Dokumentacja Tower Defence

1. **Opis folderów**

**Animations** – wszystkie animacje w grze

**Fonts** – wykorzystywane czcionki

**Materials** – tekstury nakładane na modele 3D

**Models** – trójwymiarowe modele obiektów

**Prefabs** – gotowe obiekty gry

**Scenes** – sceny/poziomy gry

**Scripts** – wszelki kod zawarty w grze

**Sprites** – dwuwymiarowe tekstury

1. **Opis składowych poziomu**

**Main camera** – pokazuje pole widzenia gracza. Skrypt odpowiadający za jej działanie: CameraController.cs

**Directional Light** – oświetlenie poziomu

**GameMaster (pkt 3)** – komponent zarządzający działaniem gry.

**UI** – Przechowuje wszystkie elementy interfejsu użytkownika:

*- UI\_3D\_Texts* – Trójwymiarowe elementy interfejsu, przemieszczające się wraz z kamerą

*- UI\_Always\_Visible* – dwuwymiarowe elementy interfejsu, zawsze widoczne na ekranie.

**LevelDesign** – Przechowuje wszystkie elementy poziomu:

- *Nodes* – pola na których można stawiać wieżyczki

- *Environment* – Elementy otoczenia np ścieżki, drzewa.

- *Waypoints* – punkty nawigacyjne, wzdłuż których przemieszczają się wrogie jednostki.

- *START* – Punkt spawnu wrogich jednostek

- *END* – Punkt, w którym wrogie jednostki kończą ścieżkę.

1. **GameMaster**

Zarządza działaniem gry. Skrypty podpięte do gamemastera:

- **WaveSpawner.cs** – zarządza spawnowaniem się wrogich jednostek. Decyduje o ilości jednostek, czasie. Pola klasy:

*EnemyPrefab* – Należy podpiąć obiekt wroga Enemy.prefab

*SpawnPoint* – Wskazuje pozycję spawnu wroga: podpiąć START.Prefab

*TimeBetweenWaves* – czas po upływie którego pojawi się kolejna fala wrogów.

*WaveCountdownText –* element intefejsu do wyświetlania czasu. Podpiąć *Timer* z UI->UI\_3D\_Texts\_>Timer

- **BuildManager.cs** – zarządza budowaniem wież. Pola:

*BuildEffect* – przechowuje wizualny efekt jaki pojawia się po pomyślnym zbudowaniu wieży. Przykładowy efekt: Assets->Prefabs->Level Elements->BuildEffect.prefab

- **PlayerStats.cs** – informacje o graczu. Pola:

*StartMoney* – początkowy stan pieniędzy gracza.

1. **UI**

*UI\_3D\_Texts* – zawiera text Money z podpiętym skryptem MoneyUI.cs, aktualizującym stan pieniędzy gracza na interfejsie oraz Timer, którego wyświetlaniem zarządza GameMaster.

*UI\_Always\_Visible* – zawiera Shop, który ma podpięty skrypt Shop.cs. Shop.cs zawiera pola typu TurretBlueprint, które przechowują informacje o prefabie wieżyczki oraz jej koszcie zakupu. Niżej w hierarchii znajdują się elementy wizualne wież.

1. **LevelDesign**

*Nodes* – wszystkie platformy na scenie, na których można budować wieże. Każdy Node to oddzielny Node.prefab z podpiętym skryptem NodeController.cs, który zarządza wizualną częścią budowania w tym podświetlania Noda na różne kolory zależnie od warunków.

*Environment* – nic innego jak wizualne elementy poziomu. Nie ma żadnych skryptów.

*Waypoints* – zawiera punkty, po których przemieszczają się wrogowie. Ma podpięty skrypt Waypoints.cs, który przechowuje wszystkie Waypointy znajdujące się w hierarchii jako listę. Ważna jest kolejność waypointów w hierarchii.

*START, END* – obiekty wizualne niezawierające żadnych skryptów, jednak zostały stworzone jako prefaby i są przechowywane w folderze Prefabs->Level Elements.

1. **PlayerPrefs (type, default value):**

**Money (int, 0) –** przechowuje kwote pieniędzy zgromadzoną na koncie gracza.

**levelReached (int, 0) –** przechowuje najwyższy poziom w rozgrywce, jaki udało się osiągnąć graczowi.

**fullItemNameLvl (int, 1)** – przechowuje aktualny poziom ulepszenia danego parametru wieży: fullItemName to np: **StandardTurretRange, MissieLauncherFireRate**

**Turrets:**

**StandardTurretRange (float, 15)**

**StandardTurretFireRate (float, 1)**

**StandardTurretUpgradedRange (float, 20)**

**StandardTurretUpgradedFireRate (float, 1.5)**

**MissileLauncherRange (float, 25)**

**MissileLauncherFireRate (float, 0.5)**

**MissileLauncherUpgradedRange (float, 30)**

**MissileLauncherUpgradedFireRate (float, 0.7)**

**LaserBeamerRange (float, 20)**

**LaserBeamerDamageOverTime (int, 30)**

**LaserBeamerSlowPercentage (float, 0.4)**

**LaserBeamerUpgradedRange (float, 25)**

**LaserBeamerUpgradedDamageOverTime (int, 40)**

**LaserBeamerUpgradedSlowPercentage (float, 0.55)**

**Bullets:**

**StandardBulletDamage (int, 50)**

**StandardBulletSpeed (float, 60)**

**StandardBulletUpgradedDamage (int, 60)**

**StandardBulletUpgradedSpeed (float, 65)**

**MissileDamage (int, 35)**

**MissileSpeed (float, 30)**

**MissileExplosionRadius (float, 5)**

**MissileUpgradedDamage (int, 40)**

**MissileUpgradedSpeed (float, 40)**

**MissileUpgradedExplosionRadius (float, 8)**

1. **dsada**