#### CAView

###### 类说明

视图类CAView是整个CrossApp引擎最基本的类，负责将各式各样的界面呈现出来，我们在App中所能看见的的一切界面其实就是一个个CAView的组合。CAView负责在屏幕上定义矩形区域，在展示用户界面及响应用户界面交互方面发挥关键作用。每个视图对象要负责渲染试图矩形区域中的内容，并响应该区域内发生的操作事件，视图是应用程序用户交互的重要机制。

除了显示内容和处理事件之外，试图还可以管理一个或多个子视图。我们可以在一个view上面添加多个子view，而作为父view，即父节点，负责管理其直接子视图，并根据需要调整他们的位置和尺寸，以及响应他们没有处理的事件。

###### 基类

[CAResponder](../CAResponder.docx)

###### 属性

|  |  |  |
| --- | --- | --- |
| 访问修饰符 | 属性名 | 说明 |
| public | [Frame](#Frame) | 相对于父视图的位置大小 |
| public | [Bounds](#Bounds) | 相对于自身的位置大小 |
| public | [Center](#Center) | 中点的frame |
| public | [Color](#Color) | 颜色 |
| public | [Alpha](#Alpha) | 透明度 |
| public | [ZOrder](#ZOrder) | z轴值 |
| public | [Scale](#Scale) | 缩放比例 |
| public | [Skew](#Skew) | 扭曲度 |
| public | [Visible](#Visible) | 是否可见 |
| public | [Rotation](#Rotation) | 旋转 |
| public | [Tag](#Tag) | 标签 |

###### 方法

|  |  |  |
| --- | --- | --- |
| 访问修饰符 | 方法名 | 说明 |
| public | [create](#create) | 创建一个空view，默认Frame为(0,0,0,0) |
| public | [createWithFrame](#createWithFrame) | 创建一个view，并指定其frame |
| public | [createWithFrame](#createWithFrame1) | 创建一个view，并指定frame和Color |
| public | [createWithCenter](#createWithCenter) | 创建一个view，并设置其center |
| public | [createWithCenter](#createWithCenter1) | 创建一个view，并设置其center和color |
| public | [createWithColor](#createWithColor) | 创建一个view，并指定Color，默认Frame为(0,0,0,0) |
| public | [addSubview](#addSubview) | 将子视图添加进当前视图 |
| public | [insertSubView](#insertSubview) | 将子视图添加进当前视图，并指定一个Z轴值。 |
| public | [getSubviewByTag](#getSubviewByTag) | 通过标签值获取子视图 |
| public | [getSubviews](#getSubviews) | 获取子视图的集合 |
| public | [getSubviewsCount](#getSubviewsCount) | 获取子视图的数量 |
| public | [removeFromSuperview](#removeFromSuperview) | 从父视图移除 |
| public | [removeSubview](#removeSubview) | 移除子视图 |
| public | [removeSubviewByTag](#removeSubviewByTag) | 根据标签值移除子视图 |
| public | [removeAllSubviews](#removeAllSubviews) | 移除当前视图的所有子视图 |
| public | [reorderSubview](#reorderSubview) | 记录子视图与对应的标签值 |
| public | [sortAllSubviews](#sortAllSubviews) | 对当前视图的所有子视图进行排序 |

###### 属性介绍

**Frame**

类型：CCRect

解释：确定view在屏幕上显示的位置和大小，参考的是父视图的坐标系统。CCRect包括两个成员，一个是起点坐标origin，另一个是宽高size，。对于CAView及其子类的frame操作，如果view有原始大小不为0，在 createWithFrame、setFrame等操作时，可以设置size为(0,0)，则不改变view的大小；如果只想改变view的坐标，不需要改变view的大小，则可采用setFrameOrigin方法来进行设置，对于center属性也提供了类似的操作，get/set{}。

**Bounds**

类型：CCRect

解释：view在自身坐标系统中的位置和大小，参考的是自身的坐标系统，CCRect的origin值永远都是(0,0)。在设置frame时，bounds也确定，其值等于frame的值。bounds和frame是有区别的，frame的值会随着view的缩放操作改变，bounds的值则不会改变，get/set{}。

**Center**

类型：CCRect

解释：view的中心点在屏幕上的位置，CrossApp采用的坐标系是屏幕坐标系，即左上角为原点，向右和向下分别对应X和Y轴。在CrossApp中我们确定一个view的位置，是通过origin和size来设定的，为了方便，这里可以直接使用Center将view的中心点设置在我们想要的位置。如果只想改变view的坐标，不需要改变view的大小，则可采用setFrameOrigin方法来进行设置，get/set{}。

**Color**

类型：CAColor4B

解释：view的颜色，默认为白色的view，get/set{}。

示例：

|  |
| --- |
| CCRect winRect = this->getView()->getBounds();  CAView\* view = CAView::createWithFrame(CCRect(300,300,200,200),ccc4(255,0,0,255));  view->setCenter(CCRect(winRect.size.width\*0.5,winRect.size.height\*0.5,0,0));  view->setColor(ccc4(0,255,0,255));  CALabel\* tf = CALabel::createWithFrame(CCRect(20,20,0,0));  tf->setText(“view”);  tf->setFontSize(25);  view->addSubview(tf);  this->getView()->addSubview(view); |

创建一个view，指定view的frame，将color设置为绿色。

**Alpha**

类型：float

解释：view的透明度，其值为0-1之间，设置alpha时将会改变view及其所有子类的透明，如果只需要设置当前view的透明度，可以在color属性中的透明度参数中传递一个0-255之间的值，get/set{}。

**ZOrder**

类型：int

解释：view的Z轴属性，多个view之间可以进行叠加，通过设置ZOrder，可以按顺序将view进行叠加，确定view的上下覆盖关系，get/set{}。

**Scale**

类型：float

解释：view可以进行缩放操作，其中缩放包括XY轴同时缩放Scale、X轴缩放ScaleX和Y轴缩放ScaleY，get/set{}。

示例：

|  |
| --- |
| CAView\* view1 = CAView::createWithColor(ccc4(255,0,0,255));  view1->setCenter(CCRect(600, 400,200, 200));  CALabel\* tf1 = CALabel::createWithFrame(CCRect(100,100,0,0));  tf1->setText(“view1”);  tf1->setFontSize(25);  view->addSubview(tf1);  this->getView()->insertSubview(view,1);    CAView\* view2 = CAView::createWithColor(ccc4(0, 0, 255, 255));  view2->setCenter(CCRect(500, 300, 200, 200));  view2->setScale(0.5);  view2->setZOrder(2);  CALabel \* tf2 = CALabel::createWithFrame(CCRect(50, 50, 0, 0));  tf2->setText(“view1”);  tf2->setFontSize(25);  view2->addSubview(tf2);  this->getView()->addSubview(view2);  CAView\* view1 = CAView::createWithColor(ccc4(255,0,0,255));  view1->setCenter(CCRect(600, 400,200, 200));  CALabel\* tf1 = CALabel::createWithFrame(CCRect(100,100,0,0));  tf1->setText(“view1”);  tf1->setFontSize(25);  view->addSubview(tf1);  this->getView()->insertSubview(view,1);    CAView\* view2 = CAView::createWithColor(ccc4(0, 0, 255, 255));  view2->setCenter(CCRect(500, 300, 200, 200));  view2->setScale(0.5);  view2->setZOrder(2);  CALabel \* tf2 = CALabel::createWithFrame(CCRect(50, 50, 0, 0));  tf2->setText(“view1”);  tf2->setFontSize(25);  view2->addSubview(tf2);  this->getView()->addSubview(view2); |

通过改变Z轴的值，设置view的层级关系，同时对view进行缩放。

**Skew**

类型：float

解释：view的扭曲值，包括XY轴同时扭曲和X、Y轴单独设置，is/set{}。

**Visible**

类型：bool

解释：view是否可见，is/set{}。

**Rotation**

类型：float

解释：view的旋转，包括XY轴同时旋转、X轴旋转和Y轴旋转，get/set{}。

**Tag**

类型：int

解释：view的标签值，get/set{}。

###### 方法介绍

**static CAView \*** **create(void)**

返回值：CAView\*

解释：创建一个view对象，默认大小为0，颜色为白色。

**static CAView\*** **createWithFrame(const CCRect& rect)**

返回值：CAView\*

参数：

|  |  |  |
| --- | --- | --- |
| 类型 | 参数名 | 说明 |
| CCRect | rect | view的位置大小 |

**static CAView\*** **createWithFrame(const CCRect& rect, const ccColor4B& color4B)**

返回值：CAView\*

参数：

|  |  |  |
| --- | --- | --- |
| 类型 | 参数名 | 说明 |
| CCRect | rect | view的位置大小 |
| CAColor4B | color4B | view的颜色 |

**static CAView\*** **createWithCenter(const CCRect& rect)**

返回值：CAView\*

参数：

|  |  |  |
| --- | --- | --- |
| 类型 | 参数名 | 说明 |
| CCRect | rect | view中点的位置及view大小 |

**static CAView\*** **createWithCenter(const CCRect& rect, const ccColor4B& color4B)**

返回值：CAView\*

参数：

|  |  |  |
| --- | --- | --- |
| 类型 | 参数名 | 说明 |
| CCRect | rect | view中点的位置及view大小 |
| CAColor4B | color4B | view的颜色 |

**static CAView\*** **createWithColor(const ccColor4B& color4B)**

返回值：CAView\*

参数：

|  |  |  |
| --- | --- | --- |
| 类型 | 参数名 | 说明 |
| CAColor4B | color4B | view的颜色 |

**virtual void** **addSubview(CAView \* child)**

返回值：void

参数：

|  |  |  |
| --- | --- | --- |
| 类型 | 参数名 | 说明 |
| CAView\* | child | 子视图 |

**virtual void** **insertSubview(CAView\* subview, int z)**

返回值：void

参数：

|  |  |  |
| --- | --- | --- |
| 类型 | 参数名 | 说明 |
| CAView\* | child | 子视图 |

解释：添加子视图，并设置其Z轴值。

**CAView \*** **getSubviewByTag(int tag)**

返回值：CAView\*

参数：

|  |  |  |
| --- | --- | --- |
| 类型 | 参数名 | 说明 |
| int | tag | 标签值 |

解释：根据子视图的标签值来获取子视图

**virtual CCArray\*** **getSubviews()**

返回值：CCArray\*

解释：获取当前视图的所有子视图，返回子视图数组。

**unsigned int** **getSubviewsCount(void) const**

返回值：unsigned int

解释：返回一个unsigned int类型的值，用于标识当前子视图的个数。

**virtual void** **removeFromSuperview()**

返回值：void

解释：从父视图移除当前视图

**virtual void** **removeSubview(CAView\* subview)**

返回值：void

解释：根据给定的子视图对象，从当前视图移除。

**virtual void** **removeSubviewByTag(int tag)**

返回值：void

参数：

|  |  |  |
| --- | --- | --- |
| 类型 | 参数名 | 说明 |
| int | tag | 标签值 |

解释：根据标签值移除子视图

**virtual void** **removeAllSubviews()**

返回值：void

解释：移除当前视图上的所有子视图

**virtual void** **reorderSubview(CAView \* child, int zOrder)**

返回值：void

参数：

|  |  |  |
| --- | --- | --- |
| 类型 | 参数名 | 说明 |
| CAView\* | child | 子视图对象 |
| int | zOrder | Z轴值 |

解释：记录子视图

**virtual void** **sortAllSubviews()**

返回值：void

解释：根据子视图的zOrder值对所有子视图进行排序