TestTask For Q-Games

Name : Isaiah Adebanji Isaiah
Project Name : Pirates
Engine : Unity

Sounds used : Demon Slayer Hashira training arc.
ExternalPackage : Polygon Arsenal(projectiles effects)

Test Content

This project is currently at a very basic stage and requires improvements to make it more engaging. The following enhancements were specifically requested:

- Improve the enemy's movement, actions, and spawning to make them more dynamic and enjoyable to play against.
- Enhance the player's movement and behavior, potentially adding new features.
- Add visual and audio effects for bullets, enemy collisions, etc.
- Implement a playable end condition for the stage, such as game over on player failure or a time limit.

Summary

Thank you for the opportunity. I've made several enhancements to the game: i added new modes, improved UI components, integrated quality sounds, and introduced an Eagle Vision mode. This mode scans enemies and reveals cargo types like iron, rum, and wood, which are essential for repairing ships or crafting rum for stronger attacks (assuming sufficient resources).

Additionally, I migrated the project to the Universal Render Pipeline for better performance and utilized post-processing for Eagle Vision effects. Finally, I implemented a robust object pool system to efficiently manage enemy spawns, bullets, and other projectiles, optimizing resource usage.

Note on Null Checking:

I have implemented null checking in the script selectively, based on the specific context and potential failure points. The approach avoids excessive defensive null checking, which can impact performance, while ensuring robustness where necessary. This strategy enhances code readability and maintains optimal performance by focusing on critical areas such as resource access and inter-component communication.

機会をいただき、ありがとうございます。ゲームにいくつかの改良を加えました。新しいモードの追加、UIコンポーネントの改善、クオリティの高いサウンドの統合、そしてイーグルビジョンモードの導入を行いました。このモードは敵をスキャンし、鉄やラム、木材などの貨物を特定します。これらは船の修理やラムの製造に必要で、攻撃力を強化するために使用できます（資源が十分である場合）。

さらに、プロジェクトをユニバーサルレンダーパイプラインに移行してパフォーマンスを向上させ、イーグルビジョン効果にポストプロセッシングを活用しました。最後に、効率的に敵の出現、弾丸、その他の投射物を管理するために強力なオブジェクトプールシステムを実装し、リソースの最適化を図りました。

Controls
Shoot : Press Z
Aim : Hold Right Mouse
Controls : WASD
EagleVision : V
Throw Rum : Aim + G
Open Inventory : Hold I

Note:
You can Craft and Repair Ship in inventory
Scan Ship with Eagle vision

Game Overview
The game involves controlling a ship to defeat all enemies present in a scene. Players can customize game modes, such as selecting the number of enemies to face and the total number to defeat. The game incorporates various features like ship movement, cannon attacks, and special attacks using "rum" for increased firepower. Players collect items (iron, rum, wood) from defeated enemies to repair their ship or craft more rum.

Key Features:

- Game Modes: Implemented using an interface (IGameMode.cs) allowing for multiple modes; currently includes "DefeatAllEnemies.cs".

- Enemy Management: Enemies are dynamically spawned and managed (EnemyManager.cs). They have individual AI states (idle, attack, roaming) and drop loot items on defeat.

- Player Controls: Ship movement and attacks are handled through Player.cs and ShipCannons.cs, allowing players to navigate, shoot cannons, and use special rum attacks.

- Game State Management: StageLoop.cs manages game state transitions, initializes game modes, and checks win conditions.

- UI and Settings: GameModeSelector.cs manages UI elements for selecting game settings like maximum active enemies and total enemies before starting a game mode.

ゲーム概要:
このゲームでは、船を操作してシーン内の全ての敵を倒すことが目的です。プレイヤーは敵の数や倒すべき総数などのゲームモードをカスタマイズできます。ゲームには船の移動、大砲攻撃、そして「ラム」を使用した特別な攻撃などの機能が組み込まれています。プレイヤーは敵からアイテム（鉄、ラム、木材）を集めて船を修理したり、より多くのラムを作成することができます。

主な特徴:

- ゲームモード: インターフェースを使用して複数のモードを実装し、現在は「DefeatAllEnemies.cs」が含まれています。

- 敵管理: 敵は動的にスポーンされ、AIの状態（アイドル、攻撃、徘徊）を持ち、撃破時にはアイテムをドロップします（EnemyManager.cs）。

- プレイヤー操作: 船の移動と攻撃はPlayer.csとShipCannons.csで処理され、プレイヤーは航行、大砲の発射、特殊なラム攻撃を行います。

- ゲーム状態管理: StageLoop.csはゲームの状態遷移を管理し、ゲームモードを初期化し、勝利条件を確認します。

- UIと設定: GameModeSelector.csは、最大アクティブ敵数やゲームモード開始前の総敵数などのゲーム設定を選択するためのUI要素を管理します。

Object Pooling System Summary
Purpose:
This object pooling system efficiently manages and reuses game objects such as bullets, enemies, and explosions. Instead of frequently instantiating and destroying objects, which can strain system resources, objects are pre-instantiated and recycled from a pool as needed.

Components:
Destructable Class:

- Handles destruction behaviors for game objects.
- Implements object pooling functionality through an interface (IDestruct).
- Manages specific behaviors for wooden ship parts, including resetting scale and position.

ObjectPool Class:

- Centralizes management of multiple object pools using dictionaries.
- Initializes pools of different types (PoolTag) with a set number of prefabs (Destructable).
- Spawns objects from pools based on tag, position, and rotation.
- Returns objects to their respective pools for reuse when no longer needed.

Pool Class:

- Defines pool configurations (PoolTag) containing prefabs (Destructable) and pool sizes.
- Serialized to allow easy configuration via the Unity Inspector.

Workflow:

- **Initialization:** All object pools are set up and initialized during startup or scene loading based on predefined settings.

- **Spawning:** When an object (e.g., bullet, enemy) is required, the system checks if an inactive instance is available in the corresponding pool.

- **Reusing:** If an inactive instance exists, it is activated and positioned as needed.

- **Returning:** Once an object is no longer in use (e.g., destroyed or out of range), it is deactivated and returned to its pool for future reuse.

オブジェクトプーリングシステムの要約
目的:
このオブジェクトプーリングシステムは、Unityで弾丸、敵、爆発などのゲームオブジェクトを効率的に管理し再利用するために設計されています。頻繁にオブジェクトをインスタンス化して破棄する代わりに、必要な時にプールから事前にインスタンス化されたオブジェクトを再利用します。

コンポーネント:
Destructable クラス:

ゲームオブジェクトの破壊挙動を管理します。
インターフェース (IDestruct) を通じてオブジェクトプーリング機能を実装します。
木製船の部品など、特定の挙動を管理します。スケールと位置をリセットします。
ObjectPool クラス:

辞書を使用して複数のオブジェクトプールを中央管理します。
異なる種類のプール (PoolTag) を設定し、指定されたプレハブ (Destructable) の数で初期化します。
タグ、位置、回転に基づいてプールからオブジェクトを生成します。
不要になったオブジェクトは、それぞれのプールに戻され再利用されます。
Pool クラス:

プレハブ (Destructable) とプールサイズからなるプール設定 (PoolTag) を定義します。
Unityインスペクタを通じて簡単に設定できるようにシリアライズ化されています。
ワークフロー:
初期化: スタートアップやシーン読み込み時に、全てのオブジェクトプールが予め設定され、初期化されます。

生成: 弾丸、敵などのオブジェクトが必要な時、システムは対応するプールから非アクティブなインスタンスが利用可能か確認します。

再利用: 非アクティブなインスタンスが存在する場合、必要な位置や回転に応じて再利用されます。

戻す: オブジェクトが不要になった時（例：破壊された場合や範囲外になった場合）、それは非アクティブ化され、将来の再利用のためにプールに戻されます。

SoundManager Class Summary
Purpose:
The SoundManager class manages audio playback efficiently by implementing object pooling for audio sources. This approach minimizes resource consumption and allows for dynamic audio playback in response to game events.

Components:
SoundManager Class:

- **Sound Settings:** Manages an array of Sound objects that define different audio clips and their properties, including maximum instances allowed concurrently.
- **Audio Source Management:** Utilizes a pool of AudioSource components to play audio clips, ensuring that multiple sounds can play simultaneously without creating new AudioSource instances.
- **Dynamic Sound Playback**: Plays audio clips associated with specific SoundTag identifiers, selecting a random clip from the available options.
- **Instance Tracking:** Tracks the number of currently playing instances for each sound to enforce maximum instance limits.
- **Background Sound Control:** Provides functionality to toggle background music on/off, with visual feedback through associated game objects (m_SoundOnObj and m_SoundOffObj).

Sound Class:

- **Sound Definition:** Defines a sound with a unique identifier (m_Id), an array of possible AudioClip variations (m_Clips), and a maximum number of instances (m_MaxInstances) that can play simultaneously.
- **Instance Tracking**: Maintains a count (m_CurrentInstances) of currently active instances of the sound to enforce concurrency limits.

Workflow:

- **Initialization:** On Start, initializes the SoundManager by populating dictionaries and creating a pool of AudioSource components based on m_audioSourceSize and m_audioSourceVolume.

- **PlaySound Method:** Plays a sound associated with a given SoundTag, selecting a random AudioClip from its list. It checks if the maximum instances for that sound have been reached before playing.

- **GetAvailableAudioSource Method**: Retrieves an available AudioSource from the pool for playing a sound clip.

- **ResetSoundInstanceCount Coroutine:** Coroutine that resets the instance count for a sound after its clip finishes playing, ensuring accurate tracking of active instances.

- **ToggleBackGroundSound Method:** Toggles background music on/off and manages associated game objects (m_SoundOnObj and m_SoundOffObj) for visual feedback. Stops background music if it's currently playing when toggled off.

- **StopSound Method:** Stops all instances of a sound associated with a given SoundTag and resets its instance count.

Reason for Implementation:
The SoundManager employs object pooling to efficiently manage and reuse AudioSource components, minimizing performance overhead compared to dynamically creating and destroying them. This approach ensures smooth audio playback and avoids cluttering the game with excessive AudioSource instances, contributing to a more optimized gameplay experience.

SoundManager クラスの要約
目的:
UnityのSoundManagerクラスは、オーディオ再生を効率的に管理するためにオブジェクトプーリングを実装しています。これにより、リソースの消費を最小限に抑えながら、ゲームのイベントに応じてダイナミックなオーディオ再生が可能となります。

コンポーネント:
SoundManager クラス:

サウンド設定: 複数の Sound オブジェクトを管理し、異なるオーディオクリップとそのプロパティ（最大同時インスタンスなど）を定義します。
オーディオソース管理: AudioSource コンポーネントのプールを使用してオーディオクリップを再生します。新しい AudioSource インスタンスを作成せず、複数のサウンドを同時に再生できます。
ダイナミックなサウンド再生: 特定の SoundTag 識別子に関連付けられたオーディオクリップを再生し、利用可能なオプションからランダムに選択します。
インスタンスのトラッキング: 各サウンドの現在の再生インスタンス数を追跡し、同時実行制限を強制します。
バックグラウンドサウンドの制御: バックグラウンド音楽のオン/オフを切り替え、対応するゲームオブジェクト（m_SoundOnObj および m_SoundOffObj）を使用してビジュアルフィードバックを提供します。
Sound クラス:

サウンドの定義: 一意の識別子（m_Id）、複数の AudioClip のバリエーション（m_Clips）、および同時に再生できる最大インスタンス数（m_MaxInstances）を定義します。
インスタンスの追跡: サウンドの現在のアクティブインスタンス数（m_CurrentInstances）を追跡し、同時実行制限を実施します。
ワークフロー:
初期化: Start メソッドで、SoundManager を初期化し、辞書を準備し、m_audioSourceSize と m_audioSourceVolume に基づいて AudioSource コンポーネントのプールを作成します。

PlaySound メソッド: 指定された SoundTag に関連付けられたサウンドを再生し、利用可能なオーディオクリップからランダムに選択します。再生可能な最大インスタンス数に達している場合は再生しません。

GetAvailableAudioSource メソッド: サウンドクリップの再生に使用するための利用可能な AudioSource をプールから取得します。

ResetSoundInstanceCount コルーチン: サウンドクリップの再生が終了した後、そのサウンドのインスタンス数を正確に追跡するためにインスタンス数をリセットします。

ToggleBackGroundSound メソッド: バックグラウンド音楽のオン/オフを切り替え、ビジュアルフィードバックとして関連するゲームオブジェクト（m_SoundOnObj および m_SoundOffObj）を制御します。オフに切り替えた場合、現在再生中の音楽を停止します。

StopSound メソッド: 指定された SoundTag に関連付けられたすべてのインスタンスを停止し、そのインスタンス数をリセットします。

実装の理由:
SoundManager は、オブジェクトプーリングを使用して AudioSource コンポーネントを効率的に管理し、動的に作成および破棄することに比べてパフォーマンスのオーバーヘッドを最小限に抑えています。このアプローチにより、スムーズなオーディオ再生が実現され、ゲーム内で過剰な AudioSource インスタンスが作成されるのを防ぎ、最適化されたゲームプレイ体験に貢献しています。

ServiceLocator Class

Overview :

The ServiceLocator class implements the Service Locator pattern, providing a centralized mechanism for managing and accessing global instances of critical services within the Unity application. This pattern promotes loose coupling between components by decoupling them from the direct references to their dependencies.

Purpose

The primary purpose of the ServiceLocator is to facilitate:

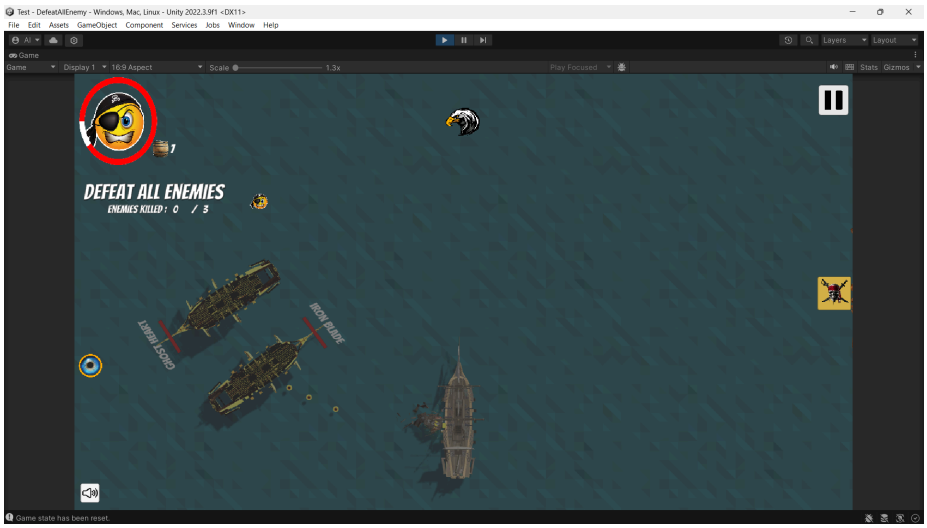
- **Dependency Management**: Managing and providing access to essential services such as object pooling, enemy management, sound management, and stage control across various parts of the Game.

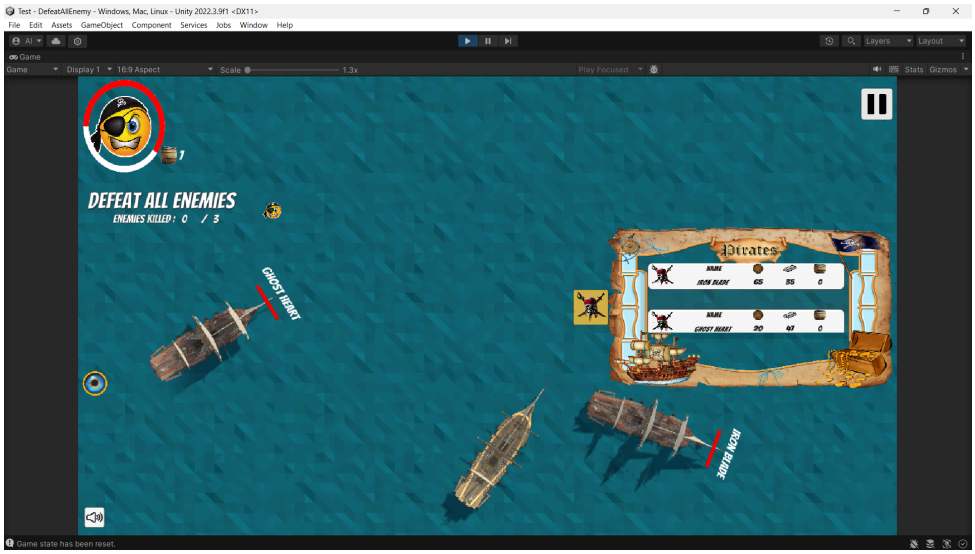
- **Global Accessibility:** Allowing components to access these services without explicit knowledge of their concrete implementations, promoting reusability and maintainability.

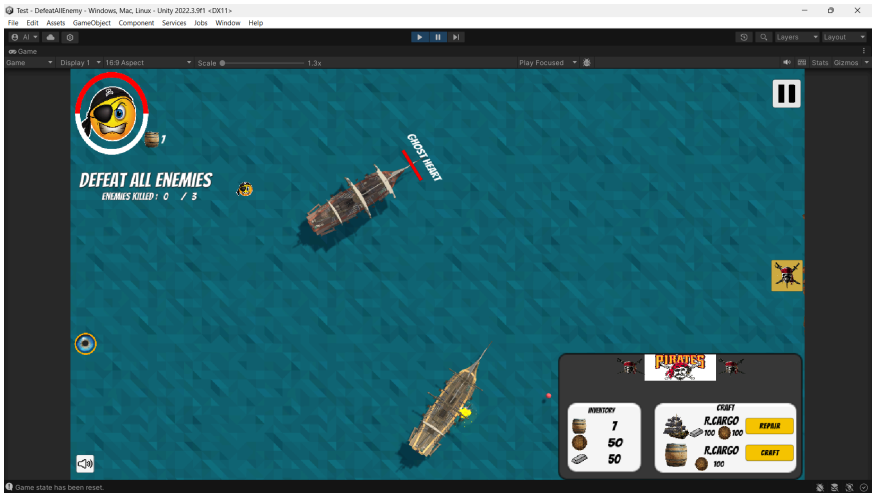
Features

- **Registration Methods**: Provides methods to register instances of various managers or services:

- **Event Handling:** Includes event mechanisms (OnEnemyManagerRegistered) to notify components when the enemyManager has been registered, facilitating synchronization and initialization of dependent systems.




DEFEAT ALL ENEMIES
ENEMIES KILLED : 0 / 3
GHOST HEART
IRON BLADE
Pirates


DEFEAT ALL ENEMIES
ENEMIES KILLED : 0 / 3
GHOST HEART
INVENTORY
CRAFT
REPAIR
CRAFT


GAMEMODE
CONTROLS
QUIT
PIRATES