

CS 4730: Computer Game Design

SFX and Level Team: Parallax Layers and Tweening

Overview

For this assignment, you add two features that are extremely common in games: parallaxing and tweening.

PART 1: Parallax Layers

Update your scene parsing code to support parallax layers. Parallax layers are used in 2D games to create the illusion of depth, by having different layers (e.g., background, middleground, foreground) scroll at different rates. Your updated code must include the following features:

- When describing a level in your scene text files, you can specify a type of object to be a “layer”. This object will be an empty display object container that holds (as children) the objects on that layer. Each layer has a scroll speed specified in the text file as well.
- When loading a scene, objects are put on layers by simply making them children of the layer object in the tree.
- When the camera moves, the layers scroll at different rates (as specified in the file).

Part 2: Tweening

In this section, you will be implementing a Tweening module for your game. The classes you will be building will be exactly as discussed in class. You will produce a very short demo showcasing your module’s ability to Tween objects in various ways.

Overview of Classes

You will be implementing the following classes:

- *TweenableParam*: A list of strings representing the parameters of a Sprite that can be Tweened.
- *TweenParam*: Object storing information relevant to one parameter being tweened.
- *Tween*: Object representing on Sprite being tweened in some way. Can have multiple *TweenParam* objects.
- *TweenJuggler*: Singleton class that handles all of the tweens in the game and updates them appropriately once per frame.
- *TweenTransition*: Class representing different possible transition functions for tweens. You MUST include at least one non-linear Tween transition.
- *TweenEvent*: Extends Event and represents progress or completion of a Tween.

**NOTE: You have the liberty to adjust these specific classes as you see fit, but we expect your tweening to generally follow this rough pattern / API.*

More Detailed API:

Here is the API that I used in my implementation of this lab. You do not have to implement your code exactly like mine, but hopefully this will be helpful to you nonetheless.

TweenableParams

- No Methods, just a list of static final strings (such as X, Y, SCALE_X, etc.). Makes code in other classes more readable. If using Java, an enum is probably more appropriate.

TweenParam

- TweenParam(TweenableParams paramToTween, double startVal, double endVal, double time);

- TweenableParams getParam();
- double getStartVal();
- double getEndVal();
- double getTweenTime();

Tween

- Tween(DisplayObject object);
- Tween(DisplayObject object, TweenTransitions transition);
- animate(TweenableParams fieldToAnimate, double startVal, double endVal, double time);
- update() //invoked once per frame by the TweenJuggler. Updates this tween / DisplayObject
- isComplete()
- setValue(TweenableParams param, double value);

TweenJuggler

- TweenJuggler()
- add(Tween tween);
- nextFrame(); //invoked every frame by Game, calls update() on every Tween and cleans up old / complete Tweens

TweenTransitions

- applyTransition(double percentDone);
- easeInOut(double percentDone); //applies a specific transition function, can have more of these for each transition your tweening supports. I will only list one here.

TweenEvent extends Event

- final static strings denoting the various events (e.g., TWEEN_COMPLETE_EVENT)
- TweenEvent(String eventType, Tween tween);
- Tween getTween();

Test Demo:

Produce a small demo to show off these new features. Build a simple level that involves a simple character platforming across a room to pick up a coin (or similar). As the character moves, the screen should scroll and parallax. In addition, you will add a couple tweens:

1. Have your character “tween into existence” at the beginning of the game. You might have the tween from scale 0 to 1 and alpha 0 to 1, or even have them move in from the bottom of the screen.
2. After collecting the coin. Have the coin tween in the following manner:
 - a. Moves to the center of the screen while growing to 2-4x its size.
 - b. Once positioned and scaled, have the coin fade out of existence (you can add more to this if you’d like, feel free to have fun with it.)

Turn In

As always, submit your code on Collab as a single zip file. Include a README.txt file so the grader knows how to play your demo.