Project Analysis

Uncertainty

Null References

2024

Sumário

General Information

Project version	*2021.3.19f1*
Version used	2021.3.19f1
Access	Github https:// github.com/Null-References/Uncertainty
Hardware for builds	1. i7-13700H 2.40 GHz, RAM 16.00 GB, GeForce RTX 4060, SSD 1 TB 2. i5-4200U 1.60 GHz, RAM 6.00 GB, HD 500 GB 3. Intel(R) Celeron(R) N4020, RAM 8.00 GB, HD 500 GB

Objectives

1. **Analyze** the project in its various areas: scripts, sprites, textures, sounds, Canvas, materials, shaders, among others;
2. **Identify** problems in the areas mentioned or that were not done in the most appropriate way.

Document

1. Indicate the problems encountered;
2. Suggest changes to improve the project's performance and architecture.

Author

URZ Programação de jogos LTDA

Abnner Urzedo

The company or person did not contract the service.

Example document.

Scrpits

NormalBullet e LookAt

Summary

Create cache variable for **Transform** component of the object itself.

Severity of issues: Low (performance)

Average time to apply improvements: 10m

Difficulty in applying improvements: Low

Although these scripts do not present major problems, I suggest implementing the cache of the Transform component of the object itself.

NormalBullet/LookAt

```
Transform _transform;

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

C#

ShipShootingController

Summary

Cache the camera's **Transform** (ShipCamera), as well as the **Screen.width** and **Screen.height** values. I also recommend using **RaycastNonAlloc** instead of the default Raycast.

***Severity of issues*:** Low (performance)

Average time to apply improvements: 1h

Difficulty in applying improvements: Low

There are no major issues with this script. However, as a suggested improvement, I recommend adding a cache variable for the **Transform** component of the "**ShipCamera**" in order to reduce calls to **"GetComponent<Transform>".** It would also be beneficial to store the values of "**Screen.width**" and "**Screen.height**", since, in general, these values do not change during the game, so there is no need to access them multiple times. One last suggestion for this script would

be to use "**RaycastNonAlloc**" instead of the default Raycast. This change would help to avoid the accumulation of garbage in memory.

ShipShootingController

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

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

C#

Health

Summary

Avoid using **UnityEvent** , replacing it with standard C# events.

***Severity of issues*:** Low (project structure)

Average time to apply improvements: 30m

Difficulty in applying improvements: Low

It is important to be careful when using **UnityEvent** events. In the current state of the project, these events will not have much impact. However, in the future, when particles, sounds and animations are added, which are triggered when an entity is hit, for example, the game may start to lag.

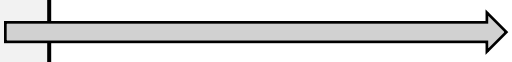
One approach to mitigate the problem is to use standard C# events. For example, instead of using UnityEvent, an "**OnDeath**" of type Action would be created that would be invoked in the Health script and received in the Player script.

Health

```
 Action OnDeath;

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

C#



Player

```
  Health health;

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

C#

This same logic would apply to future scripts, such as those previously mentioned for sound, particles and animations.

I would like to point out that, currently, the use of **UnityEvent** does not affect the performance of the project. However, if it does, the suggestion is to create a system using standard C# events.

IA Bot

Summary

Behavior Tree recommendation for programming artificial intelligences.

***Severity of issues*:** Low (project structure)

Average time to apply improvements: 8h

Difficulty in applying improvements: High

I suggest using **Behavior Tree** to improve AI behavior programming by using behavior nodes and sequences to guide task execution. Although the state machine as it stands today somewhat resembles this structure of segmented behaviors, it still has the drawback of being more difficult to maintain and add new behaviors, especially if it is something very specific.

Using **Behavior Tree** will certainly further improve the project structure and facilitate future changes, with the only downside being that it has a high initial difficulty, which decreases as more and more behavior parts are programmed.

Models/Textures

Terrain

Severity of issues: High (performance)

Average time to apply improvements: 4h

Difficulty in applying improvements: Average

The terrain of the project is the strongest candidate for drastically decreasing the performance of the project. One of the evidences is the increase in FPS when the terrain is turned off, as well as the decrease in the number of batches. Grass is especially problematic, contributing significantly to the increase in the number of batches. Optimizing the terrain is a challenge, as it requires a balance between visual quality and performance. Some suggestions for mitigating the impact on performance include:

Increase o Pixel Error – Controls the level of detail in the terrain mesh based on the distance from the camera.

Decrease o Base Map Dist – Controls the distance at which the terrain starts using lower resolution textures.

Decrease o Detail Density – It reduces the amount of grass on the land, which reduces the number of batches.

Decrease o Detail Distance – Controls the distance that grass is generated from the terrain.

Smooth Height – This tool helps reduce details and irregularities in the terrain mesh, resulting in fewer rendered triangles.

It is crucial to test different values to find the ideal balance. Prioritizing reducing the amount of grams on the ground can have the greatest noticeable impact on improving performance.

Models of islands and rocks

Severity of issues: Average (performance)

Average time to apply improvements: 3h

Difficulty in applying improvements: Low

For these models, I recommend creating LODs. There are Assets options that automatically create LODs and make this task easier.

With LODs, it is possible to reduce the number of triangles rendered in a frame, reducing the complexity of objects that are far from the camera. This can result in significant improvements in game performance, ensuring more efficient and smooth rendering, especially in scenes with many objects.

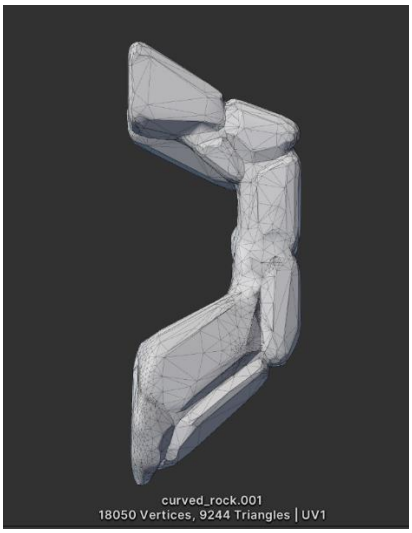
Curve Rock

Severity of issues: Average (performance, file size)

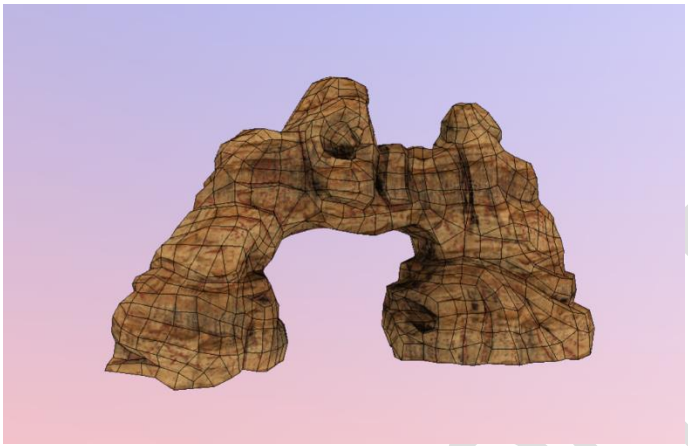
Average time to apply improvements: 4h

Difficulty in applying improvements: Low

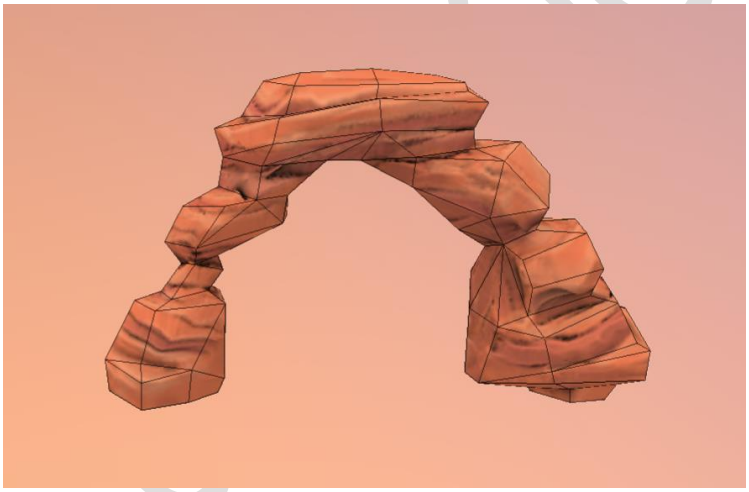
This model may have too many vertices and triangles compared to other similar models.



Original Object:
18k Vértices
9k Triângulos



A similar one:
1.2k Vértices
1.2k Triângulos



A lowpoly version:
302 Triângulos

I recommend reviewing the model, as it appears to have far more vertices and triangles than necessary.

Ground grass

Severity of issues: Low (file size)

Average time to apply improvements: 10m

Difficulty in applying improvements: Low

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.

Bees

Severity of issues: Low (file size, performance)

Average time to apply improvements: 10m

Difficulty in applying improvements: Low

Bee textures can be optimized by reducing the maximum size (**MaxSize**) and applying **Crunch compression**.

Optionally, the material for these models can use a more optimized mobile shader, such as **Bumped Diffuse**. However, doing so will change the look of these models slightly.

These changes will reduce the file size and improve the game's rendering.

Physics

Summary

Reduce **Mesh Colliders**.

***Severity of issues*:** Average (performance)

Average time to apply improvements: 2h – 3H

Difficulty in applying improvements: Low

It is necessary to reduce the number of **Mesh Colliders**, as this is the type of collider that demands the most processing. The solution is simple: replace the complex colliders with simpler options, such as **Box Collider**, **Sphere Collider** and **Capsule Collider**.

It is important to pay attention to this issue, as the project already shows signs of a problem with physics processing, since, as shown in the profile, the physics and collision calculations are already a little higher than they should be.

Rendering

Summary

Add **Occlusion Culling** and try to modify the camera's **Far Clip** to decrease the render distance.

***Severity of issues*:** Average (performance)

Average time to apply improvements: 3h – 4H

Difficulty in applying improvements: Low

Especially when not using URP, it is crucial to add **Occlusion Culling** to your project. This technique ensures that non-visible elements on the screen are not rendered, which can significantly improve performance.

One option that can help with rendering time is the Camera **Far Clip**. This option determines the rendering distance and decreasing it helps to reduce the amount of batches and triangles. In addition, the use of fog can be beneficial to mask objects that are not being rendered, providing a smooth transition and preventing the player from noticing an abrupt change in the scene.

These measures combined will help to optimize performance and improve the user experience.

Performance Builds

PC settings	*Average FPS*	*MS average*
i7-13700H 2.40 GHz *RAM 16.00 GB* *GeForce RTX 4060* *SSD 1 TB*	190 - 210	5.2 – 4.5
i5-4200U 1.60 GHz *RAM 6.00 GB* *HD 500 GB*	22 – 28	45 - 35
Intel(R) Celeron(R) N4020 *RAM 8.00 GB* *HD 500 GB*	12 - 20	83 - 50

X _____

NAME