Fast Pool

Scripting documentation

FastPoolManager.cs

Information

Used for easy managing pools. Can automatically create pools for the new objects.
For further information on using FastPoolManager look at the example scenes.

Public Properties

```
static FastPoolManager Instance      // Singletone instance

Dictionary<int, FastPool> Pools      // List of managed runtime pools
```

Methods

Create New Pool

```
// Create a new pool from provided component
public static FastPool CreatePoolC<T>(T component, bool warmOnLoad = true,
                                      int preloadCount = 0, int capacity = 0)



// Create a new pool from provided prefab
public static FastPool CreatePool(GameObject prefab, bool warmOnLoad = true,
                                  int preloadCount = 0, int capacity = 0)
```

Get Existing Pool Or Create New (with default settings)

```
// Returns pool for the specified prefab or creates it if needed (with default params)
public static FastPool GetPool(GameObject prefab, bool createIfNotExists = true)



// Returns pool for the specified prefab or creates it if needed (with default params)
public static FastPool GetPool(Component component, bool createIfNotExists = true)
```

Destroy Pool

```
// Destroys provided pool and it's cached objects
public static void DestroyPool(FastPool pool)
```

FastPool.cs

Information

Main pool class. Recycle objects for later use. Can be managed by FastPoolManager or directly from your scripts.
For further information on using FastPool look at the example scenes.

Public Properties

```
int ID                          // ID of the source Prefab

string Name                     // Name of the source Prefab

int Cached                      // Cached objects count

bool IsValid                    // Shows if the pool has been successfully initialized

GameObject sourcePrefab;        // Prefab that would be used as source

int Capacity;                   // Cache size (maximum amount of the cached items)

int PreloadCount;               // How much items must be cached at the start

bool WarmOnLoad = true;         // Load source prefab in the memory while scene is
                                // loading. A bit slower scene loading, but faster
                                // instantiating of new objects in pool

bool ParentOnCache = false;     // Parent cached objects to FastPoolManager game
                                // object. Note that turning this option on will make
                                // objects cached a bit slower.

PoolItemNotificationType NotificationType;  // How to call events OnFastInstantiate
                                            // and OnFastDestroy
```

Methods

Constructor / Initialization

```
// Creates a new instance of FastPool
public FastPool(GameObject prefab, Transform rootTransform = null,
                bool warmOnLoad = true, int preloadCount = 0, int capacity = 0)


// Initialize pool with current parameters
public bool Init(Transform rootTransform)
```

Generic FastInstantiate

```
// Quickly instantiate GameObject from pool and return provided component from an
// instantiated GameObject.
public T FastInstantiate<T>(Transform parent = null)


// Quickly instantiate GameObject from pool and return provided component from an
// instantiated GameObject.
public T FastInstantiate<T>(Vector3 position, Quaternion rotation,
                            Transform parent = null)
```

FastInstantiate

```
// Quickly instantiate GameObject from pool.
public GameObject FastInstantiate(Transform parent = null)



// Quickly instantiate GameObject from pool.
public GameObject FastInstantiate(Vector3 position, Quaternion rotation,
                                  Transform parent = null)
```

FastDestroy

```
// Disable GameObject of provided Component and put it into the pool for later use. If
// the pool size reached capacity limit - GameObject will be simply Destroyed.
public void FastDestroy<T>(T sceneObject)



// Disable provided GameObject and put it into the pool for later use. If the pool size
// reached capacity limit - GameObject will be simply Destroyed.
public void FastDestroy(GameObject sceneObject)
```

ClearCache

```
// Unload all cached objects from memory
public void ClearCache()
```