PCS3559/PCS3859 - Tecnologias para Aplicações Interativas

Figure It Out - AR Skins

Equipe:

Danilo Oliveira Sobral	9351926
Hélio Jin Wu Kim	9833106
Isabella de Camargo Nigro	4470889
Kaíque Maestrini Sacchi	9832961

Professores:
Ricardo Nakamura
Romero Tori

1. Introdução

Atualmente, há um crescimento acelerado da cultura *geek*. Com a popularização de *games*, animes e mangás, superproduções nos cinemas como os filmes da Marvel e livros e quadrinhos de grande sucesso, ocorre um impulsionamento da procura por produtos relacionados a esses universos fictícios. Cada vez mais os fãs estão consumindo itens diversos, desde roupas até as populares *action figures* [6, 7].

Estima-se que, anualmente, no mundo, o mercado de produtos *geek* movimenta mais de 138 bilhões de reais [1], sendo as *action figures* ou figuras de ação um dos destaques em meio aos itens da cultura *geek*. As *action figures* são estatuetas que representam personagens dessas produções e são itens colecionáveis que podem chegar a valores altíssimos de até milhares de reais.

Figura 1: Miniatura da personagem Tracer do jogo *Overwatch*, que pode custar mais de US$150,00 [2]

Paralelo ao aumento da venda das estatuetas colecionáveis, a popularização do mundo *geek* também impulsiona o mercado dos *games*, que, motivado pelo aumento de demanda, tenta encontrar modelos de negócio mais lucrativos. Um desses novos modelos é o baseado na

venda de cosméticos, adotado cada vez mais por empresas importantes da indústria. Jogos como *League of Legends*, *Fortnite*, *Team Fortress 2*, *Dota*, CS:GO e *MapleStory* permitem a customização da aparência dos personagens, fornecendo diferentes *skins* para cada modelo que podem ser compradas de vendedores oficiais ou de outros jogadores. Para o *League of Legends*, por exemplo, os preços das *skins* vendidas na loja oficial variam entre R$10,00 e R$75,00, enquanto versões mais raras podem ser vendidas por milhares de reais [3]. Esse alto resulta em compradores cada vez mais exigentes, que buscam a maior quantidade de informaçẽos e detalhes sobre cada customização que querem comprar.

Figura 2: *Skin* customizada do personagem *Fiddlesticks* do jogo *LoL*, conhecido como *Pumpkinhead Fiddlesticks*, podendo custar em torno de US$50,00 [4]

Ao levar em consideração o crescimento simultâneo dessas duas áreas (venda de *action figures* e *skins*), percebe-se um grupo de consumidores com gostos similares que podem ser auxiliados pela criação de uma ferramenta de visualização 3D dos modelos, antes de realizarem suas compras. Além disso, as empresas de jogos responsáveis pelas vendas das *skins* dos seus personagens muitas vezes também vendem colecionáveis dos seus jogos, como o caso da *Riot Games* (desenvolvedora do *LoL*) e da *Blizzard* (desenvolvedora de *Overwatch*). Dessa forma, uma ferramenta de visualização 3D que seja administrada pela própria empresa pode impulsionar as vendas das duas categorias de produto.

2. Objetivo

A proposta do grupo é a construção de uma ferramenta de visualização 3D dos modelos customizáveis do jogo em ambiente de realidade aumentada (AR). Tal aplicação consiste na apresentação digital do personagem em um ambiente real por meio da câmera do celular, de maneira a permitir uma visão mais detalhada - no caso do comércio de *skins* - ou analisar como a miniatura ficaria exposta - no caso do comércio de *action figures*. A visualização em 360° dos modelos, a possibilidade de colocá-los digitalmente sobre um plano definido e uma customização de parâmetros relativos a essa visualização permitiriam ao consumidor maior certeza no momento de sua compra frente ao alto preço que tais produtos podem atingir, ou seja, ele poderia ver como o produto ficaria em sua casa antes de comprá-lo no caso das *action figures* ou como seria sua aparência dentro do jogo, no caso das *skins*. Dessa forma, a

sua certeza de satisfação com a compra seria maior, possivelmente aumentando as chances de efetivá-la.

Figura 3: Proposta de funcionamento do aplicativo ao visualizar *action figures*

3. Especificações do projeto

3.1. Requisitos

O grupo propõe-se, então, a desenvolver um aplicativo para auxiliar os usuários na visualização de diferentes *skins* presentes em jogos e a ter uma boa estimativa de como ficaria a *action figure* num ambiente do mundo real (como uma prateleira com outros *action figures*, mesa etc). Assim, a ideia é que o aplicativo permita ao usuário selecionar diferentes *skins* para o modelo selecionado, posicioná-lo no mundo real e modificar seu tamanho, possibilitando uma visualização detalhada dele.

Assim, as seguintes funcionalidades são o conjunto base de *features* que estão previstas para o aplicativo no escopo da disciplina

- Posicionar modelo 3D em um plano detectado
- Deslocar o modelo posicionado
- Rotacionar o modelo posicionado

- Modificar a escala do modelo posicionado
- Substituir o modelo mostrado
- Interface com auxílio de botões
- Listagem simples de modelos disponíveis
- Remover um modelo já colocado

As seguintes funcionalidades, por sua vez, serão implementadas caso o escopo mínimo tenha sido atingido, por serem aspectos de maior complexidade técnica e não serem essenciais para o funcionamento do produto mínimo viável (MVP):

- Alterar a profundidade da projeção (Depth API)
- Reprodução de animações pré-definidas
- Configuração de Iluminação Artificial
- Interface por meio de gestos
- Marketplace simples (melhorar estética da listagem)

O aplicativo será desenvolvido utilizando as ferramentas de AR disponibilizadas no motor de jogo *Unreal Engine 4*, em particular o *ARCore*, desenvolvido pela *Google* para a criação de apps de realidade aumentada em celulares *Android*. A princípio, planeja-se usar modelos 3D disponibilizadas gratuitamente na loja de *assets* da *game engine*, como os modelos dos personagens do jogo *Paragon.*

Figura 4: Modelos do personagem *Narbash* do jogo *Paragon*, disponibilizados gratuitamente no *UE4 marketplace* [5]

3.2. Cronograma

Com base nos estudos realizados e no entendimento da proposta e dos requisitos do projeto, elaborou-se o cronograma a seguir. Os trechos destacados em amarelo indicam dias de apresentação de resultados e feriados previstos. Os em magenta, por sua vez, se referem a tarefas adicionais, previstas apenas após a finalização do MVP e às quais o grupo não se compromete a entregar.

Período	Atividades Propostas
21/09 - 27/09	● Entrega da Proposta ● Estudo da Unreal Engine 4 ● Carregar modelos 3D na Unreal Engine 4
28/09 - 04/10	● Estudo do ARCore ● Primeiro protótipo usando o AR ● Visualizar modelo 3D de personagem com o AR
05/10 - 11/10	Apresentação de andamento dos trabalhos ● Adicionar controles para rotacionar o modelo ● Adicionar controles para escalar o modelo
12/10 - 18/10	Feriado ● Mudar modelo sendo projetado pelo aplicativo ● Criar listagem com todos os modelos 3D disponíveis
19/10 - 05/10	● Substituição de modelo projetado em tempo real
26/10 - 01/11	● Adicionar controles para reposicionar o modelo em outro plano
02/11 - 08/11	Feriado ● Reset da sessão AR ● Finalização do MVP
09/11 - 15/11	● Melhoria estética do Market Place ● Desenhar ícones dos botões
16/11 - 22/11	● Executar testes de usuário para melhorias da interface ● Calibração dos gestos (Parametrização)
23/11 - 29/11	● Pesquisa sobre Iluminação artificial e implementação (se possível)
30/11 - 06/12	● Procurar modelos com animação ● Pesquisa sobre animação e implementação (se possível)
07/12 - 13/12	● Preparação para apresentação final
14/12	Apresentação final

4. Projeto e desenvolvimento

4.1. Estudo de interface

Para o desenvolvimento de uma interface mais adequada ao usuário do aplicativo, é importante conhecê-lo. Para tanto, foi necessário caracterizar o público alvo do produto e, com base no

observado, foi elaborada uma interface que traz as funcionalidades desejadas para o projeto, mantendo-se amigável e intuitiva ao usuário.

4.1.1. Caracterização do usuário

O usuário alvo da plataforma é o consumidor de *skins* e miniaturas, imerso na cultura *geek*, em geral jovem e familiarizado com múltiplas interfaces tecnológicas, como as interfaces de *games* e as telas *touch screen* dos *smartphones* atuais.

Pelo constante contato com tais tecnologias, o usuário já possui alguma intuição sobre controles *touch screen*, como comandos de *swipe* ou *pinch* para dar *zoom*. Tais comandos serão utilizados para permitir as funcionalidades de ajuste de tamanho e posição dos modelos 3D no aplicativo.

Além disso, como o aplicativo seria usado como uma plataforma de venda das *skins* e miniaturas, é importante seguir padrões do mercado para *layouts* de *market places*, facilitando o entendimento das interfaces do produto.

4.1.2. Planejamento da interface

O aplicativo terá duas principais telas, cada uma com suas funcionalidades e, portanto, interfaces diferentes. A primeira tela apresenta o menu de escolha do modelo a ser visualizado em ambiente 3D. Essa tela possui uma lista com *scroll* composta por pequenas *previews* de cada modelo e suas informações, como nome do personagem, mídia de origem e preço, além de uma *search bar* para filtrar pelo nome do personagem. Um protótipo do *design* foi desenvolvido para ilustrar este menu de escolhas.

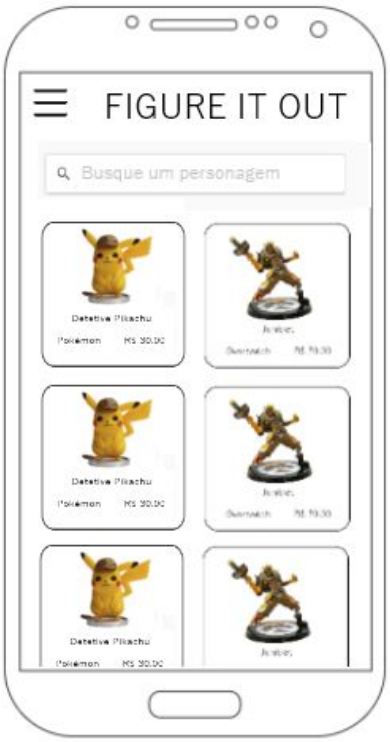


Figura 5: Protótipo do menu de *skins* e miniaturas

Nessa tela, optou-se por seguir os padrões de aplicações semelhantes, sem trazer um aspecto mais imersivo à interface, visto que tal alteração poderia afetar a interação do usuário com o aplicativo.

Ao clicar em uma das opções, o usuário é levado à segunda tela do projeto, a visualização do modelo em AR. Essa tela é composta pela imagem obtida pela câmera com o modelo 3D renderizado no ambiente e, além disso, botões indicando as diferentes funcionalidades propostas. Abaixo está um protótipo desta tela.

Figura 6: Protótipo da tela de visualização AR.

As três funcionalidades oferecidas são fazer a translação do modelo (opção à esquerda), rotacionar o modelo (opção do meio) e mudar seu tamanho (opção à direita). Ao clicar em uma das opções, o app reconhecerá certos gestos no *touch screen* para realizar a funcionalidade.

Para a translação, o usuário poderá fazer comandos de *swipe* para a esquerda/direita, movendo o modelo no sentido desejado. Não será possível alterar a posição no eixo Z, uma vez que a miniatura está renderizada no plano detectado pelo ARCore.

Para a rotação, o usuário poderá fazer *swipes* para esquerda/direita, rotacionando o modelo em torno do seu eixo Z nos sentidos horário e anti-horário.

Por fim, na terceira opção, o usuário poderá utilizar o gesto de *pinch and zoom* para aumentar ou diminuir a escala do modelo.

A implementação dos gestos será feita com auxílio de um plug-in da Unreal Engine, chamado *Ultimate Touch Components*, disponibilizado pela *Neo Wave Games* no *marketplace* da *Epic Store*. Este plug-in normalmente é pago, porém foi disponibilizado gratuitamente durante o mês de setembro.

4.2. AR

4.2.1. Estudo sobre Unreal Engine 4 e Modelos 3D

O primeiro objetivo do cronograma é estudar a plataforma da Unreal Engine e, como entregável, conseguir carregar modelos 3D na plataforma. O conhecimento adquirido será usado no produto final para carregar os modelos que serão apresentados através da realidade aumentada (AR), tópico de estudo da semana seguinte (segundo o cronograma inicialmente estipulado). Como a modelagem 3D não é o foco do trabalho, e sim o uso dos modelos para visualização, não planejamos modelar nenhum personagem. Ao invés disso, usaremos modelos disponíveis na Internet e no próprio *marketplace* da Unreal Engine. Nos interessa encontrar, preferencialmente, modelagens de personagens de jogos (já que esse é o modelo-alvo do projeto), que possam ser usadas no produto final.

Pesquisa de modelos disponíveis:

Após breve pesquisa online, encontramos modelos do jogo Paragon, desenvolvido pela própria empresa que gere a Unreal Engine - a Epic Games - disponíveis para download. Os *assets* são anunciados e disponibilizados pelo site da empresa [8]. A coleção é composta por uma série de modelos 3D dos personagens do jogo, criadas a um custo de mais de 17 milhões de dólares, e agora gratuitas para download.

Adição dos modelos na plataforma:

Decididos e obtidos os modelos, é necessário agora realizar a tarefa final da semana: carregá-los em um projeto da Unreal Engine. Para isso, a principal referência usada foi um vídeo tutorial, disponibilizado na plataforma do YouTube [9].

Seguindo o tutorial, o seguinte passo a passo foi identificado e realizado:

1. Adquirir os modelos em questão por meio do marketplace da Unreal Engine

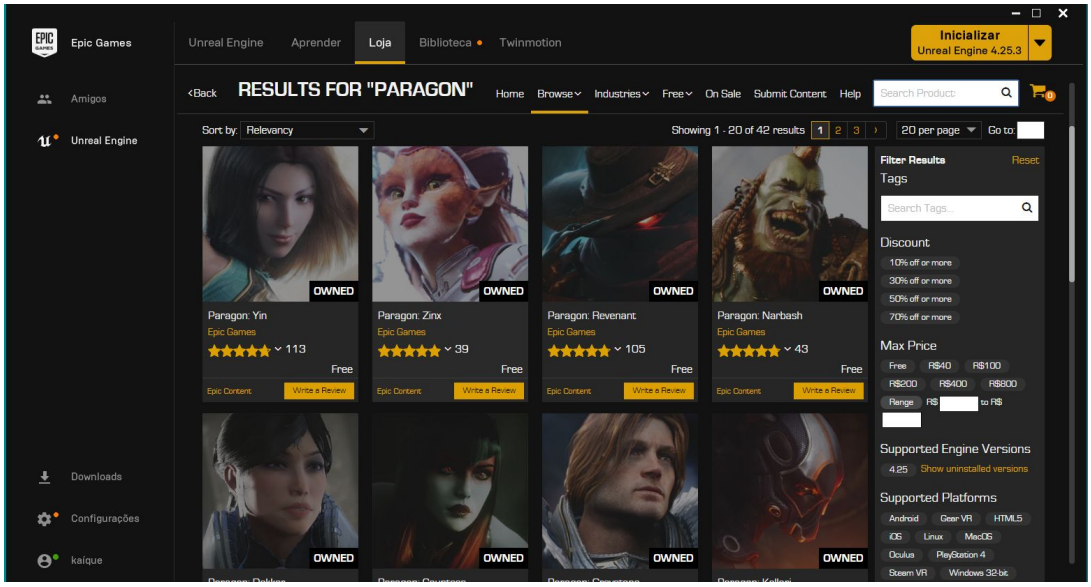


Figura 7: Modelos do Paragon no marketplace da UE4.

2. Criar um projeto, usando qualquer um dos templates (o em primeira pessoa pode ser melhor para visualizar o modelo de forma mais fácil).

3. Por meio do launcher da Epic Games, listar os modelos adquiridos e adicionar alguns ao projeto. Eles serão adicionados na aba do Content Browser

4. Para adicionar um dos personagens selecionados, acessar no Content Browser, em ‘Paragon<Nome_do_personagem>/Characters/Heroes/<Nome_do_personagem>/Meshes’. Dentro dessa pasta, existe um arquivo de Mesh nomeado com o nome do personagem. Clicar duas vezes no arquivo caso queira visualizá-lo separadamente. Uma nova aba de visualização do modelo será apresentada

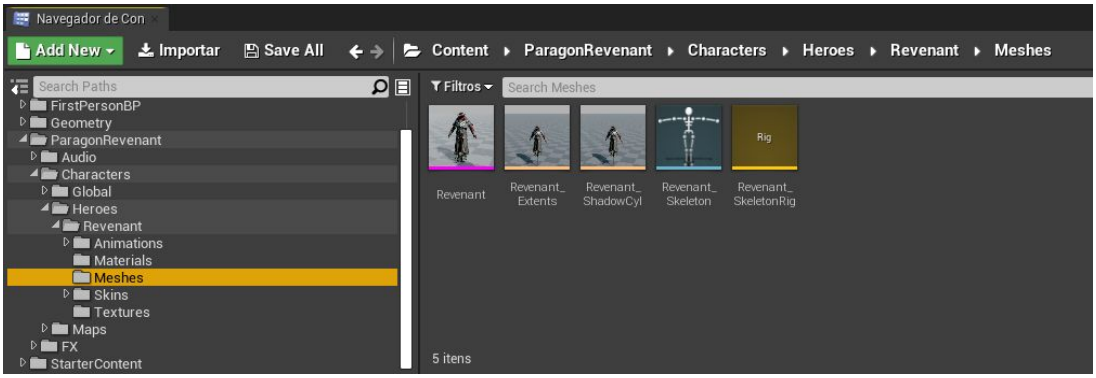


Figura 8: Personagem no Content Browser

5. Arrastar o arquivo para o meio da cena. Ele será adicionado.

Figura 9: Personagem adicionado à cena

Testes das funcionalidades:

Foi observado que os modelos dos assets do Paragon são muito grandes, o que é de se esperar de assets AAA para um jogo para PC/PS4, que não precisa se preocupar tanto com otimização de tamanho de arquivos e memória quanto aplicações para celular.
Sendo assim, tivemos que selecionar outros modelos do marketplace, a priori, alguns assets do jogo Infinity Blade (jogo mobile, com modelos muito menores).

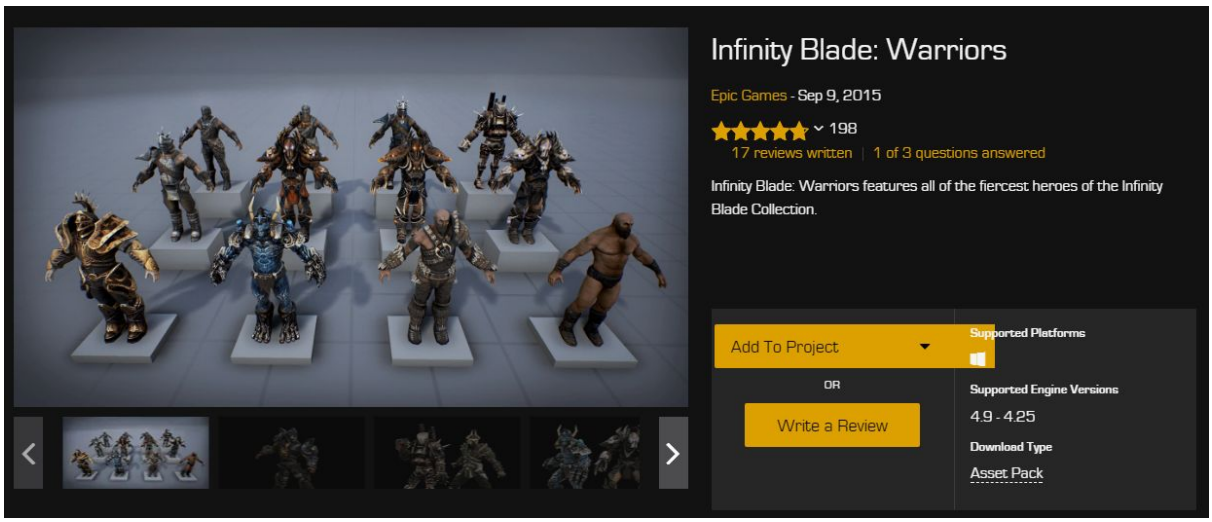


Figura 10: Modelos do Infinity Blade no Marketplace.

4.2.2. Implementação do AR

Utilizou-se a estrutura semelhante ao que é mostrado no quickstart da Unreal para o ARCore [10], para poder detectar planos e então instanciar um objeto no ponto em que o usuário clicar. A seguir, imagem de parte do código do projeto inicial, com código semelhante ao encontrado no link mencionado:

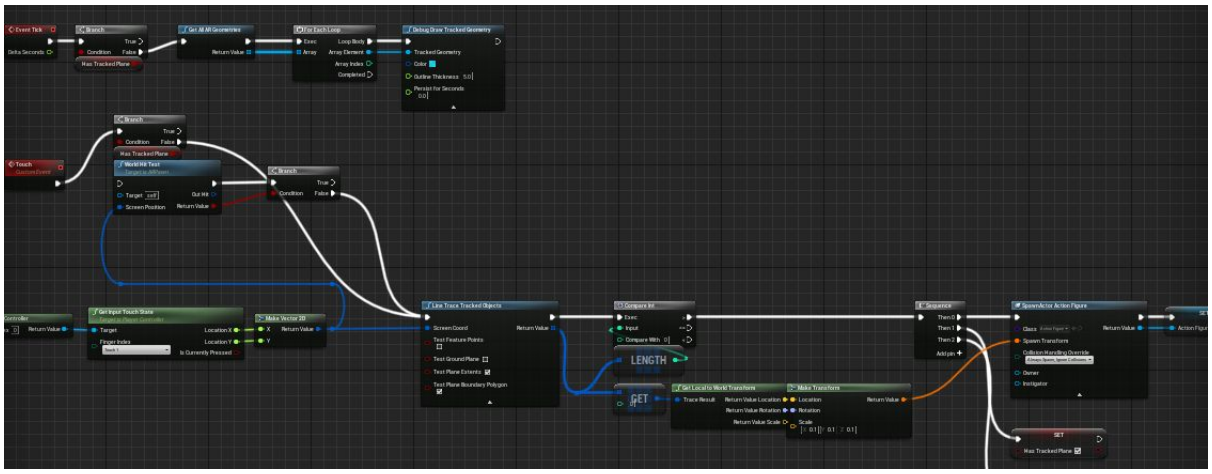


Figura 11: Demonstração do fluxo da blueprint usada para detecção de planos. A blueprint em questão pode ser acessada pelo repositório do grupo, e vista em detalhe.

Todavia, quando foram feitos testes iniciais, além do AR não estar conseguindo lidar bem com o movimento do usuário no mundo real e manter o track de onde cada objeto deveria estar posicionado no mundo, o objeto não parecia estar sendo instanciado corretamente no local que a pessoa clicou na tela. Além disso, no projeto do grupo, alguns materiais dos assets pareciam não estar sendo renderizados corretamente, ficando com a cor preta.

Sabe-se que isso não é erro da Engine (4.25 e 4.26p1) pois foram feitos testes com o ARTemplate dessas versões e esses problemas não estavam presentes. Foi feita uma comparação de quais configurações poderiam estar diferentes entre os templates e o projeto do grupo, na tentativa de sanar esse problema.

A análise e busca do problema não foi conclusiva. Após realizar mudanças no projeto em configurações que possivelmente causariam problemas, para tentar deixá-lo parecido com o template, decidimos recomeçar, partindo do template disponibilizado na plataforma. Assim foi feito, e foram adicionadas, gradualmente, as mudanças que já estavam no projeto anterior. Ele voltou a mostrar erros, embora os materiais agora não aparentavam mais estar com problema de renderização. Vale notar que um dos problemas que tanto o projeto inicial quanto o ARTemplate da UE4.25 estavam apresentando era o de que a câmera parecia ficar muito escura, com iluminação quebrada. Esse problema parece ter se amenizado ao se utilizar a versão 4.26p1.

Após mais depuração, encontramos alguns dos causadores dos erros relatados:

A imagem a seguir mostra que o SkeletalMesh é a raíz dos componentes do ator ActionFigure. Na segunda, de acordo com como estava no ARTemplate, o SkeletalMesh, ao invés de ser o root do ator, ficou como filho do SceneComponent.

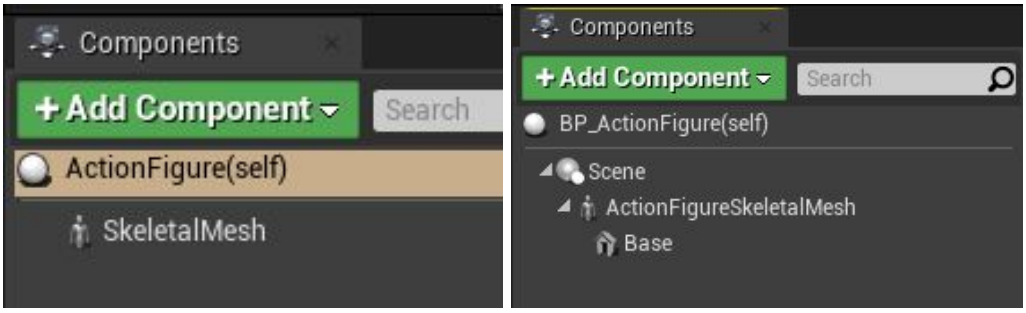


Figura 12: Hierarquia do ator que seria o ActionFigure no projeto original, antes e depois da correção.

No projeto inicial, o skeletal mesh dos assets do InfinityBlade eram as raízes dos componentes, enquanto no ARTemplate eram filhos de um Scene component, que é o root. Fazendo com que o Skeletal Mesh fosse filho do Scene também pareceu solucionar os problemas.

Fora isso, foi implementado um dropdown menu com a funcionalidade de mudar o skeletal mesh do action figure (para testar mudar o modelo) e um botão de X no canto superior direito da tela para a pessoa poder detectar outros planos, reiniciando o processo.

Figura 13: Layout básico para testar funcionalidades do AR

4.3. Uso de Gestos

No mês de Setembro, no marketplace, o plug-in Ultimate Touch Components (UTC), que fornece uma base para funcionalidades quanto ao touch do celular, como detectar swipes, pinch etc, foi disponibilizado gratuitamente.

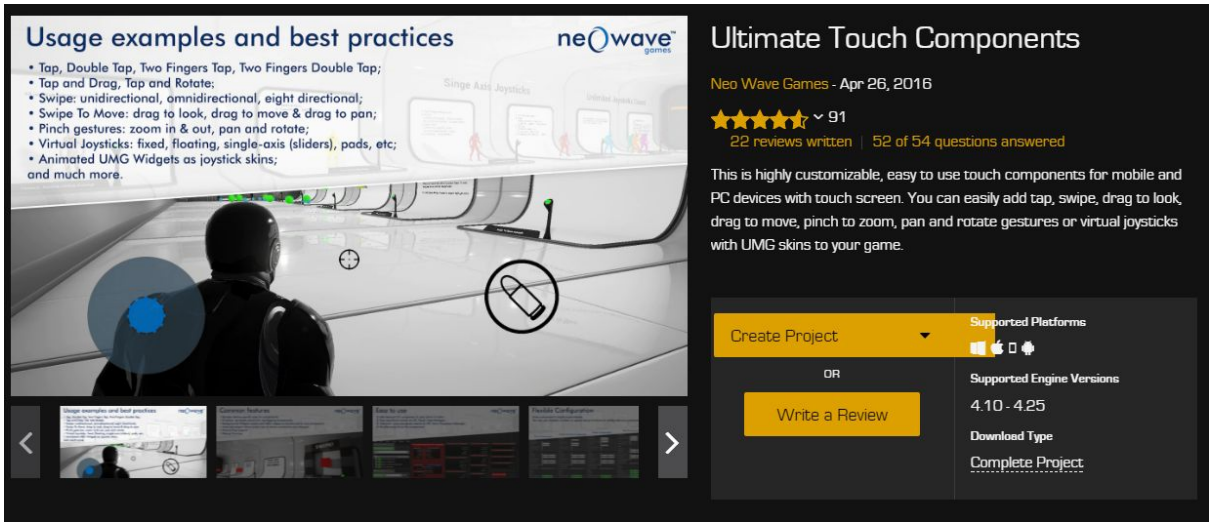


Figura 14: Plug-in para reconhecimento de gestos em touch-screen.

Foi feito um estudo de seu código e o que poderia ser trazido para o projeto, com o intuito de utilizar esses movimentos detectados nas operações de movimentação, rotação e escala do ActionFigure.

Após avaliar seu funcionamento e realizar testes para validar sua adaptação projeto, conseguiu-se de fato incorporá-lo junto ao protótipo.

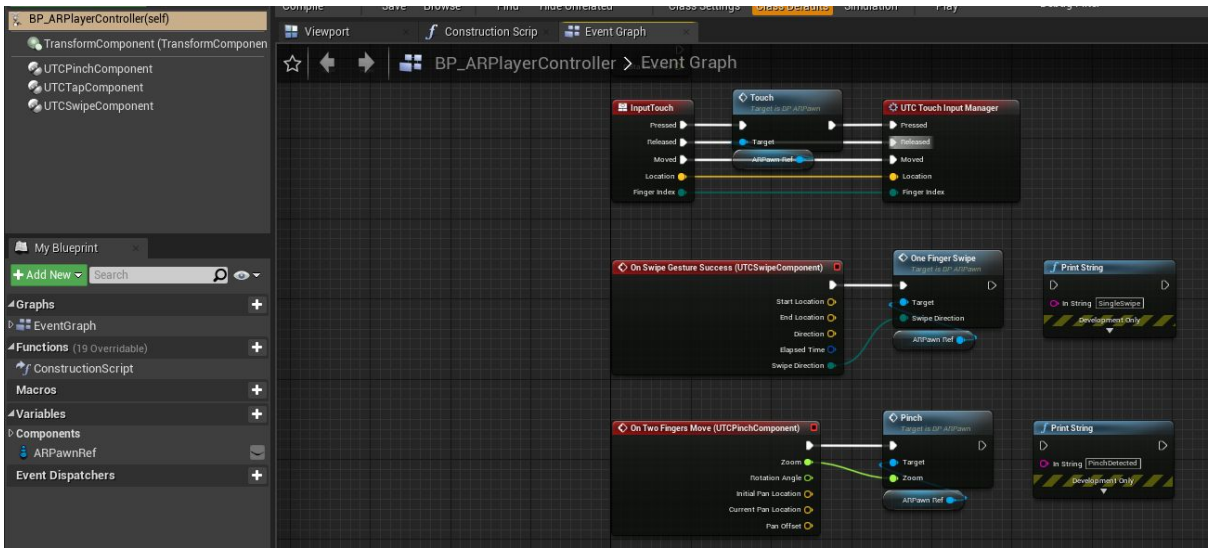


Figura 15: Parte do código do PlayerController, com integração do código do UTC

Assim, realizar o movimento de pinch ativa a funcionalidade de zoom da action figure e o swipe a de rotação.

4.4. Marketplace

4.4.1. Implementação da listagem

Conforme explicado na seção 4.1.2 deste relatório, como o aplicativo possui uma função de venda de *skins*, optou-se por apresentar os diferentes modelos em uma lista no estilo *marketplace*, utilizando, inclusive, padrões de interação típicos deste modelo de interface. Tal decisão foi tomada, pois considerou-se que adicionar um mecanismo de controle por AR à lista de seleção de *skins* não seria uma solução amigável ao usuário e, portanto, o aplicativo teria mais sucesso utilizando padrões que o usuário já está acostumado no dia-a-dia.

A implementação da página do marketplace envolve a criação de uma interface de usuário que se sobrepõe à imagem da câmera. Ao clicar nos diferentes itens apresentados, a interface deve gerar eventos que acionam o mecanismo de instanciação de modelos AR, trocando a *skin* renderizada, além de esconder a página do marketplace.

Uma segunda funcionalidade importante é a escalabilidade na população da lista de *skins*, ou seja, popular a listagem com múltiplos modelos deve ser um processo simples e, na medida que mais *skins* são adicionadas, elas devem estar presentes automaticamente no marketplace, que deve ser uma interface *scrollable*, permitindo que todas as *skins* sejam analisadas.

A primeira etapa foi a criação de um Widget de UI para o menu do marketplace. A este widget, foi adicionado uma caixa de texto e um botão, representando a caixa de busca, e uma caixa de rolagem, que é um painel *scrollable* capaz de apresentar até 100 *widgets* diferentes. Esta caixa de rolagem será populada com os itens de cada *skin*.

Para que este widget apareça no aplicativo, foi adicionado a funcionalidade de troca de widgets ao clicar no botão X na interface AR, trocando o índice do widget ativo para 1 (referente à interface do marketplace).

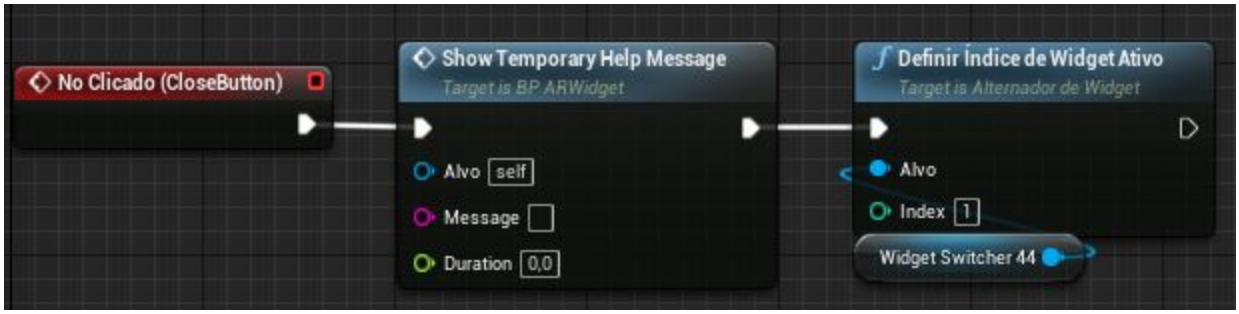


Figura 17: Blueprint para troca de widgets acionada pelo botão “Close”

Tal alteração de widgets iniciará a construção da página do marketplace, que deve ser populada com uma lista de modelos incluídos no projeto. Para que o processo de população seja o mais dinâmico possível, foi criada uma tabela de dados (*Data Table*) contendo os dados de cada *skin* a ser apresentada. No momento de construção do marketplace, a tabela é consultada e a lista é populada de acordo.

Para a criação da tabela, foi criado um modelo, chamado ST_MarketPlaceModel, contendo todos os campos que serão mostrados na lista e seus tipos: a malha esquelética e a textura da *skin*, o nome e a mídia de origem do personagem, uma imagem de *preview* do modelo e o preço do produto. Com o modelo feito, foi criada a *datatable DT_MarketPlaceModel*, arquivo no qual podem ser adicionados os dados de cada *skin.*

Figura 18: Modelo utilizado para popular a tabela de dados (acima) e tabela com os dados referentes a três *skins* (abaixo).

Figura 19: Loop principal de *SetMarketPlaceAvailableModels*. Além deste loop, a função apaga o *grid* anterior e faz uma checagem para evitar de modelos repetidos

Para cada item na tabela, é chamada a função *AddModelToGrid*. Essa função tem como objetivo instanciar e popular cada um dos *widgets* que serão apresentados dentro da caixa de rolagem do marketplace. Para isso, ela foi dividida em três partes. Inicialmente, o *widget* é instanciado, de forma a manter duas colunas de itens:

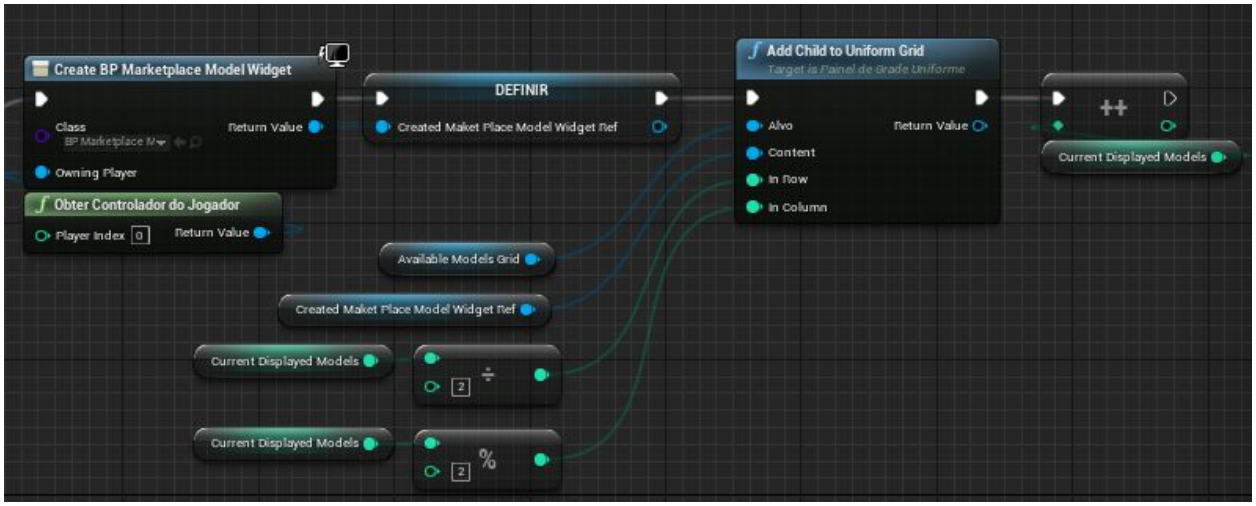


Figura 20: Blueprint para criação de widget dentro da barra de rolagem

Note que o widget instanciado é do tipo *BP_MarketplaceModelWidget*, que possui a seguinte estrutura.

Figura 21: Estrutura básica do widget para seleção de *skin*

A segunda parte da função *AddModelToGrid* é responsável por adicionar os valores específicos ao *widget* e seu esquema em blueprint está apresentado a seguir. Seu funcionamento é simples: primeiramente obtém-se uma referência ao modelo que está sendo analisado, quera-o nos seus diferentes componentes e os dados são populados (nome, imagem, preço e mídia de origem). Além disso, configura-se o valor de *skeletal mesh* do widget para o do modelo. Esse valor será usado na hora de instanciar o modelo em ambiente AR.

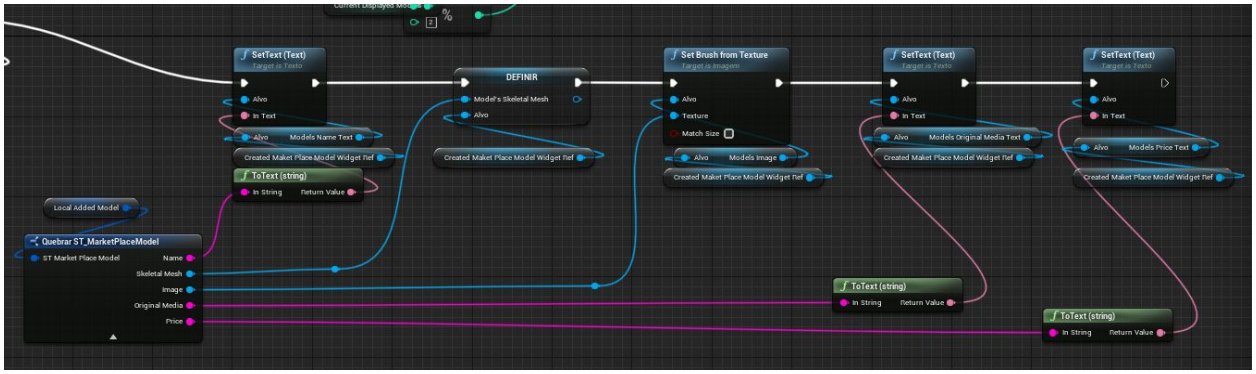


Figura 22: Blueprint para substituição de valores no *widget* do marketplace

A terceira parte da função *AddModelToGrid* é responsável por vincular um evento ao clique do widget, ou seja, quando o item da lista for clicado, ele gerará o evento *HandleModelSelected*, que irá trocar o modelo instanciado em AR.

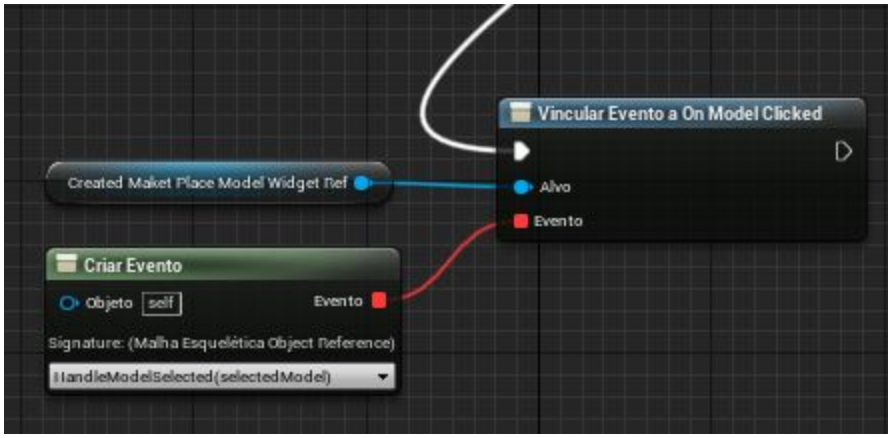


Figura 23: Vinculação de evento ao clique do *widget.*

Após a implementação dos códigos acima e a adição de três modelos à tabela de dados, o marketplace ficou com a seguinte aparência:

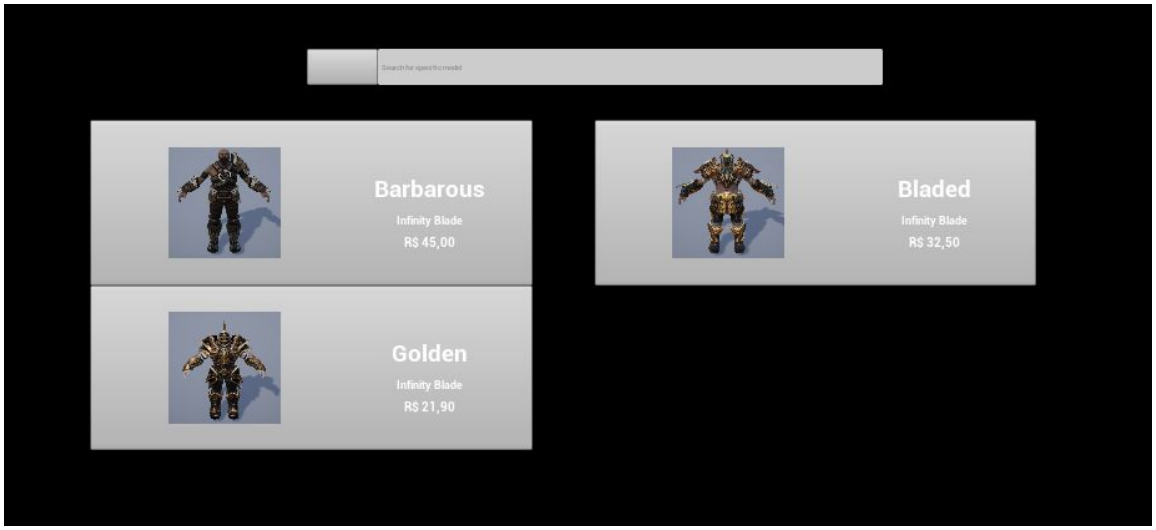


Figura 24: Primeira versão do marketplace

4.4.2. Funcionalidade de filtragem

Outra funcionalidade muito comum em marketplaces é a capacidade de filtragem dos resultados. Para isso, decidiu-se implementá-la, primeiramente, modificando a parte gráfica ao adicionar uma caixa de texto em que o usuário pode digitar o nome de um personagem específico e um botão para indicar ao aplicativo que realize tal filtragem.

Após a modificação gráfica, foi necessário criar a funcionalidade. Para isso, foi criada a seguinte blueprint que é acionada ao apertar o botão da barra de busca. Tal código utiliza a função *SetMarketPlaceAvailableModels*, que foi alterada para receber o texto da barra de busca como parâmetro.

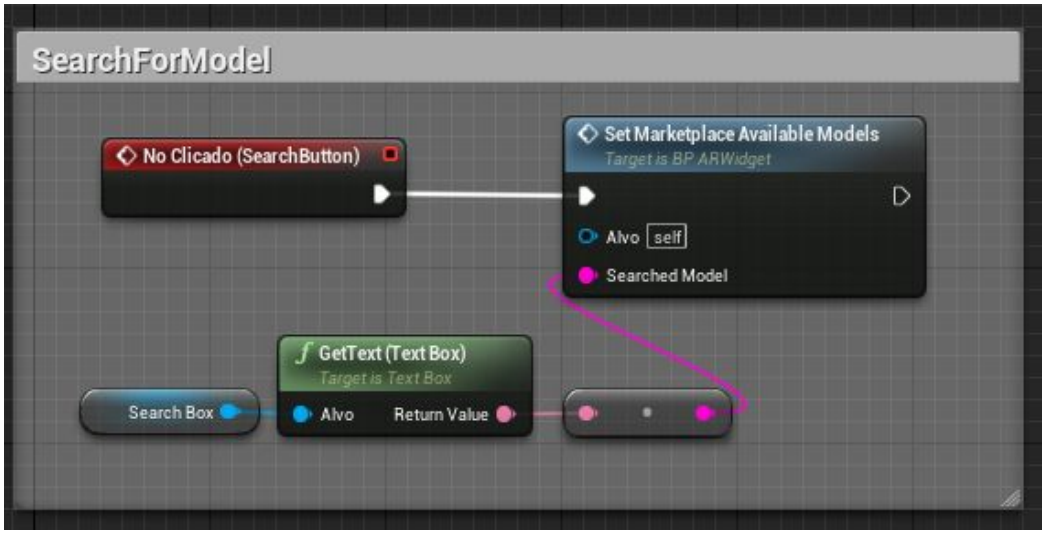


Figura 25: Chamada de *SetMarketPlaceAvailableModels* no caso de busca

A alteração feita em *SetMarketPlaceAvailableModels* é uma checagem do parâmetro *SearchedModel*. Se ele for uma string vazia, então a caixa de rolagem é populada com todos os itens da tabela de *skins*. Caso contrário, é feita a verificação se o modelo contém a *substring* recebida como parâmetro.

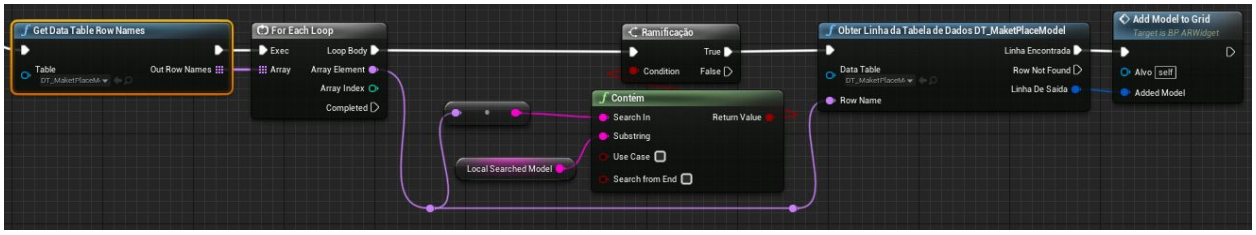


Figura 26: Loop executado por *SetMarketPlaceAvailableModels* no caso de busca

Com isso, a funcionalidade de filtragem foi implementada, como se pode ver na imagem abaixo. Nela, a barra de busca contém um “B”, e portanto a skin do personagem *Golden* não deve ser mostrada.

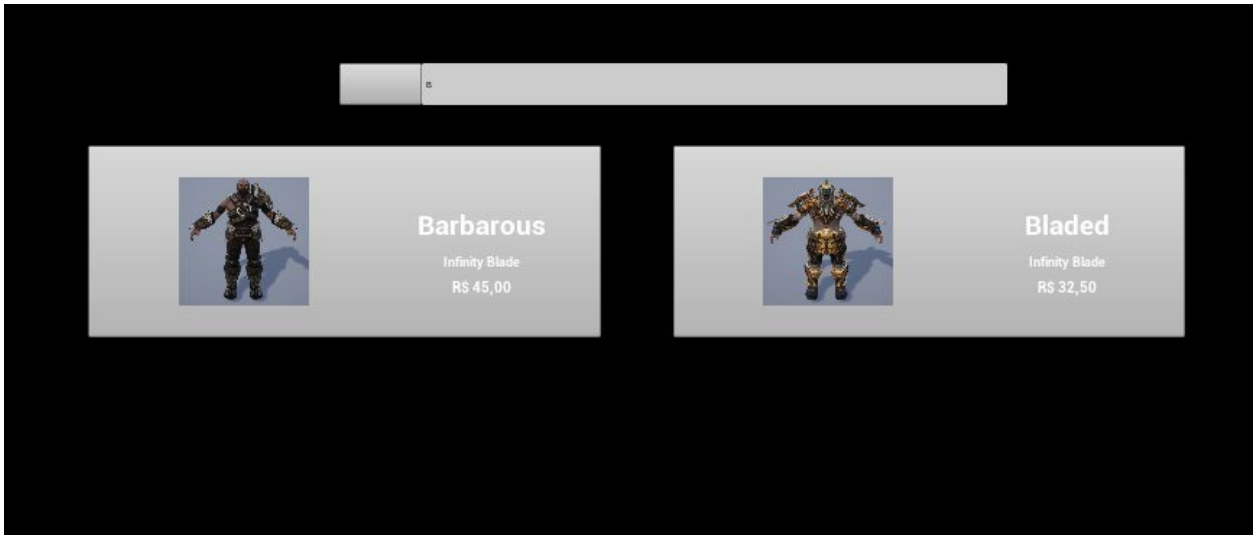
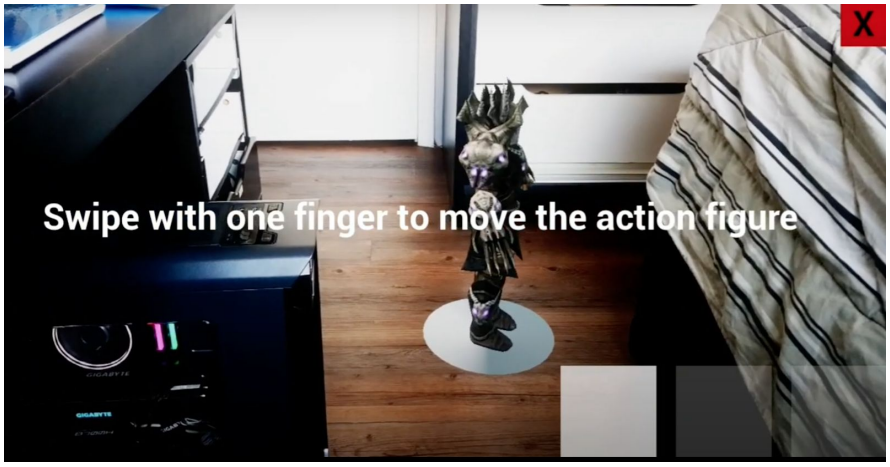
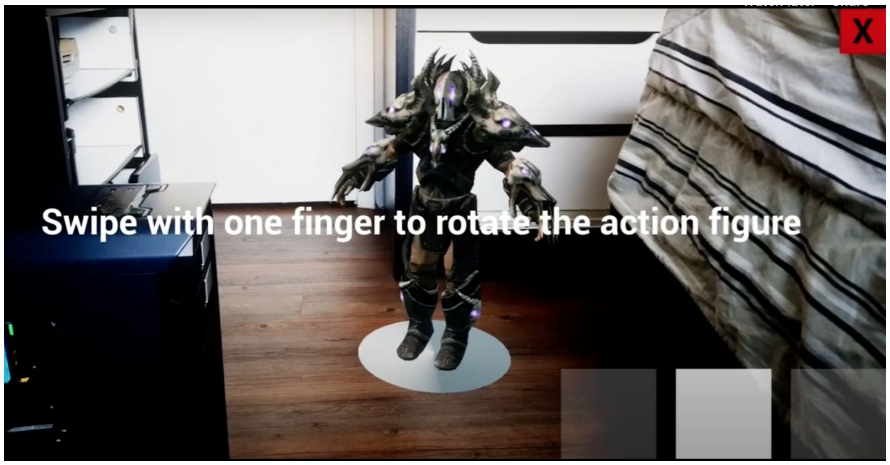
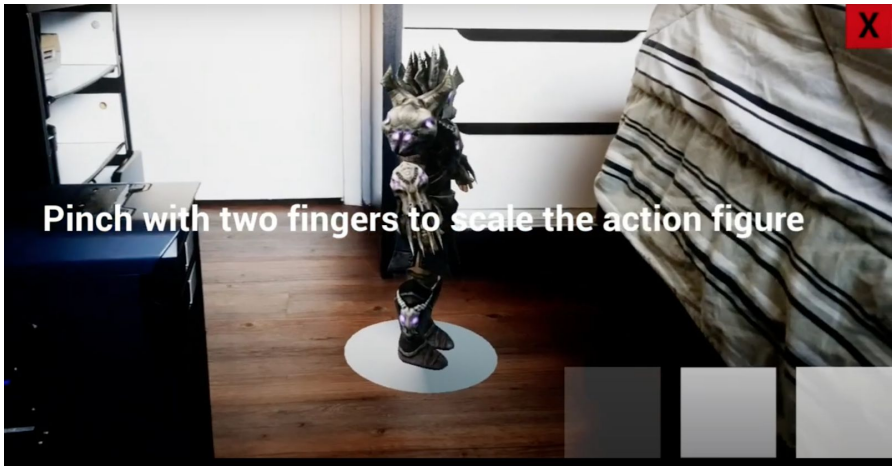


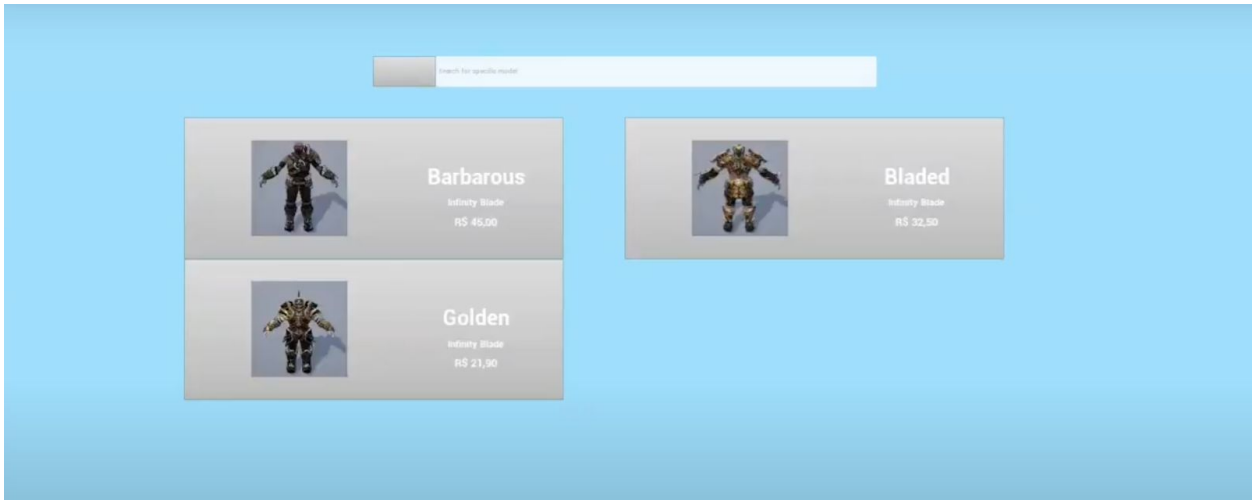
Figura 27: Resultado da implementação da filtragem

5. Resultados


X
Swipe with one finger to move the action figure


X
Swipe with one finger to rotate the action figure


X
Pinch with two fingers to scale the action figure


Barbarous
Infinity Blade
R$ 45,00
Bladed
Infinity Blade
R$ 32,50
Golden
Infinity Blade
R$ 21,90

6. Testes

Referências

[1] QUINTELA, Samuel. **Mercado Geek deve movimentar até R$ 40 milhões durante as férias.** Diário do Nordeste, [*s. l.*], 10 jul. 2019. Disponível em: https://diariodonordeste.verdesmares.com.br/negocios/mercado-geek-deve-movimentar-ate-r-40-milhoes-durante-as-ferias-1.2121816. Acesso em: 14/09/2020.

[2] BLIZZARD. **Tracer Overwatch 10.5" Premium Statue. *In*: Blizzard Gear Store**. Disponível em: https://www.blizzardgearstore.com/games/overwatch/tracer-overwatch-105-premium-statue/o-21056808+t-54729145+p-69821605+z-8-3288737785?_ref=p-PDP:m-CAV:i-r0c0:po-0. Acesso em: 19/09/2020.

[3] GOODLEY, Alvin. **12 Rarest League of Legend Skins Ever Released**. Rarest.org, [*s. l.*], 27 out. 2019. Disponível em: https://rarest.org/entertainment/league-of-legend-skins. Acesso em: 19/09/2020.

[4] WINKIE, Luke. **9 of the rarest League of Legends skins**. PCGamer, [*s. l.*], 19 nov. 2018. Disponível em: https://www.pcgamer.com/lol-skins/. Acesso em: 14/09/2020.

[5] EPIC GAMES. **Paragon: Narbash. *In*: Unreal Engine Marketplace. [*S. l.*]**. Disponível em: https://www.unrealengine.com/marketplace/en-US/product/paragon-narbash. Acesso em: 14/09/2020.

[6] Relatório SEBRAE; **Mercado de action figures: oportunidades para empreender.** Disponível em: https://atendimento.sebrae-sc.com.br/inteligencia/relatorio-de-inteligencia/mercAdo]-de-action-figures-oportunidades-para-empreender. Acesso em 19/09/2020.

[7] EDUARDO, Sammy. **Conheça o fantástico (e lucrativo) mundo das franquias geek. ABF Portal do Franchising, [*s. l.*]**, 9 jan. 2018. Disponível em: https://www.portaldofranchising.com.br/franquias/franquias-geek-nerd/. Acesso em: 14/09/2020.

[8] EPIC GAMES. Página promocional dos modelos de Paragon, disponíveis para Unreal Engine 4. Disponível em: https://www.unrealengine.com/en-US/paragon. Acesso em 28/09/2020.

[9] Michael Pavlovich. **Loading the Free Paragon Assets in Unreal Engine 4**. Disponível em: https://www.youtube.com/watch?v=0UUdhCfpj2M. Acesso em: 28/09/2020

[10] Unreal Engine. **Quickstart AR.** Disponível em: https://docs.unrealengine.com/en-US/Platforms/AR/HandheldAR/ARHowToHitTesting/index.html
Acesso em: 07/10/2020