# Análise de Projeto

**Uncertainty**

Null References

2024

# Sumário

# Informações Gerais

| | |
|---|---|
| *Versão do projeto* | *2021.3.19f1* |
| *Versão utilizada* | 2021.3.19f1 |
| *Acesso* | Github<br>https:// github.com/Null-References/Uncertainty |
| *Hardware para builds* | 1. i7-13700H 2.40 GHz, RAM 16.00 GB, GeForce RTX 4060, SSD 1 TB<br>2. i5-4200U 1.60 GHz, RAM 6.00 GB, HD 500 GB<br>3. Intel(R) Celeron(R) N4020, RAM 8.00 GB, HD 500 GB |

## Objetivos da atividade

1. **Analisar** o projeto em suas diversas áreas: scripts, sprites, texturas, sons, Canvas, materiais, shaders entre outras;
2. **Identificar** problemas nas áreas citadas ou que não foram feitas da maneira mais adequada.

## Objetivos do documento

1. Indicar os problemas encontrados;
2. Sugerir mudanças para melhorias em desempenho e arquitetura do projeto.

## Autor

URZ Programação de jogos LTDA

Abnner Urzedo

A empresa ou pessoa não contratou o serviço.

Documento demostrativo.

# Scrpits

## NormalBullet e LookAt

Resumo

Criar variável de cachê para componente **Transform** do próprio objeto.

***Gravidade dos problemas*:** Baixa (performance)

***Tempo médio para aplicar melhorias:*** 10m

***Dificuldade para aplicar melhorias:*** Baixa

Embora os estes scripts não apresentem grandes problemas, sugiro implementar o cache do componente **Transform** do próprio objeto.

NormalBullet/LookAt

```
Transform _transform;

void Start(){
(...)
_transform = GetComponent<Transform>();
        //ou
_trasform = transform;
}
```

C#

## ShipShootingController

Resumo

Cachê do **Transform** da câmera (**ShipCamera**), além dos valores **Screen.width** e **Screen.height**. Também recomendo usar o **RaycastNonAlloc** no lugar do Raycast padrão.

***Gravidade dos problemas*:** Baixa (performance)

***Tempo médio para aplicar melhorias:*** 1h

***Dificuldade para aplicar melhorias:*** Baixa

Não há grandes problemas neste script. No entanto, como sugestão de melhoria, recomendo que seja adicionado uma variável de cachê do componente **Transform** da "**ShipCamera**", a fim de reduzir chamadas de "**GetComponent<Transform>**". Também seria benéfico guardar os valores de "**Screen.width**" e "**Screen.height**", já que, no geral, esse valor não muda durante o jogo, então não é preciso acessando-os várias vezes.

Uma última sugestão para este script seria o uso de "**RaycastNonAlloc**" ao invés do **Raycast** padrão. Essa alteração, serviria para evitar o acúmulo de lixo na memória.

ShipShootingController

```
   RaycastHit results = new RaycastHit[1];

   void Update(){
   (...)
   if(Physics.RaycastNonAlloc(shipCamera.transform.position, direction, results, 50, whatIsAimAbleLayer))
   }
```

C#

## Health

Resumo

Evitar o uso de **UnityEvent** , substituindo-o pelos eventos padrão C#.

***Gravidade dos problemas*:** Baixa (estrutura de projeto)

***Tempo médio para aplicar melhorias:*** 30m

***Dificuldade para aplicar melhorias:*** Baixa

É importante tomar cuidado ao usar eventos **UnityEvent**. No atual estado do projeto, esses eventos não vão ter tanto impacto. Porém, futuramente, quando for adicionado partículas, sons e animações, que são acionados quando uma entidade é atingida, por exemplo, o jogo pode começar a apresentar lags.

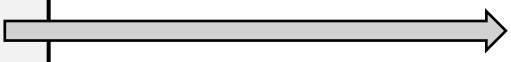
Uma abordagem para mitigar o problema, é usar os eventos padrão do C#. Como exemplo, ao invés de usar o **UnityEvent**, seria criado um "**OnDeath**" do tipo **Action** que seria invocado no script Health e recebido no script **Player**.

Health

```
 Action OnDeath;

 void ReduceHealth(){
 (...)
 if (_currentHealth <= 0)
 {
        OnDeath?.Invoke();
 }
 }
```

C#



Player

```
  Health health;

  void Start(){
  health.OnDeath += PlayerDeath;
  }
```

C#

Essa mesma lógica se aplicaria para scripts futuros, como os já citados anteriormente de som, partículas e animações.

Ressalto que, atualmente, o uso de **UnityEvent** não afeta a performance do projeto. Porém, caso venha a afetar, a sugestão é fazer um sistema usando eventos padrões do C#.

# IA Bot

Resumo

Recomendação de **Behaviour Tree** para programar as inteligências artificiais.

***Gravidade dos problemas*:** Baixa (estrutura de projetos)

***Tempo médio para aplicar melhorias:*** 8h

***Dificuldade para aplicar melhorias:*** Alta

Sugiro o uso de **Behaviour Tree** para aprimorar a programação de comportamentos da IA utilizando nodes de comportamento e sequências para orientar a execução das tarefas. Embora a **state machine** como está hoje, até lembre um pouco essa estrutura de comportamentos segmentados, ela ainda tem o defeito de ser mais difícil de se fazer a manutenção e adicionar novos comportamentos, especialmente se for algo bem específico.

O uso de **Behaviour Tree** com certeza vai melhorar ainda mais a estrutura do projeto e facilitar alterações futuras, com um único ponto negativo de ter uma dificuldade inicial alta, que vai diminuindo conforme mais e mais partes de comportamento vão sendo programadas.

# Modelos/Texturas

## Terrain

***Gravidade dos problemas:*** Alta (performance)

***Tempo médio para aplicar melhorias:*** 4h

***Dificuldade para aplicar melhorias:*** Médio

O terreno do projeto é o mais forte candidato a está diminuindo drasticamente a performance do projeto. Uma das evidências é o aumento de FPS quando se desliga o terreno, assim como a diminuição da quantidade de batches. A grama é especialmente problemáticas, contribuindo significativamente para o aumento na quantidade de batches.

Otimizar o terreno é um desafio, pois requer um equilíbrio entre qualidade visual e desempenho. Algumas sugestões para mitigar o impacto no desempenho incluem:

**Aumentar o Pixel Error** – Controla o nível de detalhe na malha do terreno com base na distância da câmera.

**Diminuir o Base Map Dist** – Controla a distância na qual o terreno começará a usar texturas de menor resolução.

**Diminuir o Detail Density** – Diminui a quantidade de grama no terreno, o que diminui a quantidade de batches.

**Diminuir o Detail Distance** – Controla a distância que é gerado a grama do terreno.

**Smooth Height** – Essa ferramenta auxilia na redução de detalhes e irregularidades na malha do terreno, resultando em menos triângulos renderizados.

É crucial realizar testes com diferentes valores para encontrar o equilíbrio ideal. Priorizar a redução da quantidade de gramas no terreno pode ter o maior impacto perceptível na melhoria do desempenho.

## Modelos das ilhas e pedras

***Gravidade dos problemas:*** Médio (performance)

***Tempo médio para aplicar melhorias:*** 3h

***Dificuldade para aplicar melhorias:*** Baixa

Para esses modelos, recomendo a criação de **LODs**. Existe opções de Assets que criam LODs automaticamente e facilitam esta tarefa.

Com os **LODs**, é possível reduzir a quantidade de triângulos renderizados em um quadro, diminuindo a complexidade de objetos que estão distantes da câmera. Isso pode resultar em melhorias significativas no desempenho do jogo, garantindo uma renderização mais eficiente e suave, especialmente em cenas com muitos objetos.

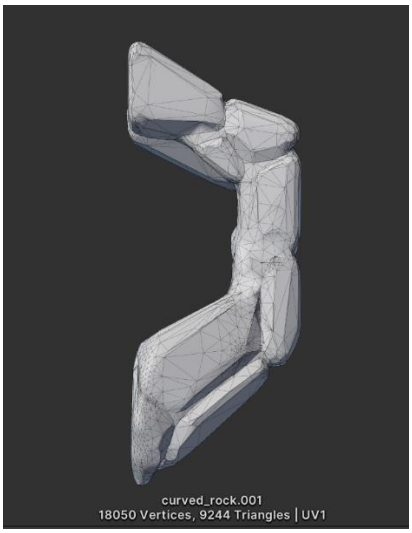
## Curve Rock

***Gravidade dos problemas:*** Médio (performance)

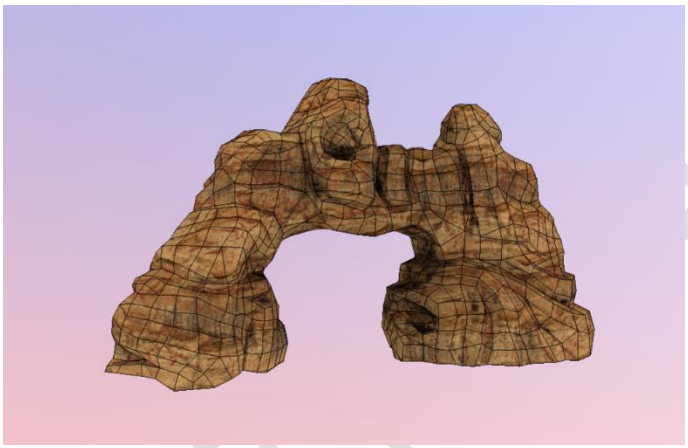
***Tempo médio para aplicar melhorias:*** 4h

***Dificuldade para aplicar melhorias:*** Baixa

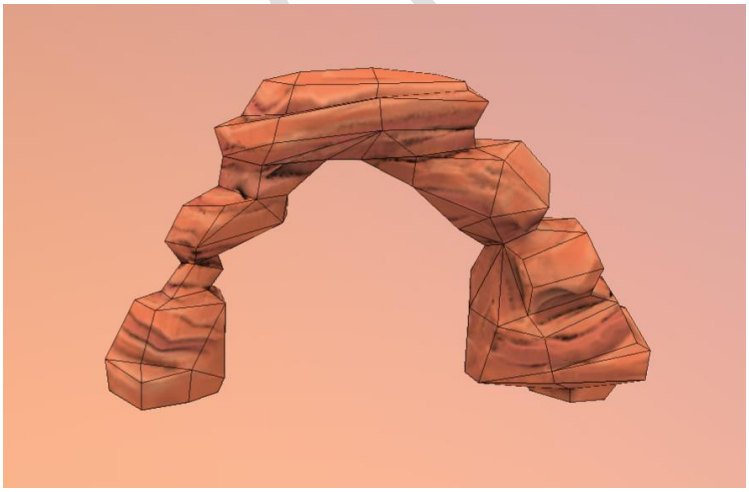
Esse modelo talvez tenha vértices e triângulos demais, se comparado com outros modelos similares.




**Objeto Original**:
18k Vértices
9k Triângulos



**Um semelhante**:
1.2k Vértices
1.2k Triângulos



**Uma versão lowpoly**:
- Vértices
302 Triângulos

Recomendo uma revisão no modelo, pois ele parece ter bem mais vértices e triângulos do que necessário.

## Grama do terreno

***Gravidade dos problemas:*** Baixa (tamanho de arquivo)

***Tempo médio para aplicar melhorias:*** 10m

***Dificuldade para aplicar melhorias:*** Baixa

A textura da grama do terreno, pode ser otimizada reduzindo o tamanho máximo (**MaxSize**) e aplicando a compressão **Crunch**. Essas opções ajudam a diminuir o tamanho do arquivo da textura, o que não só economiza espaço em disco, mas também melhora o processamento, resultando em tempos de carregamento mais rápidos.

## Abelhas

***Gravidade dos problemas:*** Baixa (tamanho de arquivo e performance)

***Tempo médio para aplicar melhorias:*** 10m

***Dificuldade para aplicar melhorias:*** Baixa

A textura das abelhas, podem ser otimizadas reduzindo o tamanho máximo (**MaxSize**) e aplicando a compressão **Crunch**.

Opcionalmente, o material desses modelos pode usar um **shader mobile**, que é mais otimizado, como o **Bumped Diffuse**. Mas, fazer isso irá alterar um pouco o visual destes modelos.

Essas mudanças vão diminuir o tamanho do arquivo e melhorar o processamento do jogo.

# Fisica

Resumo

Reduzir **Mesh Colliders**.

***Gravidade dos problemas*:** Médio (performance)

***Tempo médio para aplicar melhorias:*** 2h – 3H

***Dificuldade para aplicar melhorias:*** Baixa

É preciso diminuir a quantidade de colliders do tipo **Mesh Colliders**, pois este é o collider que mais demanda processamento. A solução é simples: substituir os colliders complexos por opções mais simples, como **Box Collider, Sphere Collider** e **Capsule Collider.**

É importante se ater a esse problema, pois o projeto já apresenta sinais de problema com processamento de física, já que, como mostra no profile, os cálculos de física e colisão já estão um pouco mais altos do que deveriam.

# Renderização

Resumo

Adicionar o **Occlusion Culling** e tentar modificar o **Far Clip** da câmera para diminuir a distancia de renderização.

***Gravidade dos problemas*:** Médio (performance)

***Tempo médio para aplicar melhorias:*** 3h – 4H

***Dificuldade para aplicar melhorias:*** Baixa

Especialmente quando não se utiliza o URP, é crucial adicionar o **Occlusion Culling** ao projeto. Essa técnica garante que elementos não visíveis na tela não sejam renderizados, o que pode significativamente melhorar o desempenho.

Uma opção que pode ajudar com o tempo de renderização, é o **Far Clip** da câmera. Essa opção determina a distância de renderização e diminui-la ajuda a **reduzir** a quantidade de **batches** e **triângulos**. Aliado a isso, o uso de **fog** (neblina) pode ser benéfico para mascarar objetos que não estão sendo renderizados, proporcionando uma transição suave e evitando que o jogador perceba uma mudança abrupta na cena.

Essas medidas combinadas ajudarão a otimizar o desempenho e aprimorar a experiência do usuário.

# Desempenho Builds

| *Configurações de pc* | *Média FPS* | *Média MS* |
|---|---|---|
| *i7-13700H 2.40 GHz*<br>*RAM 16.00 GB*<br>*GeForce RTX 4060*<br>*SSD 1 TB* | 190 - 210 | 5.2 – 4.5 |
| *i5-4200U 1.60 GHz*<br>*RAM 6.00 GB*<br>*HD 500 GB* | 22 – 28 | 45 - 35 |
| *Intel(R) Celeron(R) N4020*<br>*RAM 8.00 GB*<br>*HD 500 GB* | 12 - 20 | 83 - 50 |



X _____

NAME