Desenvolvimento de uma Releitura do Jogo *Car Town*

Juan F. C. Bailke, André C. da Silva

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas
Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP)
Campus Hortolândia

juanbailke1@gmail.com, andre.constantino@ifsp.edu.br

Abstract*. There are different genres of games, such as sports, fighting, adventure, strategy, RPG, among others. An example of a strategy game played in browsers is CarTown, in which the player manages a car repair shop. The purpose of this article is to reinterpret this game, creating a 2.5D prototype with the Unity platform. Based on the knowledge of the researcher/player, the GDD (Game Design Document) was created, recording information that was used for the development of the prototype, such as genre and technical specifications in addition to the idea of gameplay, creating the essential scripts for the game mechanics in conjunction with the use of free assets avaliable on the Internet.*

Resumo. *Existem diversos gêneros de jogos, como de esportes, luta, aventura, estratégia, RPG, entre outros. Um exemplo de jogo de estratégia executado em navegadores é o CarTown, no qual o jogador gerencia uma oficina mecânica para carros. O propósito trabalhado neste artigo é referente a releitura desse jogo, criando um protótipo em 2.5D com a plataforma Unity. A partir do conhecimento do pesquisador/jogador, elaborou-se o GDD (Game Design Document) registrando informações que foram utilizadas para o desenvolvimento do protótipo, como gênero e especificações técnicas além da ideia de gameplay, criando os scripts essenciais para a mecânica do jogo em conjunto com o uso de assets gratuitos disponíveis na Internet.*

1. Introdução

De acordo com a Pesquisa Game Brasil de 2021 a plataforma preferida dos brasileiros para jogar são os *Smartphones*. Essa crescente vem acompanhada com a maior facilidade que as classes econômicas mais baixas possuem de adquirir um aparelho que atenda a requisitos mínimos de diversos jogos. Além de ser a plataforma mais utilizada para jogar em todos os dias da semana, representando 40,8% dos entrevistados, apenas 7,7% afirmaram não jogar atualmente com celulares.

Existem diversos gêneros de jogos, como de esportes, luta, aventura, estratégia, RPG (*Role-play Game*), entre outros (SILVA, 2021). Um exemplo de jogo de estratégia executado em navegadores é o CarTown, da Cie Games, popular em 2011. Contudo o jogo não está mais disponível no mercado desde 2014, apesar de petições online pedindo o retorno do jogo com mais de 7.529 assinaturas (L. Sergio, 2017). Não existem muitas informações sobre o porquê do jogo não estar mais ativo no mercado, apenas uma nota disponibilizada pelos desenvolvedores indicando seu encerramento (CONTESINI, 2014).

Inspirando-se no jogo *Car Town*, o objetivo deste projeto é construir uma versão de jogo digital de estratégia em 2.5D para dispositivos móveis utilizando a plataforma de desenvolvimento de jogos Unity, considerando a temática de uma oficina mecânica e serviços para carros. Pretende-se desenvolver os *scripts* necessários para o funcionamento da mecânica do jogo, a utilização de *assets* gratuitos para personagens, imagens, músicas e outros elementos e a criação de novos *assets* para complementar o *design* do jogo. Além do jogo, será desenvolvido um GDD (*Game Design Document*), um documento contendo todas as informações relevantes do *design* de um jogo: temática, mecânicas, plataformas, inimigos, níveis, entre outros, a fim de auxiliar o

desenvolvimento do produto final. Será utilizado como inspiração o jogo *Car Town* da *Cie Games*, usando a análise deste jogo como ponto de partida.

Na Seção 2 é tratada a fundamentação teórica, onde constam definições sobre o que são jogos digitais, como são elaborados os processos e gerenciamento de recursos utilizados para o desenvolvimento dos jogos, juntamente sobre como são feitas avaliações de seu processo. Na Seção 3, será exemplificado o jogo utilizado como base no projeto, com as informações e dados que eram utilizados na época de seu funcionamento e que serviram como inspiração para a criação deste protótipo. Além de outros jogos que possuem funcionalidades semelhantes e que ainda estão ativos até o dia de hoje. Também serão abordados outros jogos que possuem funcionalidades semelhantes e que ainda estão ativos. Na Seção 4 será apresentada a metodologia utilizada, a utilização do *Game Design Document* para trilhagem do desenvolvimento do protótipo, a jogabilidade e eventos. A Seção 5 apresenta o desenvolvimento do projeto. Finalizando com a conclusão do trabalho e indicando melhorias que podem ser efetuadas.

2. Fundamentação Teórica

Esta Seção discute sobre definições de jogos digitais e seu desenvolvimento.

2.1. Jogos Digitais

Jogos são compostos de regras pré-definidas que jogadores voluntários optam por jogar em busca de um objetivo simbólico ou psicológico, sendo atividades não produtivas de efeitos negociados no mundo real mas utilizados em um mundo abstrato, onde possui resultado incerto. (SALEN & ZIMMERMANN, 2012; CAILLOIS, 2017).

Segundo Xexéo (2013), jogos estimulam a socialização, sendo jogos físicos que necessitam da interação de mais de uma pessoa, ou até contra o computador, que ao interagir em comunidades com mais jogadores de mesmo interesse, permitam que os jogadores descubram segredos, soluções para desafios, ou dicas compartilhadas entre os jogadores.

De acordo com Petry (2016), o jogo digital permite ao jogador a vivência de situações que não seriam possíveis ou autorizáveis no mundo real e social. Ao mesmo tempo em que o jogo digital permite a experienciação de situações protegidas e encapsuladas, ele também oferece, do ponto de vista do jogador, a manifestação de eventos não controlados pelo jogador, que, depois de sua eclosão, tem de lidar com eles.

2.2. Desenvolvimento de Jogos Digitais

A diferença entre *softwares* tradicionais e um jogo digital é que este possui sua qualidade medida por fatores que envolvem o prazer de quem está tendo a experiência com o jogo, não sendo voltado para a solução de um problema (ARAÚJO, 2019).

Casarini e Petry (2019) propõe como proposta de *framework*, a criação de um *Level Design Document*, ao qual se faz necessário uma pesquisa para um repertório de referências com semelhanças ao que se está buscando e como pode ser aplicada, aprendida ou modificada. Então ao iniciar o “partido do projeto” (definição para os conceitos a serem seguidos até a finalização do *level*, seguindo como referência para as decisões a serem seguidas), é necessário definir os pilares do *level design*, a visão desse *level design*, câmera, avatar e escala, o fluxograma de áreas, a prototipagem em papel e na *engine*, a implementação de arte e por fim o polimento.

O Documento de Projeto do Jogo (comumente chamado de GDD, *Game Design Document*), descreve a mecânica do jogo e outros elementos pertinentes norteando, assim, toda a equipe. Schuytema (2008) define as seções que este documento deve conter, entre eles, uma visão geral, o contexto do jogo e sua história, principais jogadores e personagens, estruturas das fases, dos cenários e dos objetos, conflitos e soluções, bem como o fluxo do jogo.

3. Trabalhos Correlatos

Nesta Seção apresentaremos o jogo CarTown, que foi inspiração na realização deste trabalho, e os jogos Clash of Clans e Hay Day.

3.1. O jogo CarTown

No jogo CarTown (RIBEIRO, 2013), o jogador gerencia uma oficina mecânica e de serviços para automóveis (Figura 1), podendo expandir a sua oficina e serviços oferecidos (Figura 2) à medida que ganha dinheiro realizando os serviços. Na Figura 1, é possível visualizar um carro verde, na qual um funcionário realiza o serviço, e um carro vermelho, de propriedade do jogador. Nota-se também o nível do jogador (em formato de velocímetro no canto superior esquerdo), total de moedas coletadas, um mostrador de nível de combustível (que significa a quantidade de vezes que o jogador pode participar de uma corrida: um minigame para conquistar mais moedas especiais). O jogo possuía interações online com amigos do *Facebook*, apresentados na parte inferior da tela.

Figura 1. Tela principal do jogo CarTown com a oficina em seu início de funcionamento.

Na Figura 2 é mostrada uma oficina maior, ampliada à medida que o jogador compra novos itens para sua oficina e, então, pode oferecer mais serviços. Observa-se também que o jogador ampliou sua coleção de carros.

Figura 2. Tela principal do jogo CarTown com a oficina ampliada com as conquistas do jogador.

Na Figura 3 é exibido a tela do jogo onde o jogador pode adquirir novos carros para sua coleção.

Figura 3. Tela para aquisição de novos carros no jogo CarTown.

3.2. O jogo Clash of Clans

O jogo Clash of Clans (Figura 4), é um jogo desenvolvido pela Supercell, onde possui o mesmo estilo de câmera e compõe de avanços progressivos conforme o tempo de

produção das funcionalidades do local gerido pelo jogador, além de uso da estratégia para gerir qual função deve ser concluída para maximizar o uso do tempo.

Figura 4. Tela de uma vila no Clash Of Clans.

3.2. O jogo Hay Day

O jogo Hay Day (SOUZA, 2023), desenvolvido também pela Supercell, é um jogo de gerenciamento de fazenda (Figura 5), mas possui interações com a necessidade da ação do jogador que se assemelha ao utilizado nesse jogo. E com o avanço de níveis são liberadas novas instalações, objetos decorativos e outros objetos na loja, com o dinheiro do jogo ou dinheiro que pode ser comprado pelo jogador fisicamente.

Figura 5. Tela de uma fazenda no Hay Day.

4. Metodologia

O desenvolvimento de um jogo deve ser pautado por criação de protótipos, onde as ideias e a mecânica do jogo são testadas, antes da criação do jogo em si. Entretanto, considerando que este trabalho se inspira em um jogo existente, com mecânica bem definida e próxima ao desejada (não necessitando assim a prototipação em papel/baixa fidelidade para investigar a mecânica do jogo), iniciou-se o desenvolvimento do jogo resgatando esta mecânica, registrando-a em um GDD (*Game Design Document*), e criando um protótipo de alta fidelidade. Assim para a produção desse jogo, um GDD foi criado, com o objetivo de descrever as informações mais importantes do jogo, como temática, mecânica, plataformas, inimigos e níveis para auxiliar no desenvolvimento do produto final.

Após a criação de uma versão inicial do GDD, iniciou-se a fase do desenvolvimento do jogo em sua versão de demonstração, onde além das funcionalidades básicas do protótipo, também possui os elementos da temática do jogo.

5. Desenvolvimento

O projeto foi iniciado com a criação do GDD para iniciar e guiar o protótipo, demonstrando as mecânicas básicas do jogo, sendo elas o recebimento de veículos solicitando serviços, a edição da oficina e o mercado de carros. A plataforma utilizada para o desenvolvimento foi a *Unity*, um motor de jogo consolidado no mercado com ferramentas e recursos próprios.

Dentro da plataforma do *Unity*, o programador possui acesso à ferramentas de modelagem de objetos 3D simples, que permite a inclusão de *features* adicionais com mais opções de desenvolvimento. Neste projeto, foi utilizada a *ProBuilder*, que permite mais opções para criação de objetos de cena, facilitando o trabalho e não sendo necessário utilizar ferramentas externas junto ao *Unity*.

Junto da criação de alguns dos objetos utilizados, a plataforma também possui uma loja de *assets* com recursos gratuitos e pagos que são disponibilizados pela própria comunidade para uso nos jogos. Possuindo alguns utilizados neste protótipo.

Além da criação desses objetos, a criação e gerenciamento dos recursos junto das regras utilizadas são feitas com a linguagem C# por meio de *scripts* escritos e atrelados aos objetos em cena. Essa linguagem, por se tratar de ser de alto nível, facilita a criação dos jogos na ferramenta, pois junto dos pacotes fornecidos pela plataforma .NET a Unity possui uma *library* com recursos a serem utilizados na plataforma.

5.1. Elaboração do GDD

O jogo pertence ao gênero de estratégia, que colocará o jogador para assumir o papel de um dono de oficina mecânica, e constantemente irá receber veículos com serviços variados de tempo e remuneração, onde caberá ao próprio jogador definir quais serviços irá realizar para maximizar o seu uso de tempo. Conforme o jogador evoluir seu nível, serão liberadas novas funcionalidades para sua oficina, como por exemplo adquirir novos carros e itens até a possibilidade de expansão da oficina (Anexo I).

5.2. Construção do cenário inicial

Na Figura 6 mostra o cenário principal do protótipo. A parte central é onde será a oficina, nas ruas laterais podem aparecer os veículos associados ao serviço a ser realizado e também existe a loja que permite a compra de itens ou veículos. O cenário foi criado após

a utilização de *assets* selecionados da própria loja da *Unity*, com recursos do *CarTown*, criados por BuzzKirill, no ano de 2019. Foram utilizados principalmente os modelos dos veículos e das vias no jogo, além de cubos e objetos modificados manualmente. Por se tratar de um protótipo, o jogador inicia com uma quantia significativa de dinheiro para comprar os itens disponíveis e poder subir seu nível com os serviços.

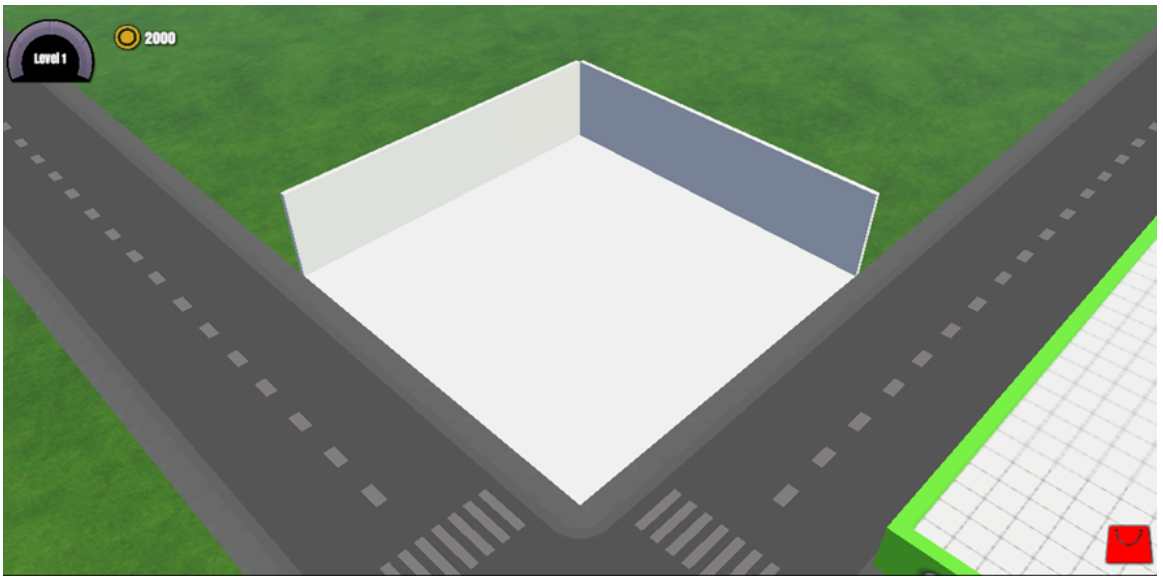


Figura 6. Cenário inicial.

5.3. Criação das telas da loja

A parte central de *gameplay* do jogador será a aquisição de veículos, onde ao abrir o menu de compras será apresentada a tela inicial da Loja (Figura 7). Após a seleção da opção de veículos, será apresentada a próxima tela (Figura 8), na qual o jogador poderá escolher o automóvel que queira adquirir e o menu abaixo (Figura 9) irá apresentar o nome e o preço do veículo. O botão Comprar só é liberado caso o jogador possua dinheiro o suficiente para compra do veículo.

Figura 7. Tela inicial do mercado.

Após efetuar a seleção de veículo, o jogador retornará para a sua oficina (Figura 10), onde poderá confirmar o local de criação do objeto ou cancelá-lo.

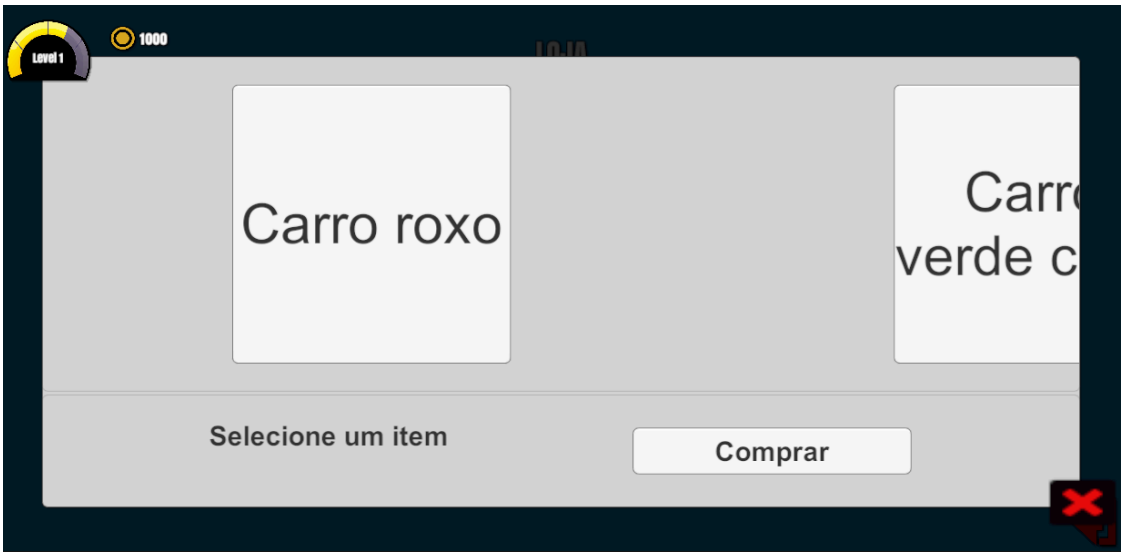


Figura 8. Menu de carros.

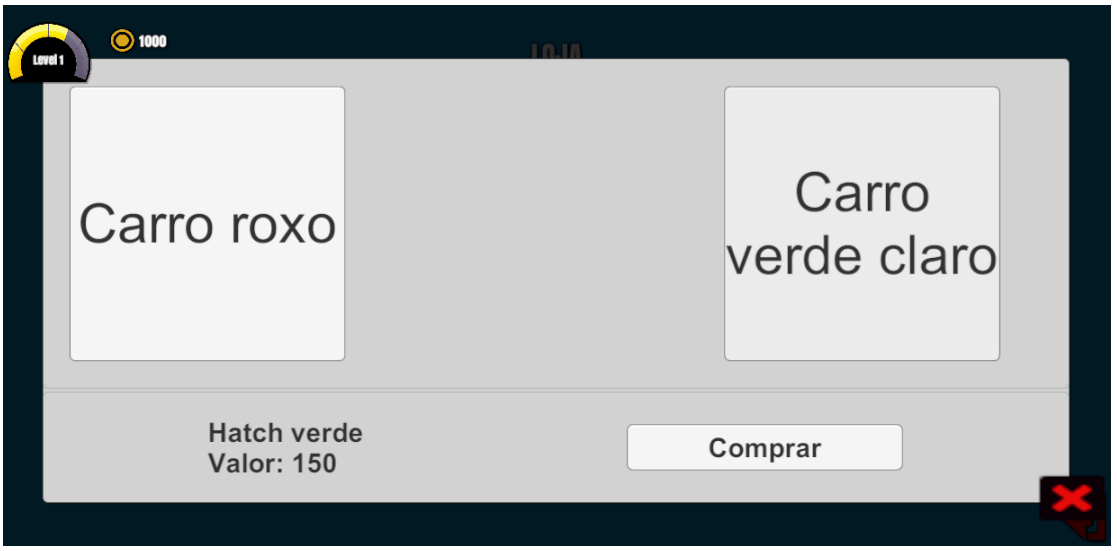


Figura 9. Seleção de compra.

Na Figura 11, é apresentado o código para gerenciamento das telas, quando clicado no botão da loja, é chamado o método da linha 114 “OpenShop()”, que ativa a primeira tela da loja (Figura 7) e desabilita a tela da oficina e passa o valor *true* para a variável global “itemSelecionado” que gerencia a movimentação da câmera. Caso selecione o botão de sair, é feito o processo inverso de ativações das telas através do método “CloseShop()” (linhas 121 a 126). Quando o jogador seleciona a opção de Carros, é chamado o método “OpenShopItems()” que ativa a tela do menu de carros (Figura 8) e deixa o botão de “Comprar” desabilitado com o valor de *false* (linhas 128 a 132).

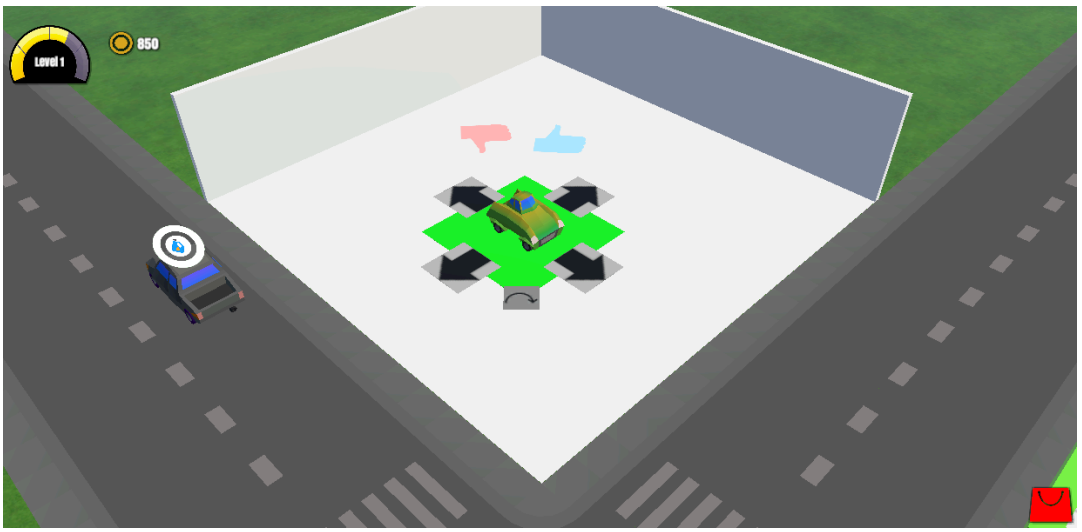


Figura 10. Confirmação da compra.

```
114    public void OpenShop()
115    {
116        panelShop.SetActive(true);
117        main.SetActive(false);
118        itemSelecionado = true;
119    }
120
121    public void CloseShop()
122    {
123        panelShop.SetActive(false);
124        main.SetActive(true);
125        itemSelecionado = false;
126    }
127
128    public void OpenShopCars()
129    {
130        panelCars.SetActive(true);
131        buttonCarBuy.enabled = false;
132    }
```

Figura 11. Código das telas do menu.

5.4. Criação da mecânica para movimentação de itens

A parte de movimentação dos itens se torna fundamental para o jogador configurar o *layout* e as decorações de sua oficina, necessário a validação de colisões e da área permitida para essa funcionalidade.

Qualquer veículo ou item que o jogador possuir terá sua área de validação, quando um desses objetos se encontra no espaço de outro objeto ou fora dos limites da oficina como mostrado na Figura 12, o objeto mostrará sua área em vermelho, indicando ser impossível salvar sua posição no local, ao colocá-lo em um local permitido ficará com a cor verde em destaque (Figura 13), seguindo para sua confirmação (Figura 14) apenas clicando fora do objeto. Este mesmo código também é utilizado para a validação de colisões com os itens dentro da oficina.

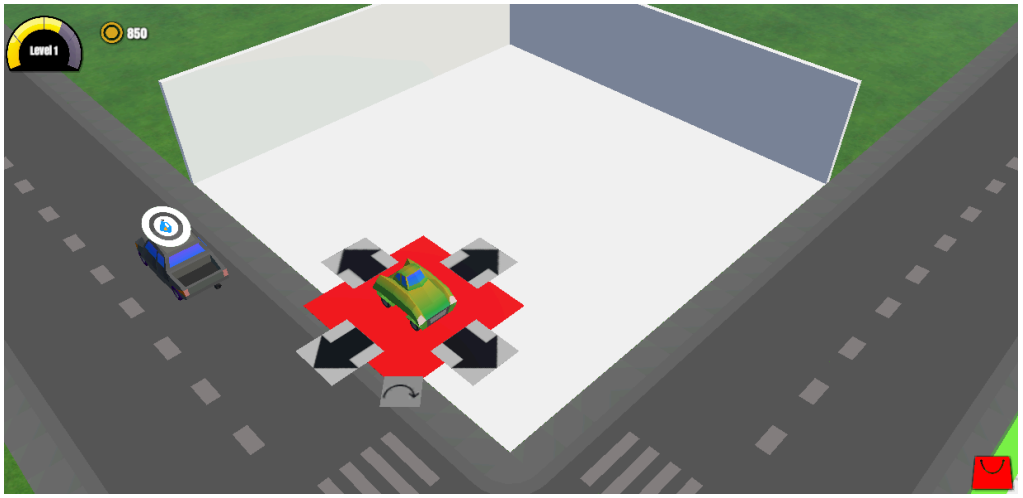


Figura 12. Área não permitida para dispor um elemento da oficina perceptível por meio da coloração vermelha abaixo do elemento.

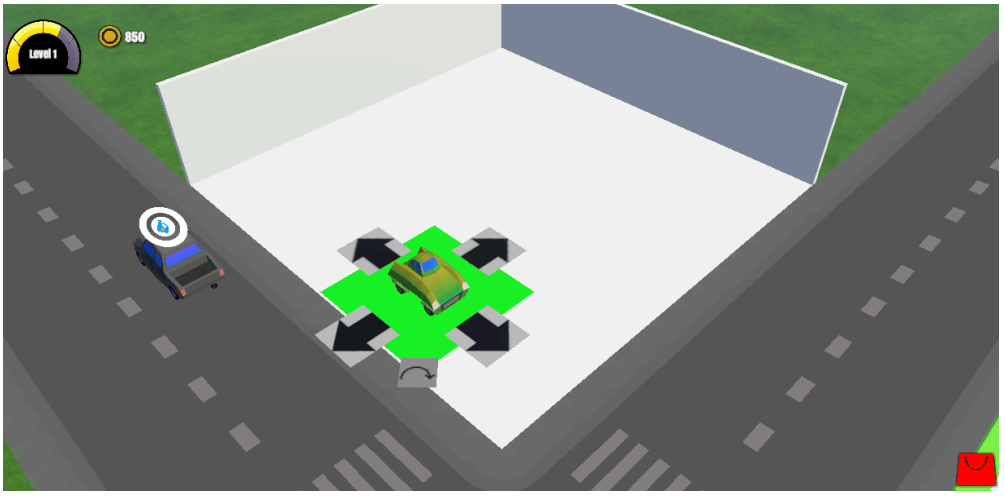


Figura 13. Área permitida para dispor um elemento da oficina perceptível por meio da coloração verde abaixo do elemento.

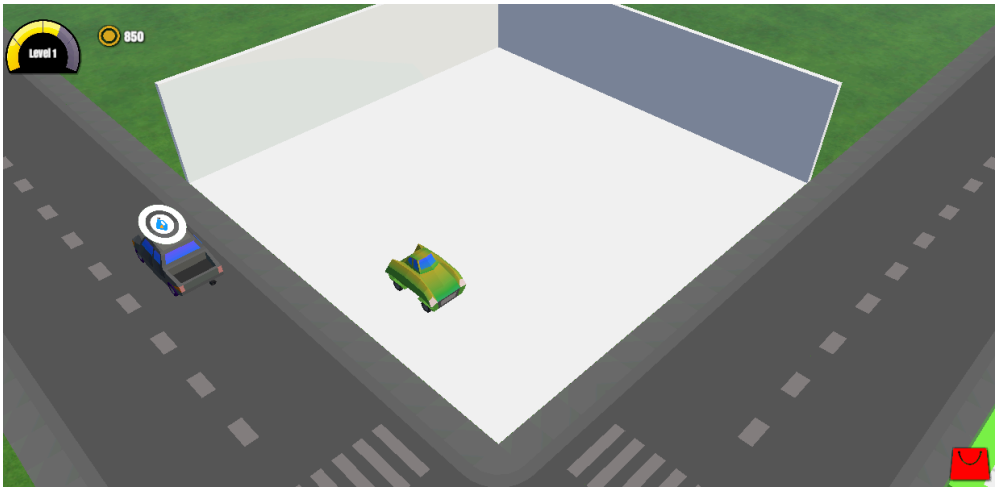


Figura 14. Confirmação de movimentação.

Na Figura 15 é descrito o código utilizado para as validações de colisão desses objetos. Na linha 92 é executado o método *FixedUpdate* da *API* de *script* da *Unity*, este método serve para possuir uma frequência fixa de atualizações independente da taxa de quadros durante a execução do jogo, ela é utilizada para executar o método "Colisoes()" com uma taxa de frequência mais alta que o convencional *Update*. Ao executar esta função, é validado se o objeto que possui o código está selecionado ou não, caso esteja, é criado uma variável de física que gera uma caixa de colisão do tamanho do objeto e confirma se ele está sobrepondo outro colisor (linha 102), caso esteja, a cor dessa caixa é alterada para vermelho na linha 106, caso não, verde (linha 110). E se a seleção do objeto for retirada, sua cor é apagada (linha 115).

```
92    private void FixedUpdate()
93    {
94        Colisoes();
95    }
96
97    //Validação de colisões
98    private void Colisoes()
99    {
100       if (selected)
101       {
102           colisao = Physics.CheckBox(new Vector3(transform.position.x, transform.position.y + 0.5f, transform.position.z),
103               new Vector3(2.5f, 0.3f, 2.5f), Quaternion.identity, layerMask, QueryTriggerInteraction.UseGlobal);
104           if (colisao)
105           {
106               GetComponent<Renderer>().material.color = colors[2]; //Vermelho
107           }
108           else
109           {
110               GetComponent<Renderer>().material.color = colors[1]; //Verde
111           }
112       }
113       else
114       {
115           GetComponent<Renderer>().material.color = colors[0]; //Retira Cor
116       }
117   }
```

Figura 15. Script de verificação de colisão utilizado para permitir ou bloquear que o jogador deixe um item em um determinado local.

5.5. Criação da mecânica de serviços

Para a criação e execução dos serviços, alguns passos foram efetuados. Inicialmente, o jogo irá instanciar carros de forma aleatória com uma imagem aleatória atribuída a ele, conforme a Figura 16, cada imagem possui um valor de experiência, dinheiro e tempo para sua conclusão pré-definidos.

Para o jogador conseguir iniciar um serviço precisa possuir ao menos um elevador disponível em sua oficina, e ao clicar em um dos veículos na rua a sua posição é alterada para o local do elevador (vide código da Figura 20), iniciando o processo visual (borda verde acima da imagem) apresentado na Figura 17 e alterado (para borda amarela) quando concluído (Figura 18). Após o processo finalizado, o jogador pode clicar no carro novamente, recebendo assim o *XP* (sigla para *experience*) e o dinheiro do serviço como mostrado na Figura 19 e pode iniciar um novo serviço utilizando o mesmo elevador. Essa experiência é atribuída para a barra de *Level* na parte superior esquerda, tendo conforme maior o nível, maior a quantidade de *XP* a ser adquirido.

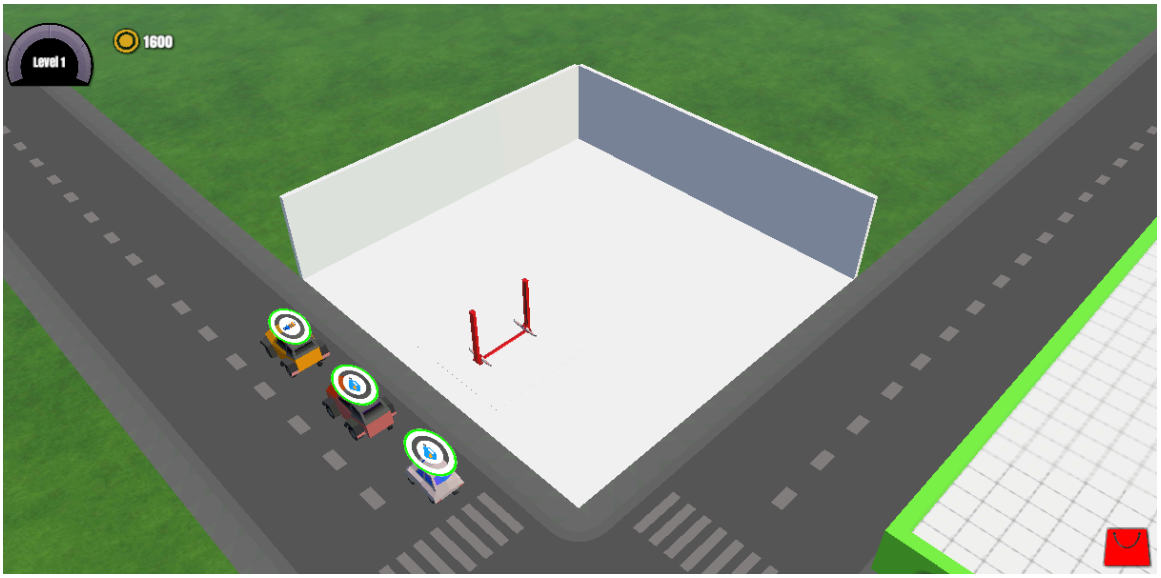


Figura 16. Criação dos serviços.

Figura 17. Execução de um serviço.

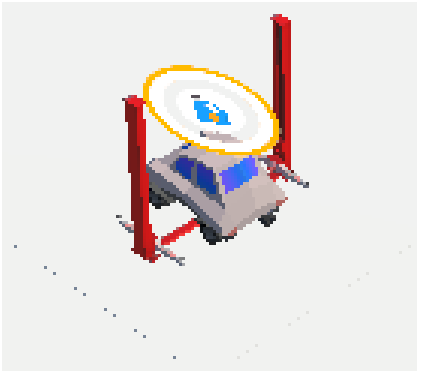


Figura 18. Conclusão do serviço.

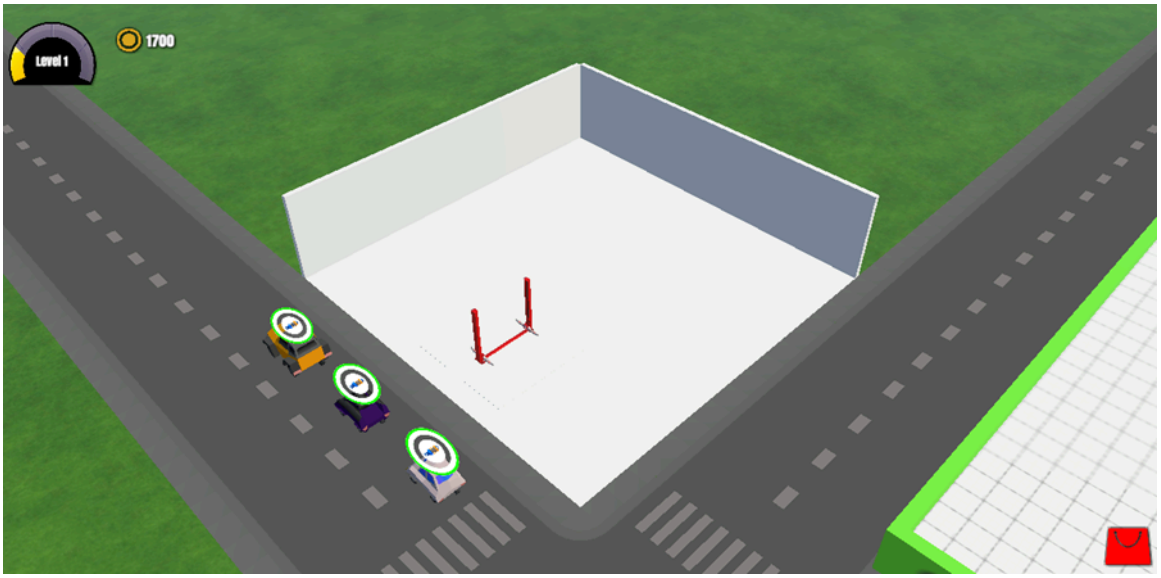


Figura 19. Recompensas recebidas.

Na Figura 20 é apresentado o código que busca elevadores ativos através do método "ProcuraLocal", que recebe como parâmetro o carro selecionado. Quando iniciado é feito uma busca na lista de elevadores que o jogador possui e se encontrar um elevador que esteja ativo para receber um novo serviço então na linha 125 o carro é movimentado para a posição do elevador, libera a sua posição na rua para que outro veículo seja gerado (linha 126), desabilita o elevador para novos serviços (linha 127), ativa a execução de seu próprio serviço (linha 128) e passa o valor 0 para a variável *destinyCount,* que executa o processo para aparecer um novo veículo. Por fim, ele retorna o elevador utilizado ou caso não tenha encontrado nenhum elevador disponível nenhuma ação é tomada.

```
119    public static Elevador ProcuraLocal(Carro car)
120    {
121        foreach (Elevador elevador in elevadores)
122        {
123            if (elevador.Active == true)
124            {
125                car.GetComponent<Transform>().transform.position = elevador.Posicao;
126                car.CarDestiny.liberated = true;
127                elevador.Active = false;
128                car.Activate = true;
129                destinyCount = 0;
130                return elevador;
131            }
132        }
133        return null;
134    }
```

Figura 20. Procurando Elevador disponível.

A criação dos serviços se dá pela utilização de imagens adicionadas em um objeto auxiliar dentro do jogo conforme Figura 21, e são selecionadas aleatoriamente quando os carros são criados na cena (descrito na Figura 22). Neste código é utilizado o método “Update()” da *Unity* para que possa ser utilizado funções de tempo, dessa forma é validado se as posições possíveis estão disponíveis. Caso não estejam, o processo da variável “globalValidation” é pausado e a contagem de destinos zerada. Caso possua locais disponíveis, a “globalValidation” é positiva e então iniciado o processo de validação de tempo que quando atinge um limite, ativa a rotina “GeraServiço()” da linha 59. Esta rotina instancia aleatoriamente um dos carros e uma das imagens disponíveis, aplica valores para um objeto “Carro” e o instancia em cena, bloqueando a posição onde foi criado para que quando chamado novamente já selecione a próxima posição.

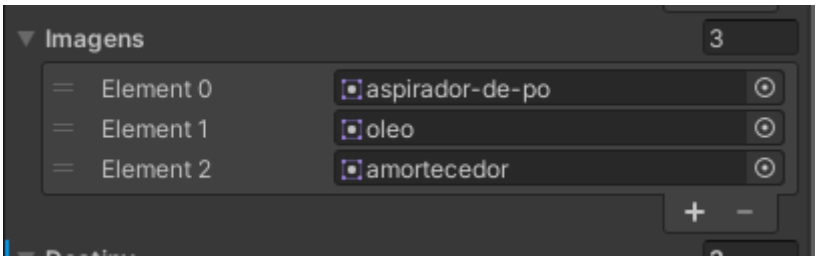


Figura 21. Adição de Imagens.

```
34  void Update()
35  {
36      timer += Time.deltaTime;
37
38      if (destiny[0].Liberated == false && destiny[1].Liberated == false && destiny[2].Liberated == false)
39      {
40          globalValidation = false;
41          destinyCount = 0;
42      }
43      else globalValidation = true;
44
45      if (timer > 5 && globalValidation)
46      {
47          StartCoroutine(GeraServiço());
48          timer = 0f;
49      }
50
51
52      tempoInicio += Time.deltaTime;
53
54  }
55
56  #region Métodos de Serviços
57
58
59  IEnumerator GeraServiço()
60  {
61      if (destiny[destinyCount].Liberated == true)
62      {
63          randomCar = Random.Range(0, carros.Length);
64          randomImage = Random.Range(0, imagens.Length);
65          carros[randomCar].indice = randomImage;
66          carros[randomCar].ImageService.GetComponent<Image>().sprite = imagens[randomImage];
67          carros[randomCar].PosicaoUtilizada = destinyCount;
68          carros[randomCar].CarDestiny = destiny[destinyCount];
69          carros[randomCar].CarDestiny.liberated = false;
70          Instantiate(carros[randomCar].GetComponent<Transform>(), destiny[destinyCount].transform.position, destiny[destinyCount].transform.rotation, parent);
71          yield return new WaitForSeconds(1);
72          destiny[destinyCount].Liberated = false;
73          destinyCount++;
74
75      }
76  }
```

Figura 22. Instanciando veículos de serviço.

Na Figura 22 é apresentado o código chamado quando um veículo é selecionado pelo jogador e ele é movimentado para a posição do elevador, validado através da linha 43. Dessa forma uma variável de tempo é iniciada para que uma imagem “IncompleteBackground” seja alterada de acordo com o tempo corrido através do método “fillAmount”, enquanto seu tempo pré definido não é atingido. Quando a variável de tempo atinge seu limite, a imagem “IncompleteBackground” é desabilitada e a “FullBackround” é ativa, indicando a conclusão do serviço, liberando também a variável “UnlockXP” para que o jogador possa receber suas recompensas.

```
41      void Update()
42      {
43          if(activate == true)
44          {
45              tempoInicio += Time.deltaTime;
46              if(tempoInicio < TimeService[indice])
47              {
48                  IncompleteBackground.GetComponent<Image>().fillAmount = (tempoInicio / TimeService[indice]);
49              }
50              else
51              {
52                  IncompleteBackground.SetActive(false);
53                  FullBackground.SetActive(true);
54                  UnlockXP = true;
55              }
56          }
57      }
```

Figura 22. Procurando Elevador disponível.

6. Conclusão e Trabalhos Futuros

Neste trabalho foi possível analisar e recriar algumas das mecânicas principais do jogo Car Town, abordando similaridades de sua execução e como pode ser desenvolvido um novo jogo com as ferramentas atuais do mercado. Após a criação do *GDD* onde são descritas inicialmente todas as abordagens a serem utilizadas, o desenvolvimento da demonstração do jogo foi criado. Nela conseguimos apresentar o núcleo da *gameplay* como o nível do jogador e as mecânicas de dinheiro.

Dos objetivos propostos, todo o ciclo de vida do jogo pôde ser efetuado, tendo o jogador suas formas de adquirir dinheiro e onde gastá-lo, a execução de serviços a serem prestados e o recurso de movimentação dos itens adquiridos.

Durante a execução das funções, a ideia proposta incluiria mecânicas de missões, tutorial, mais opções de veículos com customizações, além de opção de expansão da oficina. Porém por se tratar de apenas uma demonstração do jogo que indica suas principais mecânicas, qualquer atualização que possa ser efetuada já possui a base necessária. O código desenvolvido do protótipo encontra-se no seguinte link https://github.com/JuanBailke/CarTown-Clone.

E para atualizações futuras algumas alterações devem ser incluídas, como a imagem dos objetos no menu de compras, além de ajustes na qualidade da imagem para que possam ter um design mais atrativo. Também funcionalidades *multiplayer*, como visitar a oficina de amigos.

Foram abordadas e utilizadas durante o desenvolvimento conteúdos aprendidos em sala de aula, indicando principalmente as matérias de Linguagem de Programação, Programação Orientada a Objetos, Engenharia de Software, Estruturas de Dados e Interação Humano-Computador, além de novos estudos na linguagem utilizada e na documentação das ferramentas.

Referências

CAILLOIS, R. **Os jogos e os homens**: A máscara e a vertigem. Petrópolis: Editora Vozes, 2017.

CASARINI, M.; PETRY, L. C. Proposta de Framework para desenvolvimento de level design. *In*: LEMES, D. de O. (org.). **Estudos sobre o desenvolvimento de games**. São Paulo: Editora C0D3S, 2019.

TECHTUDO. **Clash of Clans**: veja cinco motivos que explicam o sucesso do jogo. TechTudo. 2015. Disponível em: https://www.techtudo.com.br/noticias/2015/02/clash-clans-veja-cinco-motivos-que-explicam-o-sucesso-do-jogo.ghtml. Acesso em: 15 jan. 2024.

CONTESINI, L. **O trailer de "Mad Max: Fury Road", o fim de Car Town, os detalhes do futuro Toyota Supra e mais! | FlatOut**. FlatOut. 2014. Disponível em: https://flatout.com.br/o-trailer-de-mad-max-fury-road-o-fim-de-car-town-os-detalhes-futuro-toyota-supra-e-mais/#:~:text=Jogo%20Car%20Town%20será%20desativado%20por%20seus%20desenvolvedores&text=Confira%20a%20mensagem%20na%20íntegra,longo%20desses%20últimos%20quatro%20anos.. Acesso em: 29 set. 2023.

L., Sergio. **Miniclip: Queremos que tragam o jogo Car Town de volta para o Facebook e para Android**. Avaaz. 2017. Disponível em: https://secure.avaaz.org/community_petitions/po/Miniclip_Queremos_que_tragam_o_jogo_Car_Town_de_volta_para_o_Facebook_e_para_Android/. Acesso em: 14 set. 2023.

PETRY, L. C. O conceito ontológico de jogo. *In*: ALVES, L.; COUTINHO, I. J. (org.). **Jogos digitais e aprendizagem**: fundamentos para uma prática baseada em evidências. Campinas, SP: Papirus, 2016.

RIBEIRO, D. **Como jogar Car Town, o game social para quem é apaixonado por carros**. TechTudo. 2013. Disponível em: https://www.techtudo.com.br/noticias/2013/05/como-jogar-car-town-o-game-social-para-quem-e-apaixonado-por-carros.ghtml. Acesso em: 15 jan. 2024.

SALEN, K.; ZIMMERMAN, E. **Regras do jogo**: fundamentos do design de jogos. Volume 3. São Paulo: Blucher, 2012.

SCHUYTEMA, P. **Design de games**: uma abordagem prática. São Paulo: Cengage Learning, 2008. 447 p.

SILVA, C. **Principais gêneros de jogos e suas características**. Go Gamers. 2021. Disponível em: https://gogamers.gg/gamepedia/principais-generos-de-jogos/. Acesso em: 19 out. 2023.

SOUZA, A. **Lembra de Hay Day? Tudo sobre o jogo estilo 'fazendinha' para celular.** TechTudo. 2023. Disponível em: https://www.techtudo.com.br/guia/2023/04/lembra-de-hay-day-tudo-sobre-o-jogo-estilo-fazendinha-para-celular-edjogos.ghtml. Acesso em: 18 jan. 2024.

SIOUX GROUP, GO GAMERS. **Pesquisa Game Brasil.** [s.l.:s.n.], 2021.

WIKIPÉDIA. **Unity – Wikipédia, a enciclopédia livre**. 2023. Disponível em: https://pt.wikipedia.org/wiki/Unity. Acesso em: 18 fev. 2024.

XEXÉO, G. **O que são jogos.** LUDES. Rio de Janeiro, 2013. Disponível em: https://ludes.cos.ufrj.br/wp-content/uploads/2016/07/LJP1C01-O-que-sao-jogos-v2.pdf. Acesso em: 15 mar. 2022.

ANEXO I - *Game Design Document*

Car Town

Gênero:

Estratégia

Elementos de Gameplay:

Gerenciamento da oficina:

- Administração de serviços para recebimento de dinheiro e XP;
- Personalizar a oficina.

Gerenciar veículos:

- Comprar;
- Personalizar;
- Vender.

Jogador(es):

1 Jogador.

ESPECIFICAÇÕES TÉCNICAS:

Técnica:

Será feito com técnica de 2.5D.

Visualização:

Câmera multiplano.

Plataforma:

Android.

Linguagem:

C#.

Dispositivos:

Mobile.

GAME PLAY:

Para o jogador evoluir dentro do jogo, ele irá receber periodicamente alguns veículos de NPCs (*Non-playable character*) que irão solicitar serviços para seu veículos, dentre esses serviços terão variedades para que os rápidos deem menos dinheiro e experiência e os demorados deem mais, consequentemente. Dessa forma, caberá ao jogador escolher qual o melhor serviço para o momento e sua disponibilidade de acessar o jogo para atualizar os serviços.

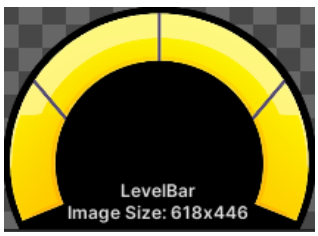
Com o dinheiro adquirido o jogador terá a opção de comprar decorações, expandir sua oficina e comprar veículos, que serão usados para exibição ou modificados para utilizar em missões.

Os veículos terão opções internas para modificação por parte do usuário, como peças de lataria, cor e mecânicas, para melhorar seu desempenho.

Conforme o nível do jogador aumenta, serão disponibilizadas missões para ele, onde poderá utilizar os veículos que adquiriu para realizar corridas, e outras *quests* relacionadas, além de missões de serviços especiais que podem aparecer em sua oficina.

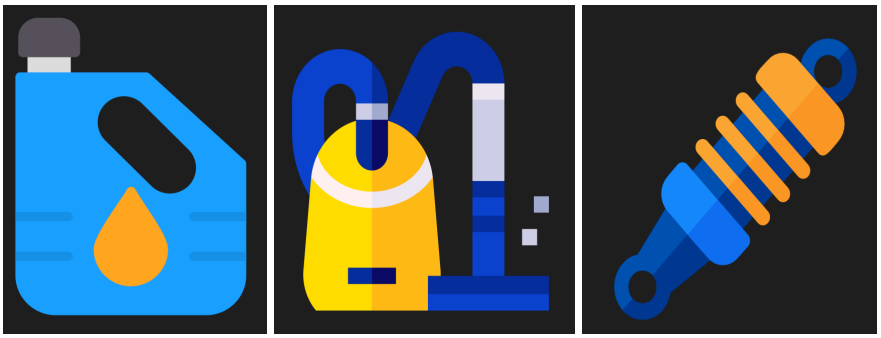
IMAGENS:

Level:



Carros:

Serviços:



Documento Digitalizado Público

TCC - Anexo I

Assunto: TCC - Anexo I
Assinado por: Andre Constantino
Tipo do Documento: Relatório
Situação: Finalizado
Nível de Acesso: Público
Tipo do Conferência: Documento Digital

Documento assinado eletronicamente por:

- **Andre Constantino da Silva**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 08/03/2024 18:01:12.

Este documento foi armazenado no SUAP em 08/03/2024. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1605704
Código de Autenticação: 1acaacbf9e