Chapter 1

Math

中文测试

Cocos3D provides common mathmetica tools include vectors, matrices, quaternions and a series of untility mathmetica function.

1.1 Coordinate system

In *Cocos3D*, X-axis points to right, Y-axis points upward and Z-axis points out of screen. which is a kind of right-hand coordinate system.

1.2 Trigonometry

Some functions of this engine accept parameter(s) represent angles, another functions may return a angle result. By default, this engine use radian represenatation for these purposes. There are also a couple of functions, in math.util module, which convert degrees representaion betwween radian and degree.

Chapter 2

Logic system

Cocos3D use a current logic system called **Entity-Component-System**(abbr. ECS) for its logic system. ECS is flexiable and natural to real life.

2.1 Concepts

Things in *Cocos3D* are abstracted as **entity**. An entity may have one or more **component**s, each of these component represents one specific attribute of the entity. For example, *Cocos3D* itself provides some components include: ModelComponent, used to describe the apperance of the entity; Animation-Component, contains various animating related attributes such as animation wrap mode and whether it's in playing state or not, etc. Logics are mostly expressed in ECS **system**s. *Cocos3D* also has some builtin system. Users can define theirown systems and components. Systems and entities are organized and managed by **app**.

2.2 Entity

Entities can be created in app, with a name, and optional the level it locates. If the level is not specified, current actived level in app is used instead.

Entity can be actived or not deactived. Entity created in app is actived default.

2.3 Component

Each Component is created in one specific entity and then owned by that entity.

User-defined components should inherit from *Component* class.

There are some callbacks called when states of a component change, given that they are manually defined in component.

- *onInit()*
  Called when the component is being added to an entity or its owner entity is being re-actived and the component had never been initalized before. After this callback executes, the component is considered as initalized.

- *onDestroy()*
  Called when the component is destroyed.

- *onEnable()*
  Called when the component is set as enabled.

- *onDisable()*
  Called when the component is set as disabled.

- *onClone(source)*
  When a clone operation is performed on a component, the produced new component's *onClone* callback is called. The *source* argument is the source component to clone.

Chapter 3

Rendering

Chapter 4

Paricles System

Chapter 5

Animating

Chapter 6

Physical

Chapter 7

Advanced

7.1 Physical-based Rendering