**Buffet Blast API Documentation “Sample”**

*Version 0.1.2d*

**Enumerators**

* [EFoodType](#k9yke85iidk)
* [ESpawnInstruction](#4rhoicr8wa52)

**Interfaces**

* [IFood](#to4yptau3f8o)
* [ISpawner](#i1pbyg6yaghm)
* [ICollectable](#4crqazwgm45j)
* [IPowerup](#5ziiqoqsln9j)
* [IPowerupable](#fvk9zz8otg1e)
* [IPoolableObject](#f5p8fjflhp2p)

**Classes**

* [Character](#texeurm5yuji)
* [CharStats](#eva3tp2ybnqe)
* [Food](#ddoizgg5hv4n)
* [FoodSpawner\_MicroPool](#k2tynf1b0mvh)
* [FoodSpawner\_ObjectPool](#ymwiabggaf08)
* [FoodSpawner\_Single](#l3xamexswl4z)
* [FoodSpawner](#7dvxvqqh5fn9)
* [Plate](#5zkbgvwt1c2q)
* [Level](#xpkxqboxd6a9)
* [ObjectPool](#115mlbpzqv2i)
* [ObjectPoolManager](#cvdrvopzz3rt)
* [PoolableObject](#cp084wweuz3)

**EFoodType**

* Summary
  + Fixed values that define food
* Members
  + Small
    - 1 (Defined by the GDD)
  + Medium
    - 2 (Defined by the GDD)
  + Large
    - 4 (Defined by the GDD)

**ESpawnInstruction**

* Summary
  + Values that define a spawner’s spawn behavior.
* Remarks
  + Specific will always spawn the first object listed (Depreciated).
  + Random will select any object in the list.
  + Round Robin will spawn objects in their listed order (Depreciated).
  + Sequential replaces Round Robin.
* Members
  + Specific
  + Random
  + RoundRobin
  + Sequential

**IFood**

* Summary
  + Interface for food objects.
* Remarks
* Methods
  + GetFoodType, EFoodType
    - Summary
      * Returns the type of food this object is
  + SetSpawnerParent, void
    - Summary
      * Sets which spawner this object came from.
    - Params
      * ISpanwer parent

**ISpawner**

* Summary
  + Interface for spawner objects.
* Remarks
* Methods
  + ClearSpawn, void
    - Summary
      * Resets the last object that spawned from this spawner.

**ICollectable**

* Summary
  + Interface for collectable objects.
* Methods
  + Collect, void
    - Summary
      * Routine that processes acquisition of this object.
    - Remarks
      * Depreciated.

**IPowerup**

* Summary
  + Interface for using and interacting with power ups.
* Methods
  + Activate, void
    - Summary
      * Activates the effect of whatever this powerup does and applies its side effects to a given game object.
    - Params
      * GameObject affectedObject

**IPowerupable**

* Summary
  + Interface for objects that use power ups.
* Remarks
* Methods
  + OnPowerupActivate, void, <T>
    - Summary
      * Event method that triggers when a powerup is used.
    - Remarks
      * Depreciated.

**IPoolableObject**

* Summary
  + Interface for objects that are part of an object pool.
* Remarks
  + Depreciated.
* Methods
  + OnCheckOut, void
    - Summary
      * Routine that runs when this game object is checked out from the pool. The initial state of all the components on this game object are journaled here.
  + OnReturn, void
    - Summary
      * Routine that runs when this game object is returned to the pool. All components are reset to their initial state here.
  + IsCheckedOut, bool
    - Summary
      * Is this object currently checked out from the pool?

**Character**

* Summary
  + Character data structure that contains button logic for food interaction.
* Parent Class
  + UnityEngine.MonoBehaviour
* Interfaces
* Constants
  + DEBUG, bool
    - Summary
      * Debug flag for switching debug code.
* Properties
* Fields
  + interactionRange, public float
    - Summary
      * The distance at which a character can interact with a power up.
  + canGrabPowerUp, public bool
    - Remarks
      * Flagged as false when a character currently has a power up.
  + powerup\_, IPowerup
    - Summary
      * Ref to a power up that is within interaction range.
  + food\_, Food
    - Summary
      * Food object that is within interaction range.
* Static Members
* Constructor
  + Summary
    - Unity default.
* Static Methods
* Public Methods
* Protected Methods
* Private Methods
  + Start, void, inherited
    - Remarks
      * Initializes a box collider that acts as a character’s interaction range.
  + Update, void, inherited
    - Remarks
      * Processes player input for interacting with various objects.
  + OnTriggerStay, void inherited
    - Remarks
      * Acquires a powerup from the Collider object.
    - Params
      * Collider c
  + OnTriggerEnter, void, inherited
    - Remarks
      * Acquires a Food object from the Collider object and save a reference.
    - Params
      * Collider c
  + OnTriggerEnter, void, inherited
    - Remarks
      * Releases the Food reference of the Collider object.
    - Params
      * Collider c

**CharStats**

* Summary
  + Component for character objects that track character statistics.
* Remarks
* Parent Class
  + UnityEngine.MonoBehaviour
* Interfaces
* Constants
* Static Members
* Fields
  + ScoreText, public Text
    - Summary
      * Ref to a Unity canvas object that displays the current score.
  + score\_, private int
    - Summary
      * Record of the current score.
  + explosions\_, int
    - Summary
      * Record of the current number of explosions
  + platesUsed\_, float
    - Summary
      * Record of the current number of plates used.
  + foodCollected\_, int
    - Summary
      * Record of the current number of food objects acquired.
  + powerupsCollected\_, int
    - Summary
      * Record of the current number of power ups acquired.
* Properties
* Constructor
  + Summary
    - Unity default.
* Static Methods
* Public Methods
  + AddScore, void
    - Summary
      * Adds the given value to score.
    - Params
      * int val
  + SubScore, void
    - Summary
      * Subtracts the given value form score.
    - Params
      * int val
  + GetScore, int
    - Summary
      * Returns the current score.
  + AddExplosion, void
    - Summary
      * Adds 1 to number of explosions.
  + AddPlateFraction, void
    - Summary
      * Adds a fraction of a plate based on food value to the number of plates used.
    - Remarks
      * Useful if fractions of a plate count.
      * Needs a plate to determine if the it’s full.
    - Params
      * [Plate](#5zkbgvwt1c2q)
  + AddPlateWhole, void
    - Summary
      * Adds 1 to plates used.
  + AddFood, void
    - Summary
      * Adds 1 to food collected.
  + AddPowerup, void
    - Summary
      * Adds 1 to power ups collected.
  + GetExplosions, int
    - Summary
      * Returns the number of explosions.
  + GetPlatesUsed, float
    - Summary
      * Returns the number of plates used.
  + GetFoodCollected, int
    - Summary
      * Returns the number of food collected.
  + GetPowerupsCollected, int
    - Summary
      * Returns the number of power ups collected.
* Protected Methods
* Private Methods

**Food**

* Summary
  + Component used to identify what type of food this object is.
* Parent Class
  + UnityEngine.MonoBehaviour
* Interfaces
  + IFood
* Constants
* Static Members
* Fields
  + foodType, public [EFoodType](#k9yke85iidk)
    - Summary
      * The type of food for this item.
    - Remarks
      * Assigned in the inspector.
  + parentSpawner, private [ISpawner](#i1pbyg6yaghm)
    - Summary
      * Reference to the spawner that created this item.
  + audioSource, private AudioSource
    - Summary
      * Plays audio cue when this object is collected.
* Properties
  + ParentSpawner, [ISpawner](#i1pbyg6yaghm)
    - Summary
      * Gets or sets the parent spawner of this object.
  + GetFoodType, [EFoodType](#k9yke85iidk)
    - Summary
      * Returns the food type.
* Constructor
  + Summary
    - Unity default.
* Public Methods
  + AddToPlate, void
    - Summary
      * Takes a plate ref as input, adds this food's value to the plate, and cleans up the food's parent spawner.
    - Params
      * ref [Plate](#5zkbgvwt1c2q) plate
  + OnCheckOut, void override
    - Summary
      * Inherited from [PoolableObject](#cp084wweuz3).
    - Remarks
      * Depreciated.
    - Params
      * Vector3 position
  + OnReturn, void override
    - Summary
      * Inherited from [PoolableObject](#cp084wweuz3)
    - Remarks
      * Depreciated.
  + SetSpawnerParent, void
    - Summary
      * Inherited from [IFood](#to4yptau3f8o).
    - Params
      * [ISpawner](#i1pbyg6yaghm) spawner
* Protected Methods
* Private Methods

**FoodSpawner\_MicroPool**

* Summary
  + Spawner class that creates food items at wherever this object's location is.
* Remarks
  + This spawner class is designed to spawn one item at a time. It will not spawn a new item until its previously spawned item has been removed.
  + Modified Food Spawner that utilizes a self encapsulated object pool for spawning objects.
* Parent Class
  + UnityEngine.MonoBehaviour
* Interfaces
  + [ISpawner](#i1pbyg6yaghm)
* Constants
  + DEBUG, bool
    - Summary
      * Debug flag for switching debug code.
* Static Members
* Fields
  + spawnStartDelay, public [NanoTimers.Time](https://drive.google.com/open?id=1kRCUKxN29wzUnP9nFZWcsgXjx_qpoyw81qvdWJTCQOs)
    - Summary
      * Amount of time this spawner will delay at start before spawning.
    - Remarks
      * Assigned in editor.
  + spawnTimer, public [NanoTimers.Time](https://drive.google.com/open?id=1kRCUKxN29wzUnP9nFZWcsgXjx_qpoyw81qvdWJTCQOs)
    - Summary
      * Amount of time this spawner will wait before spawning a new object.
    - Remarks
      * Assigned in editor.
  + canSpawnAtStart, public bool
    - Summary
      * Does this spawner spawn something at start?
    - Remarks
      * If otherwise false, food will spawn after spawn timer has first expired. This does nothing if start delay timer has been assigned.
      * Assigned in editor.
  + spawnInstruction, public [ESpawnInstruction](#4rhoicr8wa52)
    - Summary
      * How the spawner spawns objects in it’s pool.
    - Remarks
      * Assigned in editor.
  + spawnableFood, public GameObject[]
    - Summary
      * List of prototypes that are cloned to spawnPool.
    - Remarks
      * Assigned in editor.
  + spawnedIndex, private int
    - Summary
      * Current index of the spawn pool this spawner is at.
  + spawnedFood, private GameObject
    - Summary
      * Currently spawned object from the spawn pool.
    - Remarks
      * Is set to null when a character has acquired the object.
  + timer, private [CountdownTimer](https://drive.google.com/open?id=1kRCUKxN29wzUnP9nFZWcsgXjx_qpoyw81qvdWJTCQOs)
    - Summary
      * Timer that counts down to the next spawn instance.
  + spawnPool, private GameObject[]
    - Summary
      * List of cloned objects from spawnableFood that the spawner activates when spawned.
  + isInitialized, private bool
    - Summary
      * Is true when spawnPool has been successfully initialized.
    - Remarks
      * Flag is set from InitializeSpawnPool
* Properties
* Constructor
* Static Methods
* Public Methods
  + ClearSpawn, void
    - Summary
      * Clears the previous food reference and restarts the timer.
    - Remarks
      * Is invoked from the Food component on the food object this spawner created.
* Protected Methods
* Private Methods
  + OnStartDelayExpire, void
    - Summary
      * Initializes the spawn timer after delay timer has finished.
  + InitializeSpawnPool, GameObject[]
    - Summary
      * Pure, recursive function that returns an initialized pool of spawnable food.
    - Remarks
      * Will return an empty array if prefabList is empty.
    - Params
      * int i
      * GameObject[] prefabList
      * GameObject[] spawnPool
  + ChangeActiveState, void
    - Summary
      * Recursively changes the active state of all game objects in the provided list.
    - Remarks
      * Designed to prevent multiple objects in the spawn pool from being active at one time.
    - Params
      * int i
      * GameObject[] spawnPool
      * bool isActive
  + Spawn, void
    - Summary
      * Creates food based on given spawn instruction for which order in the list to spawn.
  + Start, void, inherited
    - Summary
      * Initializes the spawn pool and the spawn timer.

**FoodSpawner\_ObjectPool**

* Summary
  + Spawner class that creates food items at wherever this object's location is.
* Remarks
  + This spawner class is designed to spawn one item at a time. It will not spawn a new item until its previously spawned item has been removed.
  + Modified Food Spawner that utilizes a self encapsulated object pool for spawning objects.
  + Depreciated.
* Parent class
  + UnityEngine.MonoBehaviour
* Interfaces
  + [ISpawner](#i1pbyg6yaghm)
* Constants
  + DEBUG, bool
    - Summary
      * Debug flag for switching debug code.
* Static Members
* Fields
  + spawnStartDelay, public [NanoTimers.Time](https://drive.google.com/open?id=1kRCUKxN29wzUnP9nFZWcsgXjx_qpoyw81qvdWJTCQOs)
    - Summary
      * Amount of time this spawner will delay before spawning.
    - Remarks
      * Assigned in editor.
  + spawnTimer, public [NanoTimers.Time](https://drive.google.com/open?id=1kRCUKxN29wzUnP9nFZWcsgXjx_qpoyw81qvdWJTCQOs)
    - Summary
      * Amount of time this spawner will wait before spawning a new object.
    - Remarks
      * Assigned in editor.
  + canSpawnAtStart, public bool
    - Summary
      * Does this spawner spawn something at start?
    - Remarks
      * If otherwise false, food will spawn after spawn timer has first expired. This does nothing if start delay timer has been assigned.
      * Assigned in editor.
  + spawnInstruction, public [ESpawnInstruction](#4rhoicr8wa52)
    - Summary
      * How the spawner spawns objects in it’s pool.
    - Remarks
      * Assigned in editor.
  + spawnableFood, public [EFoodType](#k9yke85iidk)[]
    - Summary
      * List of prototypes that are cloned to spawnPool.
    - Remarks
      * Assigned in editor.
  + roundRobinIndex, private int
    - Summary
      * Current index of the spawn pool this spawner is at.
  + spawnedFood, private [PoolableObject](#cp084wweuz3)
    - Summary
      * Ref of the currently spawned object from the spawn pool.
    - Remarks
      * Is set to null when a character has acquired the object.
  + timer, private [CountdownTimer](https://drive.google.com/open?id=1kRCUKxN29wzUnP9nFZWcsgXjx_qpoyw81qvdWJTCQOs)
    - Summary
      * Timer that counts down to the next spawn instance.
* Properties
* Constructor
* Static Methods
* Public Methods
  + ClearSpawn, void
    - Summary
      * Clears the previous food reference and restarts the timer.
    - Remarks
      * Is invoked from the Food component on the food object this spawner created.
* Protected Methods
* Private Methods
  + OnStartDelayExpire\_, void
    - Summary
      * Initializes the spawn timer after delay timer has finished.
  + Spawn\_, void
    - Summary
      * Creates food based on given spawn instruction for which order in the list to spawn.
  + Start, void
    - Summary
      * Initializes the spawn timer.

**FoodSpawner\_Single**

* Summary
  + Spawner class that creates food items at wherever this object's location is.
* Remarks
  + Spawns only one type.
  + This spawner class is designed to spawn one item at a time. It will not spawn a new item until its previously spawned item has been removed.
* Parent class
  + UnityEngine.MonoBehaviour
* Interfaces
  + [ISpawner](#i1pbyg6yaghm)
* Constants
  + DEBUG, bool
    - Summary
      * Debug flag for switching debug code.
* Static Members
* Fields
  + spawnStartDelay, public [NanoTimers.Time](https://drive.google.com/open?id=1kRCUKxN29wzUnP9nFZWcsgXjx_qpoyw81qvdWJTCQOs)
    - Summary
      * Amount of time this spawner will delay at start before spawning.
    - Remarks
      * Assigned in editor.
  + spawnTimer, public [NanoTimers.Time](https://drive.google.com/open?id=1kRCUKxN29wzUnP9nFZWcsgXjx_qpoyw81qvdWJTCQOs)
    - Summary
      * Amount of time this spawner will wait before spawning a new object.
    - Remarks
      * Assigned in editor.
  + canSpawnAtStart, public bool
    - Summary
      * Does this spawner spawn something at start?
    - Remarks
      * If otherwise false, food will spawn after spawn timer has first expired. This does nothing if start delay timer has been assigned.
      * Assigned in editor.
  + foodPrefab, public GameObject
    - Summary
      * Prototype food object that gets cloned.
    - Remarks
      * Assigned in editor.
  + foodInstance, private GameObject
    - Summary
      * Ref of the currently spawned object.
    - Remarks
      * This object stays in memory and is never destroyed but instead turned off when undesired.
  + spawnedFood, private GameObject
    - Summary
      * Ref of the currently spawned object from the spawn pool.
    - Remarks
      * This ref gets set to foodInstance when it’s active.
  + timer, private [CountdownTimer](https://drive.google.com/open?id=1kRCUKxN29wzUnP9nFZWcsgXjx_qpoyw81qvdWJTCQOs)
    - Summary
      * Timer that counts down to the next spawn instance.
* Properties
* Constructor
* Static Methods
* Public Methods
  + ClearSpawn, void
    - Summary
      * Clears the previous food reference and restarts the timer.
    - Remarks
      * Is invoked from the Food component on the food object this spawner created.
* Protected Methods
* Private Methods
  + OnStartDelayExpire, void
    - Summary
      * Initializes the spawn timer after delay timer has finished.
  + Spawn, void
    - Summary
      * Creates food based on given spawn instruction for which order in the list to spawn.
  + Start, void, inherited
    - Summary
      * Initializes the spawn pool and the spawn timer.

**FoodSpawner**

* Summary
  + Spawner class that creates food items at wherever this object's location is.
* Remarks
  + This spawner class is designed to spawn one item at a time. It will not spawn a new item until its previously spawned item has been removed.
  + Depreciated in favor of [FoodSpawner\_Single](#l3xamexswl4z) and [FoodSpawner\_MicroPool](#k2tynf1b0mvh).
* Parent class
  + UnityEngine.MonoBehaviour
* Interfaces
  + [ISpawner](#i1pbyg6yaghm)
* Constants
  + DEBUG, bool
    - Summary
      * Debug flag for switching debug code.
* Static Members
* Fields
  + spawnStartDelay, public [NanoTimers.Time](https://drive.google.com/open?id=1kRCUKxN29wzUnP9nFZWcsgXjx_qpoyw81qvdWJTCQOs)
    - Summary
      * Amount of time this spawner will delay before spawning.
    - Remarks
      * Assigned in editor.
  + spawnTimer, public [NanoTimers.Time](https://drive.google.com/open?id=1kRCUKxN29wzUnP9nFZWcsgXjx_qpoyw81qvdWJTCQOs)
    - Summary
      * Amount of time this spawner will wait before spawning a new object.
    - Remarks
      * Assigned in editor.
  + canSpawnAtStart, public bool
    - Summary
      * Does this spawner spawn something at start?
    - Remarks
      * If otherwise false, food will spawn after spawn timer has first expired. This does nothing if start delay timer has been assigned.
      * Assigned in editor.
  + spawnInstruction, public [ESpawnInstruction](#4rhoicr8wa52)
    - Summary
      * How the spawner spawns objects in it’s pool.
    - Remarks
      * Assigned in editor.
  + spawnableFood, public GameObject[]
    - Summary
      * List of prototypes that are cloned to spawnPool.
    - Remarks
      * Assigned in editor.
  + roundRobinIndex, private int
    - Summary
      * Current index of the spawn pool this spawner is at.
  + spawnedFood, private GameObject
    - Summary
      * Ref of the currently spawned object from the spawn pool.
    - Remarks
      * Is set to null when a character has acquired the object.
  + timer, private CountdownTimer
    - Summary
      * Timer that counts down to the next spawn instance.
* Properties
* Constructor
  + Summary
    - Unity default.
* Public Methods
  + ClearSpawn, void
    - Summary
      * Clears the previous food reference and restarts the timer.
    - Remarks
      * Is invoked from the Food component on the food object this spawner created.
* Protected Methods
* Private Methods
  + OnStartDelayExpire\_, void
    - Summary
      * Initializes the spawn timer after delay timer has finished.
  + Spawn\_, void
    - Summary
      * Creates food based on given spawn instruction for which order in the list to spawn.
  + Start, void, inherited
    - Summary
      * Initializes the spawn timer.

**Plate**

* Summary
  + Plate component for keeping track of how much food value a character has collected.
* Remarks
* Parent Class
  + UnityEngine.MonoBehaviour
* Interfaces
* Constants
  + DEBUG, bool
    - Summary
      * Debug flag for switching debug code.
  + MAXVAL, int
    - Summary
      * Maximum possible aggregate value of food this plate can hold as defined by the GDD.
* Static Members
* Fields
  + totalValue, private int
    - Summary
      * Current aggregate value of food this plate has.
  + tomatoHitValue, public int
    - Summary
    - Remarks
      * Assigned in editor
  + plateCells, public GameObject[]
    - Summary
      * References to Unity canvas objects that indicate food value on this plate.
    - Remarks
      * Assigned in editor.
  + voidColor, public Color32
    - Summary
      * Represents an empty plate cell.
    - Remarks
      * Assigned in editor
  + fillColor, public Color32
    - Summary
      * Represents an occupied plate cell.
    - Remarks
      * Assigned in editor.
* Properties
  + MaxValue, int
    - Summary
      * Returns the max value this plate can hold.
  + TotalValue, int
    - Summary
      * Returns the current value of this plate.
  + IsPlateFull
    - Summary
      * Returns true if the plate is full.
  + IsPlateEmpty
    - Summary
      * Returns true if the plate is empty.
* Properties
* Constructor
  + Summary
    - Unity default.
* Static Methods
  + ConstructPlateBar, GameObject[]
    - Summary
      * Plate bar factory that appends a plate bar to which ever object is given and returns a list of bar segments.
    - Remarks
      * Testing purposes
      * Depreciated.
    - Params
      * GameObject parent
* Public Methods
  + AddValue, bool
    - Summary
      * Adds value to the plate while not exceeding the maximum limit.
    - Remarks
      * Returns whether or not the value has been successfully added to the plate.
    - Params
      * int val
  + RemoveSingleValue, void
    - Summary
      * Subtracts the plate’s current value by 1.
    - Remarks
      * Depreciated.
      * Use RemoveValue.
  + RemoveValue, void
    - Summary
      * Removes a variable number of segments from the plate bar.
    - Params
      * int val
  + ResetTotalValue, void
    - Summary
      * Sets the current value of this plate to 0.
  + TomatoHit, void
    - Summary
    - Remarks
* Protected Methods
* Private Methods
  + UpdateCellColor, void
    - Summary
      * Changes the color of a given list of game objects that contain an Image component.
    - Remarks
      * Every item in the list is changed to the given void color then changed to the given fill color based on the total plate value.
    - Params
      * GameObject[] plateCells
      * int val
      * Color32 voidColor
      * Color32 fillColor

**Level**

* Summary
  + Overseer class that monitors and executes level-wide events and maintains references to core game objects.
* Remarks
  + Depreciated.
* Parent class
  + UnityEngine.MonoBehaviour
* Interfaces
* Constants
* Fields
  + characters, public GameObject[]
  + roundTime, public [NanoTimers.Time](https://drive.google.com/open?id=1kRCUKxN29wzUnP9nFZWcsgXjx_qpoyw81qvdWJTCQOs)
  + startTimeDelay, public [NanoTimers.Time](https://drive.google.com/open?id=1kRCUKxN29wzUnP9nFZWcsgXjx_qpoyw81qvdWJTCQOs)
  + endTimeDelay, public [NanoTimers.Time](https://drive.google.com/open?id=1kRCUKxN29wzUnP9nFZWcsgXjx_qpoyw81qvdWJTCQOs)
  + rounderTimerText, public Text
  + timer\_, private [CountdownTimer](https://drive.google.com/open?id=1kRCUKxN29wzUnP9nFZWcsgXjx_qpoyw81qvdWJTCQOs)
* Properties
* Static Methods
* Public Methods
* Private Methods
  + Start, void, inherited
  + OnStartTimerExpire\_, void
  + OnRoundTimerExpire\_, void
  + OnEndTimer\_, void

**ObjectPool**

* Summary
  + Object pooling class that stores pre instantiated foods for use with food spawners.
* Remarks
  + Designed to prevent memory fragmentation associated with instantiating and deleting objects. Any and all objects that are used throughout the game's lifetime will be switching objects on and off.
  + Depreciated in favor of managing self encapsulated pools within food spawners themselves. Shared state of objects flagged for checkout became too risky.
* Parent class
* Interfaces
* Constants
* Constructor
  + Summary
  + Remarks
    - Initializes the object pool by instantiating a given number of game objects.
  + Params
    - int initialSize
    - GameObject[] prefabList
* Fields
  + m\_list, private [PoolableObject](#cp084wweuz3)
  + m\_poolId, private System.Guid
* Properties
* Static Methods
* Public Methods
  + CheckOut, [PoolableObject](#cp084wweuz3)
    - Summary
      * Returns the first Poolable OBject that is not currently checked out.
    - Remarks
    - Params
      * [EFoodType](#k9yke85iidk) foodType
      * Vector3 position
  + Return, void
    - Summary
      * Returns the given object to the object pool if it's ID matches.
    - Remarks
    - Params
      * [PoolableObject](#cp084wweuz3) obj
* Private Methods

**ObjectPoolManager**

* Summary
  + MonoBehaviour class that grants editor access to object pool initialization.
* Remarks
  + Depreciated.
* Parent class
  + UnityEngine.MonoBehaviour
* Interfaces
* Constants
* Constructor
  + Summary
    - Unity default.
* Static Members
  + fodoPool, [ObjectPool](#115mlbpzqv2i)
* Fields
  + prefabList, public GameObject[]
  + prefabAmount, public int
* Properties
* Static Methods
* Public Methods
* Private Methods
  + Start, void, inherited
    - Summary
      * Initializes object pools.

**PoolableObject**

* Summary
  + Enables game objects to be compatible with an object pool.
* Remarks
* Parent class
  + UnityEngine.MonoBehaviour
* Interfaces
* Constants
* Constructor
  + Summary
    - Unity default
* Static Members
* Fields
  + id, protected int
  + poolId, protected System.Guid
  + isCheckedOut, protected bool
* Properties
  + Id, int
    - Summary
      * Returns the inner pool id.
  + PoolId, System.Guid
    - Summary
      * Returns the set id.
  + IsCheckedOut, bool
    - Summary
      * Returns true if this object is checked out.
* Constructor
  + Summary
    - Unity default.
* Static Methods
* Public Methods
  + Initialize, virtual void
    - Summary
      * Setup routine during pool initialization.
    - Params
      * int id
      * System.Guid poolId
  + OnCheckOut, virtual void
    - Params
      * Vector3 position
  + OnReturn, virtual void
* Protected Methods
* Private Methods