Spaces

Sean Middleditch
DigiPen Game Engine Architecture Club

About Myself

DigiPen CSRTIS Senior

GEAC Founder

Games include Subsonic, SONAR, Core

Employed by Gas Powered Games

About This Talk

What are Spaces?

What are they used for?

How to implement them?

Spaces Explained

What Spaces are all about

Before Spaces

Game engines have levels and game objects

A level is loaded and all its corresponding game objects are created

Game objects for previous level are destroyed... maybe

Potentially very bad interaction if more than one level loaded at once

Problem Example

Implement all menus with game objects, load a menu by loading a level

Pause in the middle of a game, then unpause

- How to manage the lifetime of pause menu GOs?
- How to prevent menus from interacting with level content?

First Rejected Solution

Could use a master menu object that tracks all related menu GOs for management

This is error-prone, and doesn't easily deal with GOs spawned after the menu opens without extra work

Doesn't handle separating interaction between GOs that shouldn't interact

Second Rejected Solution

Could use a different **tag** (bitmask) for GOs in each level able to be loaded simultaneously

Limited number of tag values in practical implementations (32, and many will be used for other purposes)

Doesn't easily allow for separating physics, logic, and graphics *control* (even though it allows separating interaction)

Third Rejected Solution

Could just put all objects at wildly different coordinates so they're nowhere near each other

Extremely error-prone

Really really extremely error-prone (I've worked on larger projects that use this method; it doesn't work)

Accepted Solution

Create a collection for game objects

Each collection is independent and isolated from the others; physics objects in one container cannot interact with physics object in another

Containers can be cleared at will

Physics, logic, graphics updates can be explicitly controlled for each container

Spaces

The containers from the accepted solution are called **Spaces**

- Note that this is not a formal term, just the one that Professor Peters popularized

Levels in Spaces

When a level is loaded, a new Space is created for it

If a level is unloaded, simply clear and delete the corresponding Space

Not much more to it

Menus in Spaces

Menus can be levels, and each menu is loaded into its own Space

Menus and regular level content won't interact

Different menu objects won't interact

Menus can remain loaded but the Space can be "switched off" to hide it

Subspaces

Each Space has a collection of **Subspace** objects

These replace or complement the traditional **Systems** of game engines

For example, there would be a PhysicsSubspace, GraphicsSubspace, NetworkSubspace, and so on

Subspaces and Components

Traditional component based game engines have systems manage components

Space based game engines have components managed by Subspaces

Allows updating physics for one Space at a time

Allows opting out of Subspaces

Use Cases for Spaces

More than just level management

Many Use Cases for Spaces

The basic use case is simply object management

There are many more cool things that can be done with game object containers

Physics Properties

When all GOs are managed by a single global engine and systems, it is difficult to give different objects different global properties (like gravity)

Spaces allow multiple isolated simulations to have different "global" properties

Players on the moon have 1.6 m/s^2 gravity while player on Earth have 9.1 m/s^2 gravity

Network Testing

Create two Spaces with networking Subspaces and have them talk to each other

Makes it easier to develop and test the messaging and interaction portions of networking

Obviously does not test real-world bandwidth/latency issues

In-Game Editor Support

Blueprints (aka Templates, Prefabs, Archetypes, etc.) can be spawned in a Space isolate for editing, then serialized back out

Allows to create and modify individual objects with no special-case code

Removes need for a direct Blueprint editor, instead allowing reuse of GO editor

Interesting Uses For Isolated Levels

Preload levels as a player nears a transition, then move the player between Spaces at the transition point, for instant level transitions

Separate ambient background areas from foreground areas in sidescrollers

Easily manage separate areas of a game world, like interior vs exterior "cells" in many RPGs

Insert Your Use Case Here

There are probably many other cool uses of Spaces

They're very rare and aren't available in almost all AAA game engines

The industry has yet to embrace and really explore the applications of Spaces

Making Spaces

Implementation strategies

Systems vs Subspaces

Systems are components of an Engine that handle specific details, e.g. Physics

Subspaces are components of a Space that handle specific details, e.g. Physics

Broadly, you can convert your systems to subspaces and that's it

Game Objects

Game objects are normally registered with a GameObjectFactory on the main Engine

Convert these to be managed by their respective Space instead (or in addition to)

Each GameObject has a reference to its containing Space

Components

Components must register themselves with the appropriate Subspace, since there is no longer a global System

If Systems are also responsible for allocating their respective Components, the Subspace must also do this as well

Component factory looks up reference to Subspace from GameObject's Space

Management

There's no special need to have a special SpaceSystem or global manager

Can be useful to have one "main space" the engine has built-in, possibly a "menu space" too

Can create a special SpaceComponent that contains a Space

Create and update them as needed

That's It

Really, that's all there is to it

Adding them into an existing engine is very easy

Totally worth doing

Summary

Key points

Spaces are Great

Spaces are a straightforward concept

They're easy to implement

Grant lots of power, flexibility, and cleanliness

Questions

Ask away!