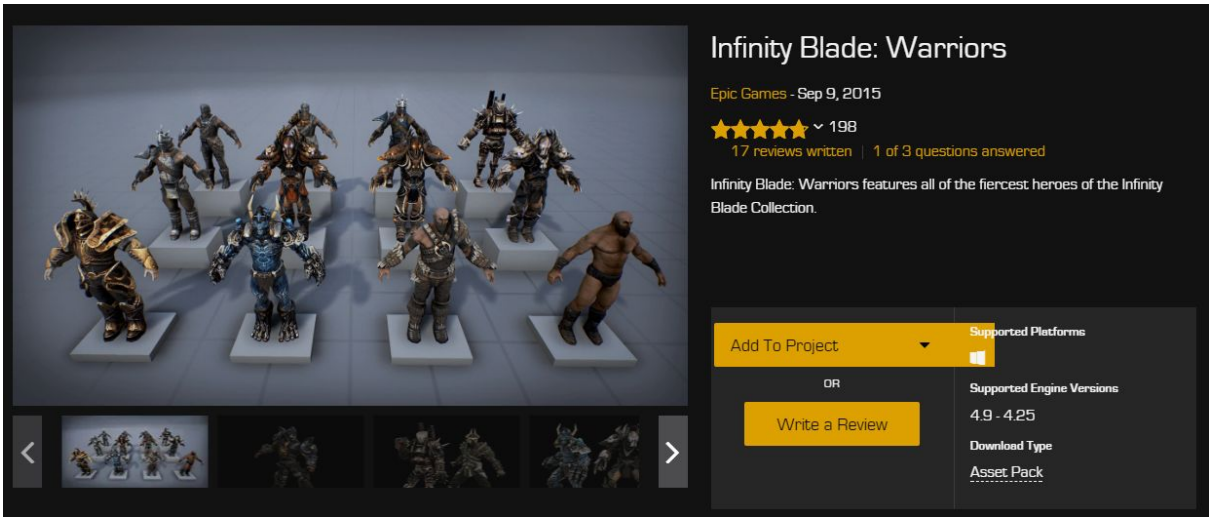
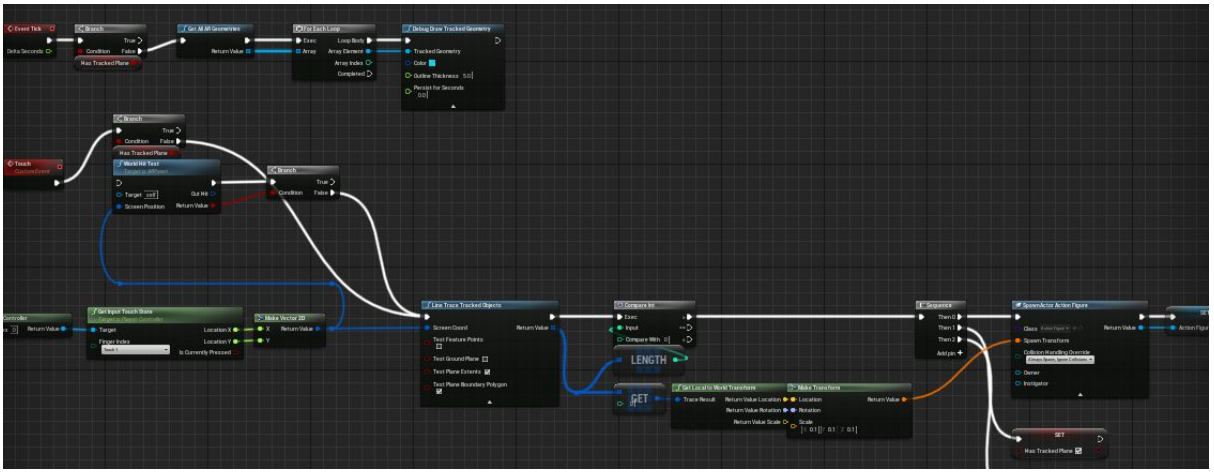
Foi observado que os modelos dos assets do Paragon estão muito grandes, que é de se esperar de assets AAA para um jogo para PC/PS4, que não precisa se preocupar tanto com otimização de tamanho de arquivos e memória quanto aplicações para celular.
Sendo assim, optou-se que seriam utilizados, a priori, alguns assets do jogo Infinity Blade (que é um jogo mobile, então deve já apresentar um pouco dos cuidados para não serem muito grandes), que também podem ser obtidos de graça no marketplace.



Foram então adicionados dentro do projeto.

Utilizou-se a estrutura semelhante ao que é mostrado no quickstart da Unreal para o ARCore, que pode ser encontrado em <https://docs.unrealengine.com/en-US/Platforms/AR/HandheldAR/ARHowToHitTesting/index.html> para se fazer a estrutura básica de um projeto com AR para poder detectar planos e então instanciar um objeto no ponto do plano em que a pessoa clicar.
A seguir imagem de parte do código do projeto inicial, com código semelhante ao encontrado no link mencionado



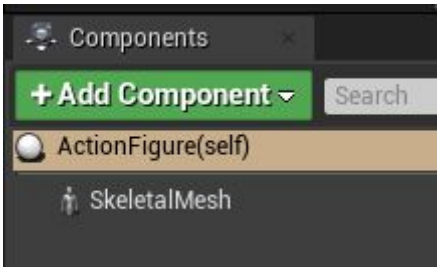
Todavia, quando foram feitos testes iniciais, além do AR não estar conseguindo lidar bem com a pessoa se mover no mundo real e manter o track de onde cada objeto deveria estar posicionado no mundo, o objeto não parecia estar sendo instanciado corretamente no local

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.

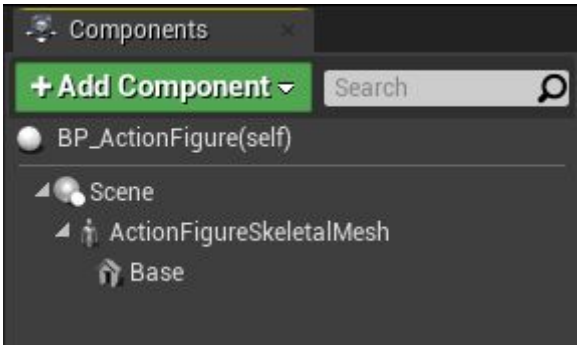
Sabe-se que isso não é erro da Engine (4.25 e 4.26p1) pois foram feitos testes com o ARTemplate dessas versões e esses problemas não estavam tão presentes e graves.

Foi feita uma comparação de quais configurações poderiam estar diferentes entre os templates e o projeto do grupo, na tentativa de sanar esse problema.
Não se chegou a uma conclusão de qual seria o problema. Após realizar mudanças no projeto de possíveis configurações que dariam problema para tentar deixá-lo parecido com o template, chegou-se na abordagem de, ao invés de fazer um projeto do 0, pegar o ARTemplate e usar de base para o projeto.

Assim foi feito, e foram adicionadas gradualmente as mudanças que já estavam no projeto para o template. Ele voltou a mostrar erros como instanciar o objeto no local clicado e manter o track dele no mundo real. Todavia, 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 4.26p1.
Após mais depuração, mudou-se uma configuração na BluePrint do ActionFigure que não se imaginava que daria problema mas que fez o projeto parar de apresentar esses problemas:
A primeira imagem 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.



Hierarquia do ator que seria o ActionFigure no projeto original, que não estava funcionando

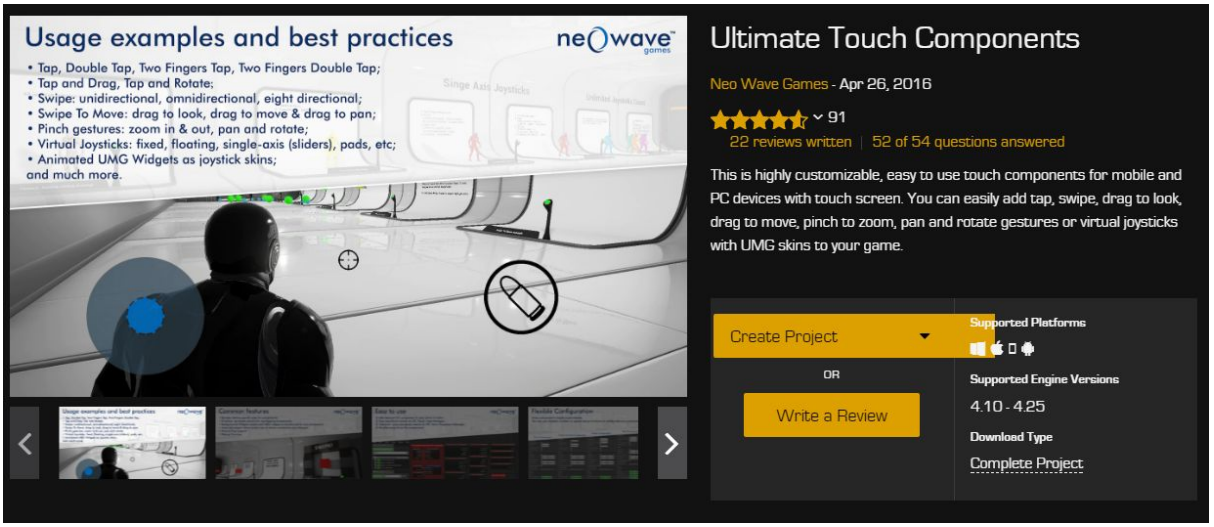


Hierarquia do ator que seria o ActionFigure no projeto baseado no ARTemplate, que está funcionando

No projeto inicial, o skeletal mesh dos assets do InfinityBlade estavam como sendo a raíz dos componentes, enquanto que no ARTemplate eles são filhos de um Scene component, que é o root.
Fazendo que o Skeletal Mesh fosse filho do Scene também pareceu solucionar os problemas.

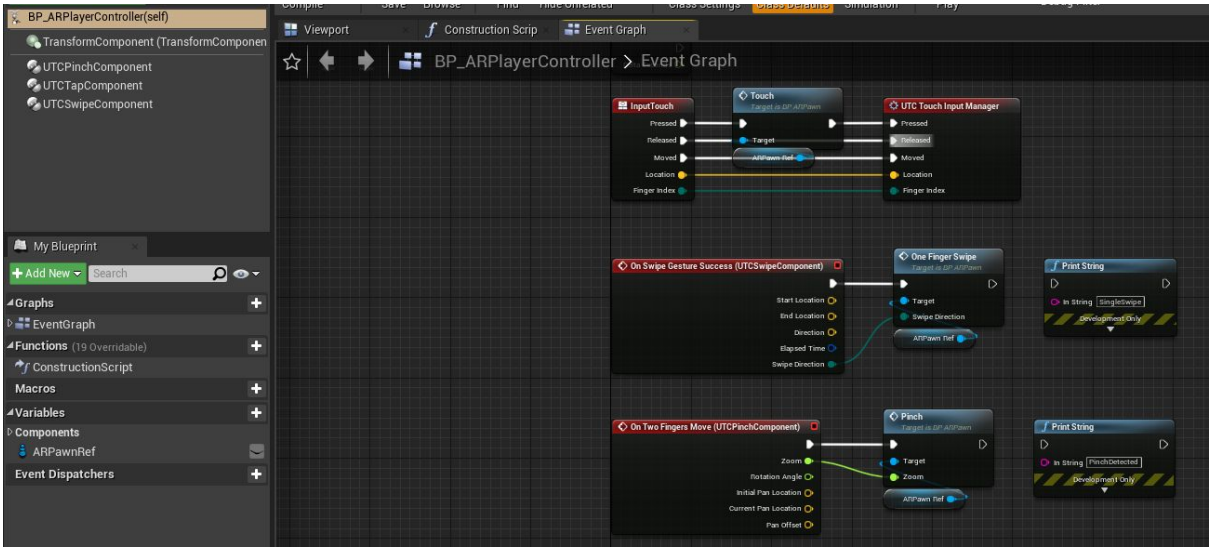
Fora isso, foi implementado um dropdown menu com a funcionalidade de mudar o skeletal mesh do action figure (para testar mudar o modelo) e um botão de X no canto superior direito da tela para a pessoa poder detectar outros planos, iniciando o processo do 0.

Em paralelo a tudo isso, um produto que ficou disponível de graça no mês de Setembro no marketplace foi o Ultimate Touch Components (UTC), que fornece uma base para funcionalidades quanto ao touch do celular, como detectar swipes, pinch etc.



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 estudar seu código e funcionamento, e de ter feito testes para ver se seria possível adaptá-lo ao projeto, conseguiu-se de fato incorporá-lo junto nesse protótipo:



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.
Vale notar que eles foram adicionados também para testar o quão possível é de ser utilizado. Para uma entrega final, ajustes e modificações teriam que ser feitos neles, além do modo como a seleção do modelo sendo renderizado é feita.

Ademais, o git ignore para a Unreal Engine utilizado para o projeto foi retirado de https://github.com/github/gitignore/blob/master/UnrealEngine.gitignore