DigiPen INSTITUTE OF TECHNOLOGY

CS 175
Advanced Scripting

Object Manager

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved.

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

Introduction

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

Introduction

- All games have multiple objects located throughout the game world, each being updated according to its own defined behavior.
- These objects are saved in a single list called the ***"object list"*** which contains all the object instances of the ***current game state***.
- It is not unusual for an object list to contain hundreds, if not thousands of objects.
- These objects interact with the player and with each other. Ultimately this interaction defines the gameplay.

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

Introduction

- In order to manage all those objects in a clean and consistent way, we created an object manager.

- The beauty of having such a manager is that, once stable and tested, you will have:
  - A consistent way of adding objects
  - A very low risk of getting memory leaks when destroying objects.
  - etc…

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

When is it used?
What should it contain?

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

State's Create

- During game state create:
  - Loading all objects data (Textures, meshes, AI behavior...) of the current state and initialize them properly.
  - *Load the list of objects instances that will never be destroyed when going through the state.*
- Should contain:
  - Load object data method
  - Create/Add object instance method

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

State's Initialize

- During game state initialize:
  - Loading the list of objects instances used in this state *(The objects that need to be reloaded (Deleted then recreated) when resetting the state).*
  - *Re-initialize the object instances loaded in the game state load function and initializes the object instances that were created in state initialize function.*
- Should contain:
  - Create/Add object instance method
  - Initialize all objects method

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

State's Update

- During game state update:
  - Updating all objects instances found in the object list.
  - Creating new object instances and adding them to the object list depending on the game logic and input.
  - Removing "dead" instances from the list.

- Should contain:
  - Update method
  - Destroy dead objects method

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

State's Render / Draw

- During game state draw:
  - Send all the objects to the graphics manager in order to be drawn.

- Should contain:
  - Draw objects method

Note: We are lucky that we don't have to worry about graphics in flash

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

State's Uninitialize

- During game state uninitialize:
  - Remove all objects instances that were loaded in the game state initialize method and free their allocated memory.
- Should contain:
  - Remove Object Instance

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

State's Destroy

- During game state destroy:
  - Remove all remaining objects instances and free their allocated memory.
  - Unload all object material that was loaded in the game state load (constructor).
- Should contain:
  - Destroy all object data
  - Destroy (Destroy the game object manager, needed when exiting the game).

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

More…

- More methods can be added to the Object Manager. It all depends on your engine and other components.
  - GetObjectByName
  - GetObjectByIndex
  - RemoveObjectByName
  - RemoveObjectByIndex
  - etc…

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

Environment Manager

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

Static Objects

- As explained previously, the object manager takes care of object instances that are “alive” in the world, which in other words are the instances which the player can interact with.

- A game usually contains object instances of a different kind, which are static objects.

- These object instances can't be interacted with; their only purpose is to make the game scene look pretty and more appealing for the player. Examples:
  - Sun/Moon in an open area game.
  - Birds in a real time strategy game.
  - Grass in a racing or shooter game

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

Static Objects

- These type of object instances are handled by the environment manager.
- The main task of the environment manager is to determine which static instances are visible and send them to the graphics manager component in order to be drawn.
- Since these objects instances are static, they are usually preprocessed during the initialization of the game state in order to reduce the amount of time spent at run time.

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

Static Objects

- Note that when we said that the player can't interact directly with these static objects, that doesn't mean that they can't be affected by the player.
  - For example, if a game has a moon and a sun rotating in the sky, their positions are dictated by the game time.
  - While playing the game, the player could obtain special powers that allow him to speed up or speed down the game time, thus indirectly affecting the position of the moon and the sun.

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

Game Object

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

Game Object

- In order for the object manager to manage the objects, all the object instance entities need to provide some common functionalities:
  - **Initialize:** Used to reset the object instance to its initial state (Reset the animation counter to 0, reset the instance position...)
  - **Update:** Used to update the object instance according to its own behavior.
  - **Render:** Used to render the object instance ***(not in flash)***.
  - **CollisionReaction:** Called when the object collides with another object.
  - **Free / Destroy:** Used to free the object instance's resources prior to being deleted.

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

Our Engine

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

Object/Environment Managers in One

- We will not have two separate managers but will definitely separate the static objects from the dynamic objects.

- Multiple object lists will be found in our object manager and all functions will take that into consideration.

- The object manager will be responsible for:
  - Adding the objects into the correct list and of course on the stage.
  - Updating the objects
  - Sending dynamic objects to the collision manager
  - Calling the collided object's collision reaction methods
  - Remove dead objects
  - Re-initialize objects when restarting a state
  - Destroy the state by removing all the objects
  - etc…

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

Game Object

- As explained previously, in order for the object manager to manage the objects, all the object instance entities need to provide some common functionalities.

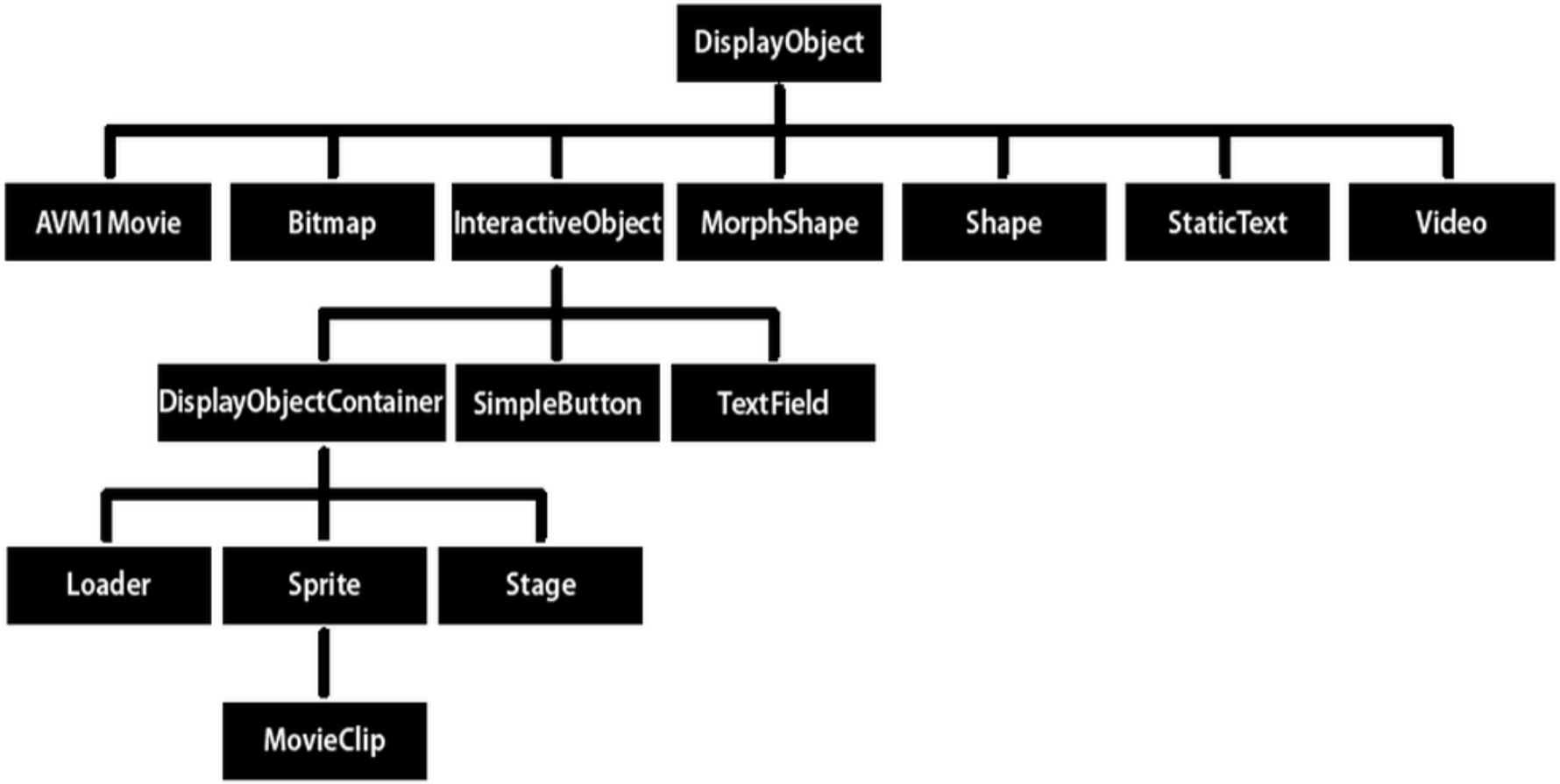
- Our game object will contain the following properties:
  - An integer representing the Type (object's type → Ship, Enemy, Bullet…)
  - A boolean that represents if the object is dead or not
  - A DisplayObject instance

- And will contain the following methods:
  - Initialize
  - Update
  - Destroy
  - CollisionReaction

Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

Displayobject & hierarchy

- Why does it contain a DisplayObject instance???



Copyright © 2011, DigiPen Institute of Technology. All Rights Reserved

The End ☺