SPACE ROYALE: DESENVOLVIMENTO DE JOGO BATTLE ROYALE COM UNREAL ENGINE

Aluna: Sara Helena Régis Theiss da Silva

Orientadora: Luciana Pereira de Araújo Kohler

FURB
UNIVERSIDADE DE BLUMENAU

Roteiro

- Introdução
- Objetivo geral e objetivos específicos
- Fundamentação teórica
- Trabalhos correlatos
- Requisitos funcionais e não funcionais
- Especificação: diagrama de casos de uso e diagrama de componentes
- Implementação
- Análise dos resultados
- Conclusões e sugestões

Introdução

- Os primeiros jogos multiplayer utilizavam divisão na tela;
- Depois passaram a utilizar consoles separados e conectados por uma rede local (LAN);
- Em 1999 uma futura vertente dos jogos multiplayer ganhava forma com a obra Battle Royale;
- Jogos Battle Royale vem ganhando muita popularidade.

Introdução

- Para o jogo Space Royale foi combinada a popularidade de jogos Battle Royale com o promissor motor de jogos Unreal Engine;
- É uma batalha com naves espaciais, na qual os jogadores se enfrentam com dois tipos de arma: básica e especial;
- A arma especial possui 4 categorias: Ataque, Bomba, Lentidão e Escudo.

Objetivos

Objetivo geral:

- Disponibilizar um jogo do gênero Battle Royale que permita a interação entre jogadores de diferentes lugares utilizando o motor de jogos Unreal Engine.

Objetivos

Objetivos específicos:

- Disponibilizar um jogo de batalha com naves em que os usuários lutarão entre si para que apenas um seja o vencedor;
- Criar armas categorizadas entre armas de ataque, bombas, armas de lentidão e escudos;
- Fornecer um jogo com boa jogabilidade.

Fundamentação Teórica

Battle Royale

- A obra polêmica de Koushun Takami tornou o termo muito comum quando adaptada para os cinemas nos anos 2000.
- Começou a ser utilizado para descrever jogos nos quais os jogadores enfrentam uns aos outros até que reste apenas um.
- Para suprir o objetivo final que é a sobrevivência, faz uso de armas, habilidades e estratégias, acarretando em um resultado imprevisível.

Fundamentação Teórica

Jogos Multiplayer

- Permitem a interação competitiva ou colaborativa entre jogadores em uma mesma partida;
- Os jogadores podem enfrentar inimigos mais desafiadores, pois usam artifícios como a inteligência, espontaneidade e intuição de um ser humano;

Trabalhos Correlatos

Batalha Naval

- Simulador de batalha naval individual ou em duplas com 3 modos: clássico, tiro extra ao acertar, e atire tantas vezes enquanto tiver navios.
- Cada jogador começa com cinco navios dentre: porta-aviões, *destroyer*, cruzador, submarino e barco patrulha. Eles são abatidos, respectivamente, com 5 tiros, 4 tiros, 4 tiros, 3 tiros e 2 tiros;
- Os jogadores posicionam suas embarcações e o jogo pode ser iniciado. Cada um deles tenta encontrar os navios inimigos e ganha quando afunda todos eles.

Trabalhos Correlatos

Planeta X

- Simulador de nave que tem por objetivo encontrar e analisar pistas sobre o que aconteceu aos colonizadores do Planeta X;
- Oferece recursos, como objetos no mapa, para que o jogador descubra o mistério por meio de raciocínio;
- Possui duas fases: a primeira é dentro de uma nave e o jogador precisa descobrir como liga-la; na segunda, a nave leva o jogador ao planeta que deve ser explorado atrás de pistas.

Trabalhos Correlatos

Fawe

- Um protótipo de MMORPG, no qual o jogador tem por objetivo salvar os remanescentes e seguir para as maiores cidades dos 3 continentes mais seguros e desenvolvidos;
- No mundo do jogo, para evitar sua destruição completa, os deuses fizeram que, dentre os 12 continentes, sempre ficassem 3 deles ocultos;
- Em um tempo determinado, os 3 continentes mais destruídos pelos humanos são engolidos e 3 novos emergem renovados.
- O jogador inicia em um desses continentes destruídos, Finraza, e deve resgatar os que restaram e seguir para um continente seguro.

Requisitos

Requisitos Funcionais:

RF01: O jogo deve permitir que o usuário crie um lobby e dê início à partida.

RF02: O jogo deve permitir que o usuário entre em um lobby existente.

RF03: O jogo deve apresentar os números de jogadores aguardando na sala antes de uma partida.

RF07: O jogo deve permitir que o usuário mova sua nave no mapa.

RF08: O jogo deve permitir que o usuário ative suas armas disponíveis.

RF09: O jogo deve aplicar dano à nave do jogador quando atingido por um disparo, de acordo com o tipo do disparo.

RF12: O jogo deve apresentar os resultados da partida quando esta acaba para o jogador.

Requisitos

Requisitos Não Funcionais:

RNF01: O jogo deve ser liberado para plataforma desktop.

RNF02: O jogo deve ser desenvolvido no Unreal Engine 4.

RNF03: O tratamento dos atributos da Nave a das Armas deverá ser feito por meio da linguagem de programação C++.

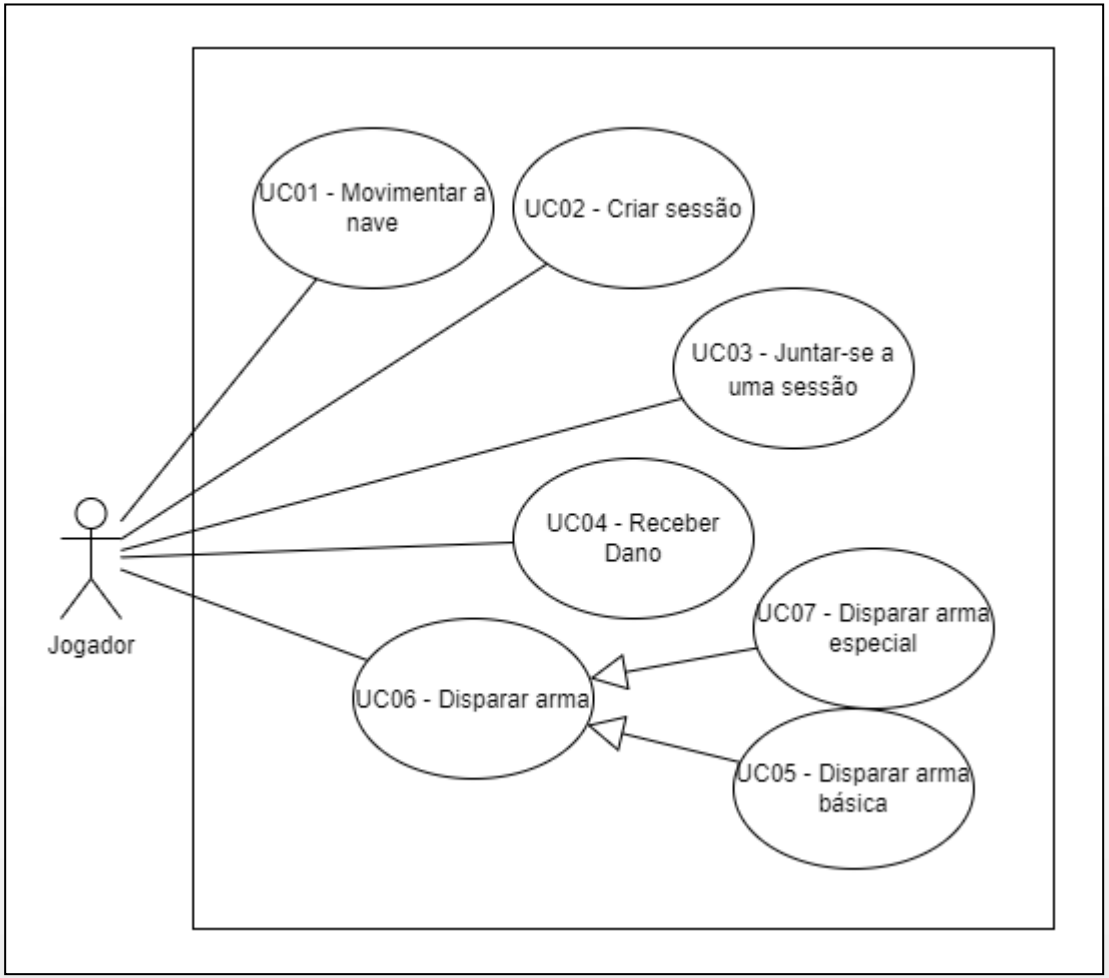
RNF04: A relação dos atributos com a interface gráfica deverá ser feita por meio da linguagem de script visual Blueprints.

RNF05: A nave do jogo deve ser modelada pela ferramenta Blender.

RNF06: A nave deve possuir uma coloração neutra e possuir duas saídas para as armas básicas e especiais.

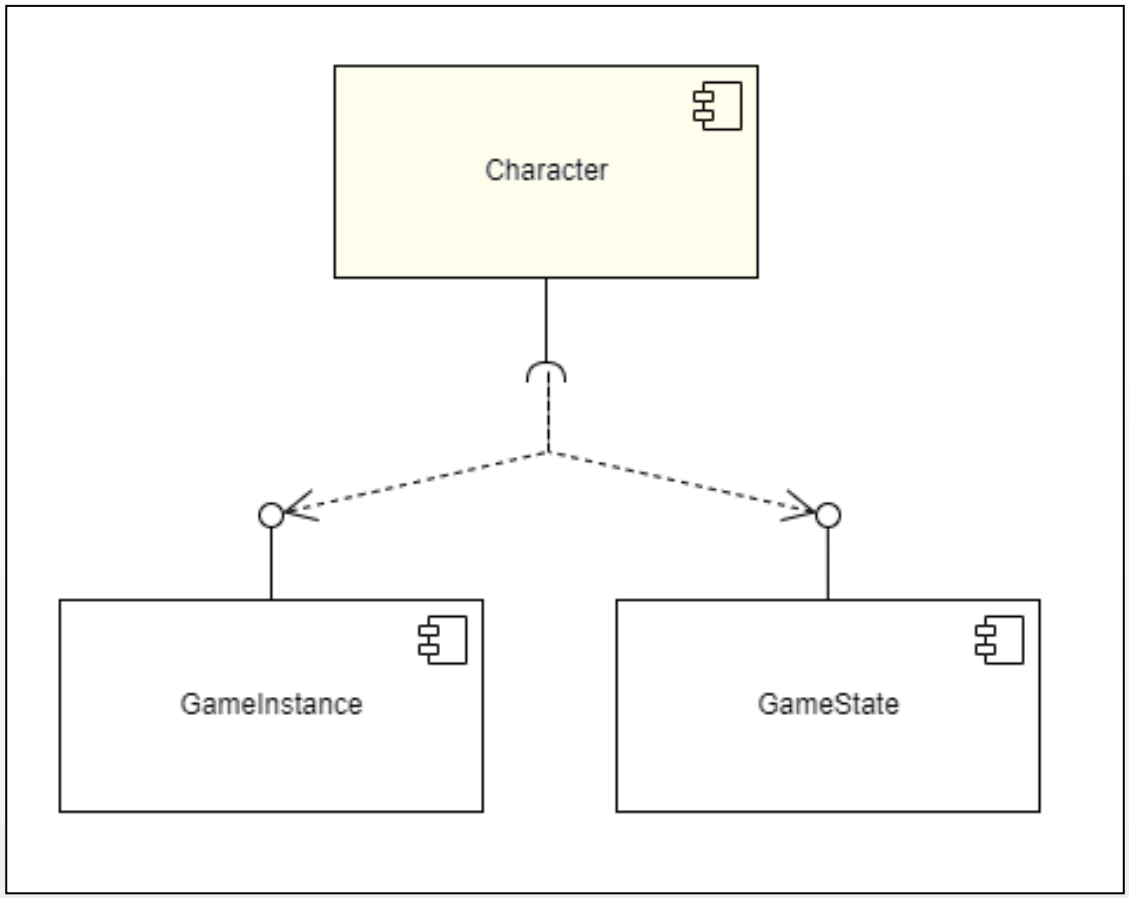
Especificação

Diagrama de casos de uso:



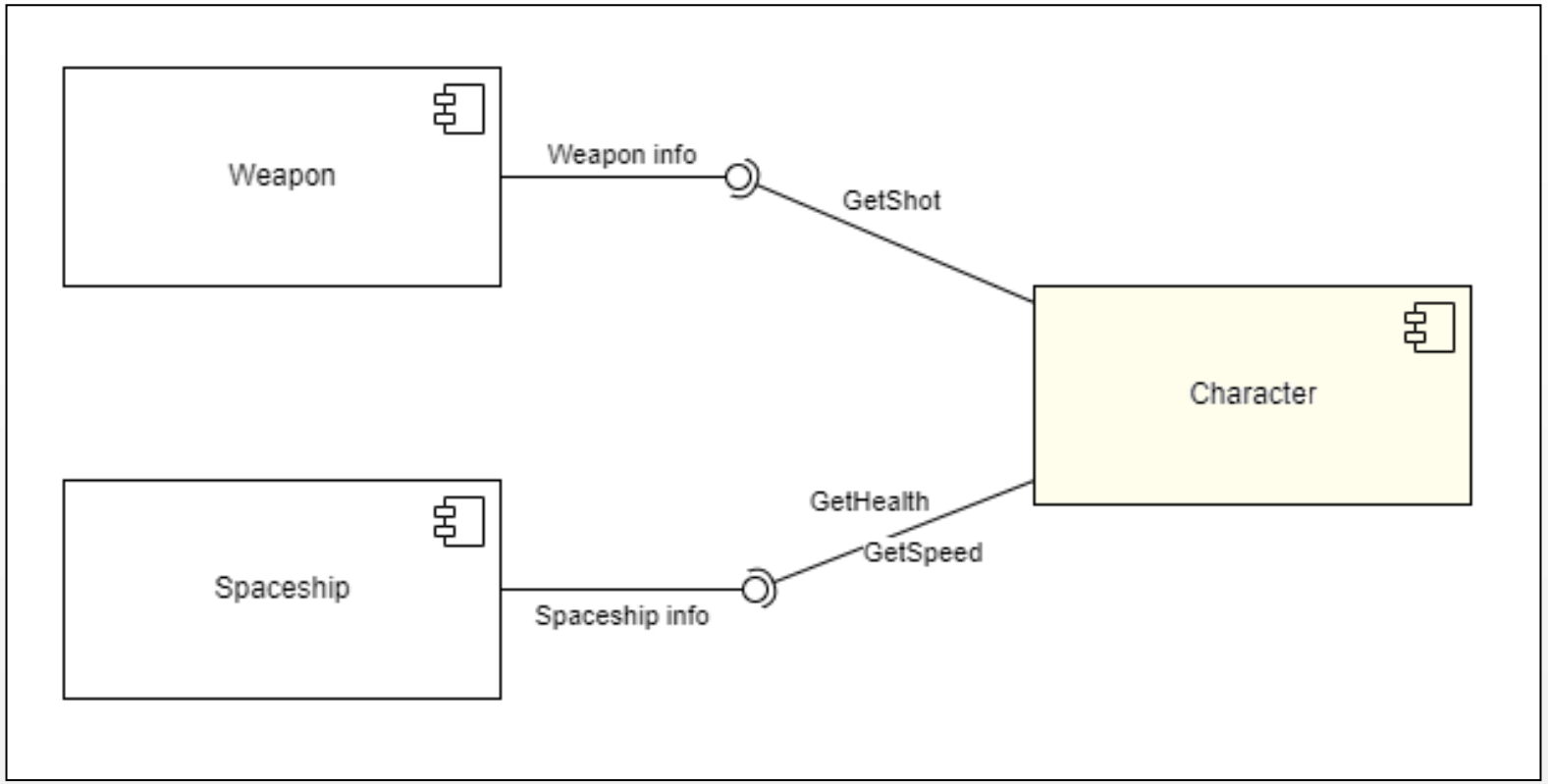
Especificação

Diagrama de componentes:



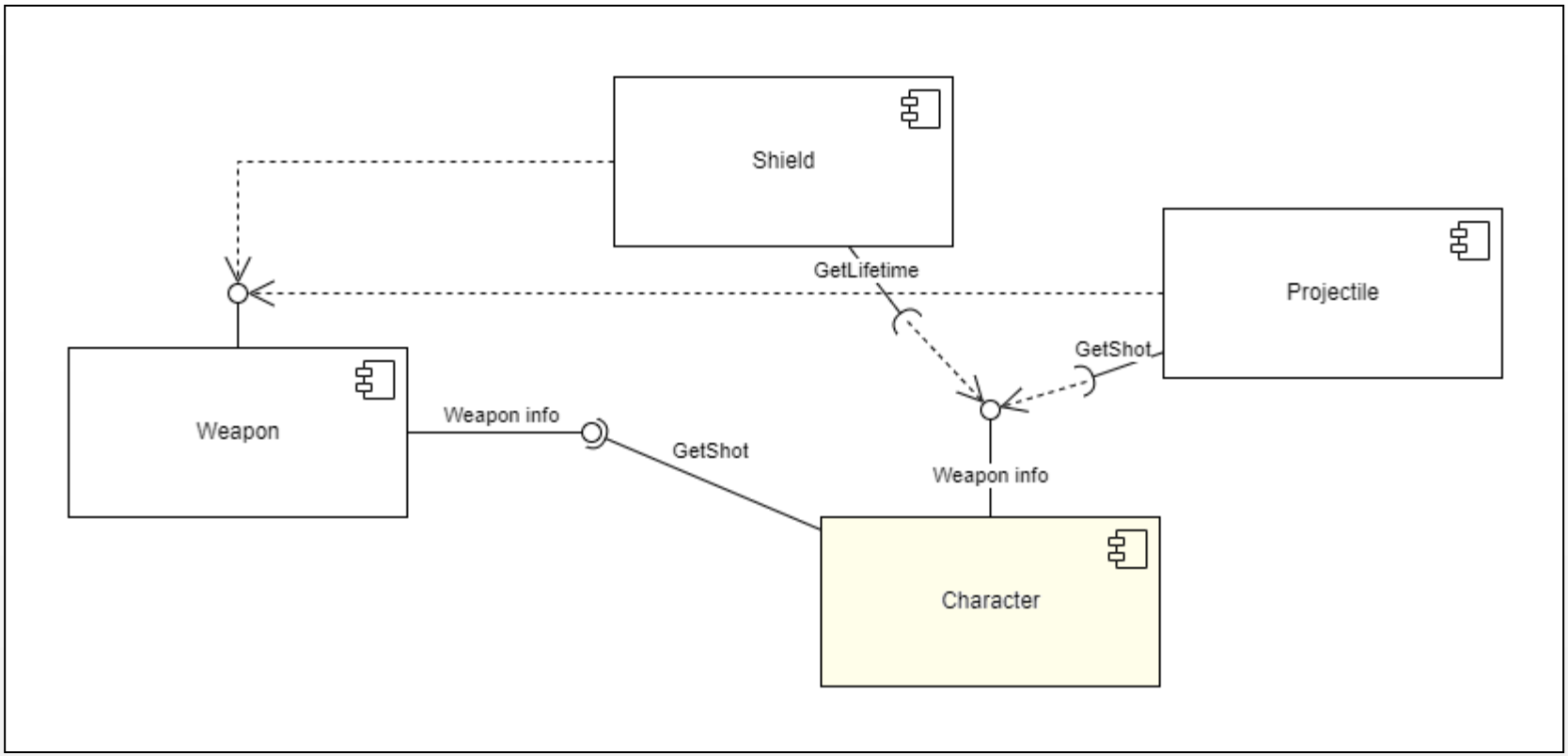
Especificação

Diagrama de componentes:



Especificação

Diagrama de componentes:

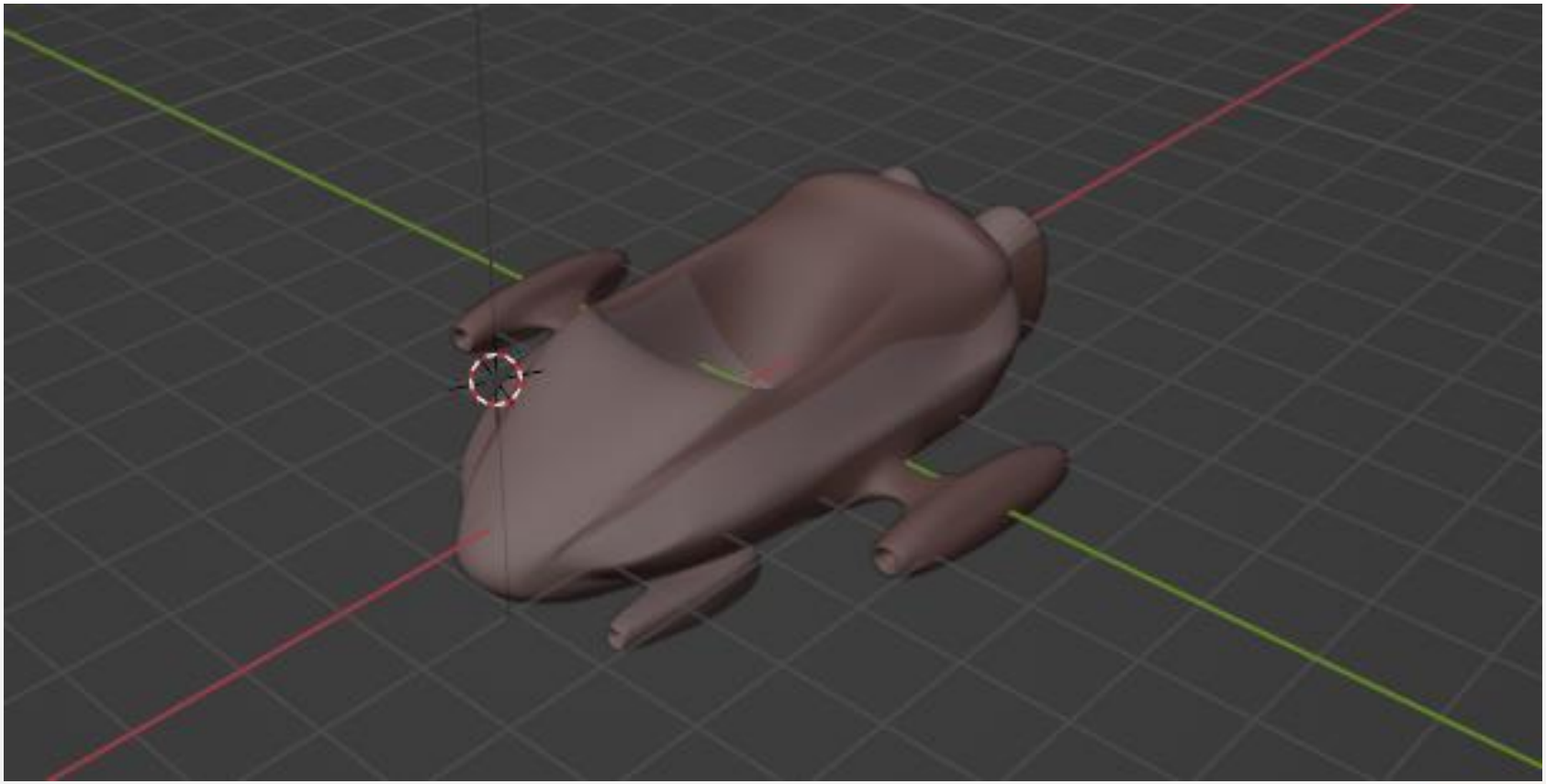


Implementação

- Utilizando o motor de jogos Unreal Engine, o jogo foi escrito em C++ em conjunto com Blueprints;
- A nave do jogo foi modelada no Blender a partir de um cubo, utilizando recursos como espelhar e subdividir. Também foi esculpida a lataria da nave e aplicada uma coloração a ela. A partir do plugin *"Pipeline: Send to Unreal"* o objeto 3d pode ser enviado diretamente para dentro de um projeto do Unreal Engine;
- Com base em um projeto padrão de naves foi iniciado o desenvolvimento do jogo, sendo necessária a criação de um novo mapa e alteração das características da nave exemplo agora com o novo modelo.

Implementação

Nave do jogo modelada no Blender:



Implementação

- Para poder criar sessões e tornar o jogo multiplayer é utilizado o componente `GameInstance` e, criando uma classe C++ filha desse componente, puderam ser feitos os tratamentos necessários para chamar os métodos `CreateSession`, `FindSessions`, `JoinSession` e `DestroySession`;
- A implementação por meio do C++ não trouxe resultados positivos pois, mesmo que um jogador conseguisse criar uma sessão, outros jogadores não conseguiam encontra-la;
- Então, foi criado um Blueprint derivado do mesmo componente. Criando os devidos comandos, foi possível encontrar sessões e se juntar a elas, resolvendo assim o problema.

Implementação

- Com relação aos tratamentos do jogo e a utilização dos atributos dos elementos principais, o primeiro passo foi a criação de estruturas referenciando a nave e as armas;
- As estruturas foram criadas e ajustadas para possibilitar a utilização dentro dos Blueprints a partir de *defines* do UE4.

```
1   USTRUCT(BlueprintType)
2   struct FRegWeapon
3   {
4       GENERATED_BODY()
5
6       UPROPERTY(EditAnywhere, BlueprintReadWrite)
7           int id;
8       UPROPERTY(EditAnywhere, BlueprintReadWrite)
9           bool isBasicWeapon;
10      UPROPERTY(EditAnywhere, BlueprintReadWrite)
11          FString name;
12      UPROPERTY(EditAnywhere, BlueprintReadWrite)
13          FString description;
14      UPROPERTY(EditAnywhere, BlueprintReadWrite)
15          FRegShot shot;
16  };
```

Implementação

- Para o comportamento do jogador, foi criada uma classe C++, derivada do componente `Character`, e um Blueprint derivado da classe. A classe pai irá possuir objetos do tipo `FRegSpaceship` e `FRegShot` e será a responsável pela alteração e utilização dos atributos dessas estruturas;
- Assim, ela é quem trata como a vida do jogador é afetada quando atingido por um tiro;
- Já na classe derivada, são tratados os aspectos visuais do jogador, como sua movimentação, o *spawn* dos tiros e a relação com as informações disponíveis na tela do usuário.

Implementação

- Após realizar testes com o resultado da classe do jogador, porém, foi identificado que os comandos não eram replicados para todos;
- Visando o compartilhamento de informações, foi necessário aplicar uma diferente técnica ao criar os *spawns* na tela. A função que realiza os *spawns* foi alterada para rodar *multicast* e ela precisa ser chamada por uma nova função que roda no servidor;
- Ao invés de chamar diretamente o *spawn*, agora o jogador local chama a função no servidor e ele que faz a chamada. Essa chamada passa a ser feita, então, para todos os jogadores, replicando as ações entre eles.

Análise dos Resultados

Comparativo de todos os trabalhos correlatos:

Trabalhos Correlatos Características	Trupel (2008)	Araujo e Cordenonsi (2003)	Jansen (2020)	SpaceRoyale
Ferramenta de desenvolvimento	Plataforma J2ME utilizando MIDP.	Ambientes virtuais utilizando VRML e HTML	HTML 5 e WebSockets	C++, Unreal Engine e Blueprints
Multiplayer	X		X	X
Simulador	X	X		X
Performance	Problemas de latência	Requer uma taxa de transferência maior	Lentidão e travamento com a movimentação do personagem	Travamento e bugs com a movimentação da nave
Perspectiva	Top-down	Primeira pessoa	Top-down	Terceira pessoa

Análise dos Resultados

Questionário para validação dos resultados obtidos

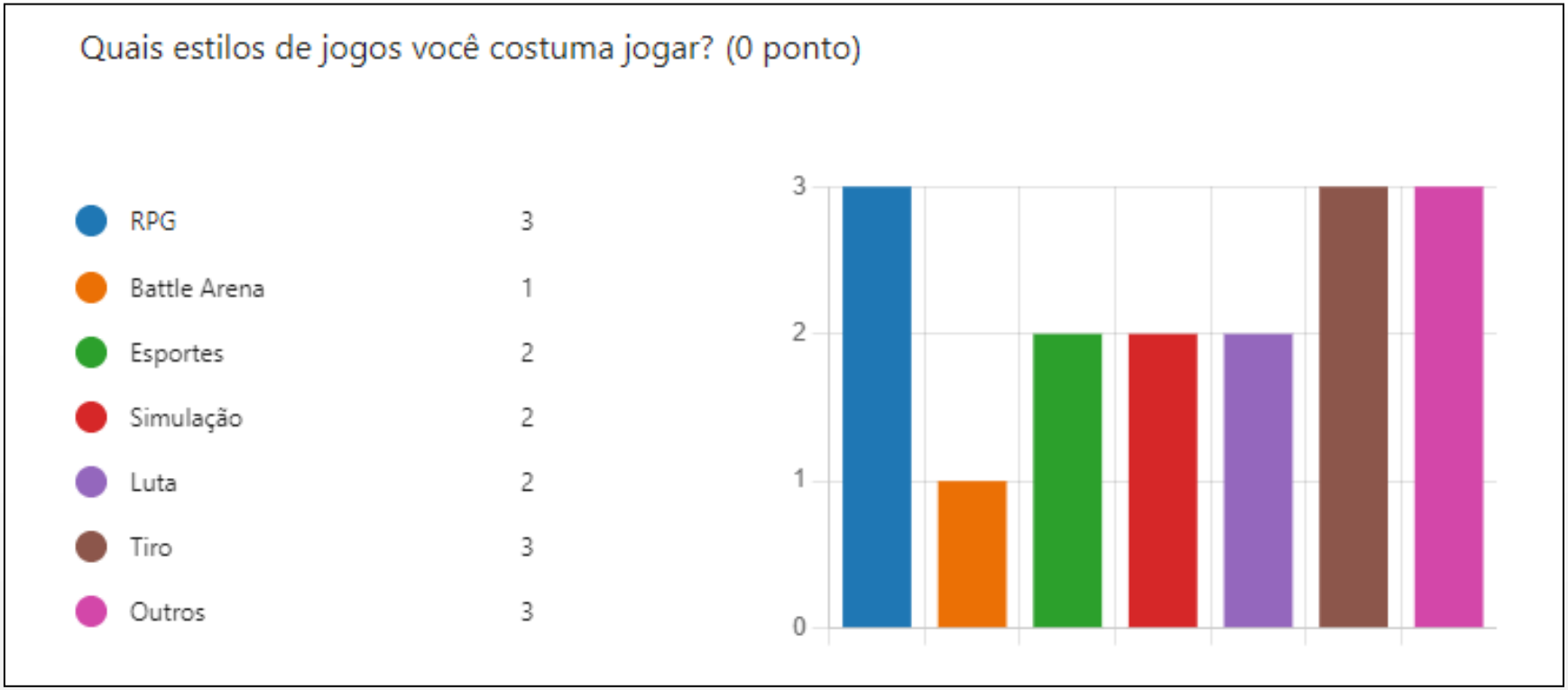
- No questionário foram apresentadas 12 perguntas sobre o perfil dos participantes com relação a jogos digitais e suas impressões quanto à jogabilidade do Space Royale.

Perfil dos participantes:

- *Frequência com que costumam jogar*: 1 joga com grande frequência; 2 jogam diariamente;
- *Preferência sobre interação com outros jogadores*: 1 prefere jogos single player; 2 preferem jogos multiplayer;

Análise dos Resultados

Estilos de jogos mais costumeiros:



Análise dos Resultados

Resultados obtidos quanto ao jogo:

- Sobre o jogo, os participantes deram notas de 1 a 5 para as seguintes questões: jogabilidade, animação, som, interação entre jogadores, regras do jogo e pontuações;
- Para todos os pontos, a média de nota obtida ficou entre 3,0 e 3,6 de 5,0.

Análise dos Resultados

Visão geral sobre a jogabilidade:

- Foi apontado um problema no qual a nave de um dos jogadores parou de se movimentar e outro no qual as naves ficaram invisíveis para o oponente quando restarem somente dois jogadores no mapa.
- De forma geral, os participantes acharam a jogabilidade fácil e gostaram do Battle Royale e do design do jogo, mas acharam também que os erros atrapalham e, no caso do jogador que perdeu o movimento de sua nave, ele não conseguiu jogar como queria já que podia somente atirar.

Conclusões e Sugestões

Dificuldades encontradas durante o desenvolvimento:

- O principal desafio foi trabalhar os aspectos do jogo que eram afetados quando dentro de uma sessão;
- A utilização de sessões acabou trazendo maior complexidade ao desenvolvimento.
- Contudo, a utilização do Unreal Engine somente foi conturbada em quesitos onde não se tinha um conhecimento inicial, sendo necessárias muitas pesquisas em busca de soluções.

Conclusões e Sugestões

Cumprimento dos objetivos específicos:

1. *Disponibilizar um jogo Battle Royale:* foi alcançado com a entrega de um jogo funcional que possui características de um Battle Royale;
2. *Categorização das armas*: foi alcançado parcialmente pois, dentre as armas tratadas, o escudo apresentou um comportamento irregular, na qual em alguns casos bloqueia o dano e em outros não;
3. *Fornecer um jogo com boa jogabilidade*: foi alcançado parcialmente por conta do problema com a movimentação que é um empecilho que afeta a experiência dos usuários, mesmo possuindo uma “jogabilidade fácil”.

Conclusões e Sugestões

Cumprimento do objetivo geral:

- Pode-se dizer que o objetivo geral de desenvolver um jogo de Battle Royale utilizando o Unreal Engine foi alcançado. O jogo possui as características de um jogo multiplayer no estilo Battle Royale e foi criado apenas com as ferramentas disponibilizadas pelo Unreal Engine.

Conclusões e Sugestões

Possíveis extensões para este trabalho:

- Melhorar performance durante a partida;
- Listar as sessões encontradas para que o usuário possa escolher em qual quer entrar;
- Aumentar a capacidade de jogadores por sessão;
- Implementar um perfil para o usuário, no qual possam ser registradas as suas conquistas;
- Implementar uma lista de amigos para que os jogadores possam manter contato após as partidas;
- Implementar a delimitação de zona ao mapa com uma nuvem tóxica que vai suprimindo as áreas das pontas até o centro e queima jogadores que a tocarem.

Muito Obrigada!

FURB
UNIVERSIDADE DE BLUMENAU