo do desenvolvimento não foi muito forte. Perto do fim do projeto surgiu a ideia de fazer um *Fork* do repositório do Github original do Funny Places, mas muitas mudanças foram aplicadas de uma vez só e bem no final.

Futuramente a ideia é juntar esses dois repositórios e fazer um único repositório para que trabalhos futuros tenham um lugar para acessar os trabalhos antigos e poder guardar lá para também ser expandido no futuro.

8 Considerações finais e Trabalhos futuros

A proposta deste trabalho teve em mente desde o início que é a continuação de um projeto de jogo e que também possivelmente não seria o fim dele, mas sim um ponto ao longo do desenvolvimento. A parte do projeto que foi o foco desse trabalho foi principalmente a coleta das analítica de jogo e do salvamento dessas analítica e do questionário e outras informações que o psicólogo necessita para fazer uma avaliação dos dados.

O motivo desse foco é que mesmo se todo o jogo estivesse completo, funcional e operante, mas não fosse possível coletar e guardar essas informações o psicólogo não teria como aproveitar a maior força do jogo enquanto ferramenta para ele. É através da coleta desses dados que o psicólogo pode traçar padrões, ver o comportamento dos jogadores, entender como ele opera, fazer relações entre certas atividades e as métricas do jogo. Portanto para que nas futuras implementações essas ferramentas já estejam disponíveis e que seja possível fazer proveito das ações e dados do jogo, esse foi o foco escolhido.

Além dessa parte teve algumas tentativas de melhorar alguns aspectos do jogo que envolviam essa parte da relação servidor-cliente, além de já preparar a base necessária para outras implementações. O tutorial de reconhecimento por exemplo ainda precisa de mais alguns passos para que ele fique apropriado com a proposta original do jogo, no entanto a base para a transição da cena, a permanência das métricas, o objeto necessário para manter as informações do jogador nesse processo de transição, já são alguns objetivos muito úteis para a implementação final dessa cena.

Outro ponto de atuação do projeto foi em garantir e corrigir algumas das funções que já estavam ou parcialmente implementadas ou já fortemente conceituadas, entre essas os NPCs, a barra de energia, a seta de jogador mais próximo. Uma implementação que infelizmente não pode ser feita, mas que já estava fora do escopo desse projeto era a parte da coleta dos itens. Apesar de ser um ponto importante para o objetivo principal do jogo pela parte do jogador, a implementação é um tanto mais complicada do que parece, além de envolver uma interação entre objeto e jogador que precisa ser sincronizada entre servidor e cliente, é necessário também manter os itens no inventário, garantir que é o item certo, balancear onde os itens surgem no jogo, balancear o preço dos itens, tratar o caso de cálculo do valor pelo jogador, gerar a senha para o cartão, repetir a senha do cartão caso o jogador erre, entre outras coisas.

Novamente é importante lembrar que assim como esse trabalho herdou o projeto de um antecessor ele também será herdado por alguém que continue o

projeto. Eu espero que se caso eu não retorne a esse projeto para uma pós-graduação ou algum outro trabalho do gênero, que alguém dê continuidade a esse projeto, ele se tornou muito querido para mim e eu vejo um enorme valor nele. Ele tem um potencial enorme para ser uma ferramenta que ajude os psicólogos a entender melhor nossas mentes e comportamentos e as mazelas que nos afligem.

O projeto foi uma grande oportunidade de aprendizado especialmente nas relações entre cliente e servidor no ambiente de jogos, o Mirror apesar de não ser a solução mais atual é uma boa base para aprender sobre essas interações e sobre as dificuldades de manter o jogo sincronizado entre todas as partes. Trabalhar com a analítica de jogos também foi bem interessante, me fazendo reconsiderar o potencial que elas têm para ajudar os desenvolvedores e outras partes interessadas a entender o comportamento do jogador e como melhorar a experiência para ele.

Do ponto de vista de desenvolvimento de jogos também foi uma nova oportunidade de aprendizado, não havia até esse momento trabalhado com jogos 3D, nem multiplayer online. Já havia feito alguns jogos sozinho, mas nada tão complexo antes. Quanto a engine do Unity, já tinha tido algumas experiências prévias, mas não como programador, inclusive para me familiarizar mais cursei uma matéria no período de 2021.2 que envolvia desenvolvimento de jogos onde a engine foi o Unity, minha experiência como programador e no geral maior era com o GameMaker Studio 2, todavia não foi ruim trabalhar com o Unity após aprender algumas de suas tendências e peculiaridades.

Pensando no que eu teria feito diferente caso tivesse o conhecimento de hoje no início do projeto, primeiramente teria tentado acumular mais experiência com o Unity 3D antes de começar o projeto, além disso numa questão mais prática faltou visão de perceber que para o visualizador de resultados apenas o conhecimento que eu tinha da biblioteca pandas não era o suficiente, duas coisas faltaram, uma foi entender melhor que tipo de visualizações seriam úteis ao psicólogo e a outra foi um trabalho de interface visual melhor. Ao desconsiderar a parte visual o programa, apesar de funcional e existente não se apresenta de forma muito amigável para um psicólogo que não tenha um mínimo conhecimento de programação.

Para trabalhos futuros o primeiro e mais urgente do meu ponto de vista agora é completar as funções principais do jogo, ainda há algumas coisas especialmente no loop principal de jogo, a coleta e entrega dos itens, que precisa ser implementado ou refinado. As bases para isso existem, porém precisam ser expandidas e conectadas, além disso seria interessante aumentar a interação do psicólogo com o jogo, atualmente a maior capacidade dele durante o jogo em si é de dar dicas aos jogadores.

No projeto original[1] havia a ideia de permitir ao psicólogo que criasse imprevistos em tempo real de modo que atrapalhasse e causasse certas dificuldades para o jogador com o intuito de fazê-lo repensar e mudar de curso de ação, seja fazendo com que um certo item não esteja disponível na loja ou mudar certas coisas de lugar. Essa flexibilização da ação do psicólogo é uma ótima implementação futura quando o jogo já estiver mais estável para que essas interações sejam significativas.

Tendo uma versão com a base do jogo completa, incluindo todos os modos já idealizados operantes em sua forma total, o jogo tem vários lugares para onde pode crescer. Mais cenários, mais tarefas, mais interações do psicólogo no meio da partida, além de ser mais conteúdo que ajuda com a possibilidade de jogar novamente o jogo, aumenta a possibilidade de misturar diferentes circunstâncias e criar cenários novos e inesperados que ajudam a trabalhar as funções cognitivas, o pensamento flexível e a adaptação a circunstâncias inesperadas.

Um caminho possível também é de expandir o porte do jogo, se necessário com um servidor dedicado pelo Mirror que tem um custo relativamente baixo pelo serviço (60 dólares), tendo mais jogadores e mais dados coletados seria possível expandir a estrutura atual de coleta de dados e com uma base maior de dados as informações coletadas poderiam passar a representar estatísticas reais e ajudar a encontrar padrões em um grupo bem maior do que atualmente.

Há muitas outras possibilidades e inclusive seria interessante que após a parte central do jogo estar fechada haja nova cooperação com o departamento de Psicologia, que ajudou a criar a base do projeto original [1], para decidir quais dessas direções ou quais outras direções o projeto poderia tomar para ser o mais útil possível e possa fazer uma diferença real no mundo.

Referências

- [1] Tassara, B.P. **Um Jogo Multijogador Online para Identificação de Funções Executivas em Idosos**. Dissertação de Mestrado, Dept. de Informática, PUC-Rio, 2021.
- [2] Valladares-Rodriguez S, Pérez-Rodriguez R, Fernandez-Iglesias JM, Anido-Rifón LE, Facal D, Rivas-Costa C. **Learning to Detect Cognitive Impairment through Digital Games and Machine Learning Techniques. Methods of Information in Medicine**. Setembro de 2018, pp. 57(4):197-207.
- [3] Krawczyk DC, Han K, Martinez D, Rakic J, Kmiecik MJ, Chang Z, Nguyen L, Lundie M, Cole RC, Nagele M, Didehbani N. **Executive Function Training in chronic traumatic brain injury patients: study protocol**. Trials. 20, 2019, Vol. 435.

- [4] A, Diamond. **Executive functions**. Annual Review of Psychology 64. 2013, pp. 135-168.
- [5] Mograbi, D. C, Faria, C. A, Fichman H. C, Paradela, E. M. P, Lourenço, R. A. **Relationship between activities of daily living and cognitive ability in a sample of older adults with heterogeneous educational level** . Annals of Indian Academy of Neurology. Janeiro de 2014, pp. 17(1):71-76.
- [6] Fichman H. C. Neuropsicologia. Notas de Aula, Departamento de Psicologia PUC-Rio, 2020.
- [7] Mirror, Remote Actions. In Mirror Networking User Manual, section Guides. Disponível em <https://mirror-networking.gitbook.io/docs/guides/communications/remote-actions> Acesso em 11/07/2022.
- [8] Code Monkey, **What is JSON? (Unity Tutorial for Beginners)**. Disponível em <https://youtu.be/4oRVMCRCvN0> Acesso em 08/07/2022
- [9] Pandas development team. **pandas documentation user guide**. Disponível em https://pandas.pydata.org/docs/user_guide/index.html#user-guide Acesso em 10/07/2022
- [10] Abramychyev, Alexey, Riley, S, Juhl, E. **Mirror Network Guides** Disponível em <https://mirror-networking.gitbook.io/docs/guides> Acesso em 21/01/2022
- [11] Unity Technologies. Documentação do **Text Mesh Pro**. Disponível em <https://docs.unity3d.com/Packages/com.unity.textmeshpro@2.0/api/TMPro.html> Acesso em 13/05/2022
- [12] Unity Technologies. **Unity Scripting API Reference** Disponível em <https://docs.unity3d.com/2019.4/Documentation/ScriptReference/index.html> Acesso em 20/01/2022