这篇文档介绍了Bolt引擎的扩展元对象高斯模糊对象(GaussianBlurObject) 的属性, 使用方法和性能数据.

[对象的属性, 使用方法和性能数据](#_对象的属性,_使用方法和性能数据)

[属性](#_属性)

[使用方法](#_使用方法)

[性能](#_性能)

## 高斯模糊对象的属性, 使用方法和性能数据

扩展对象名称: GaussianBlurObject

继承自: LayoutObject

该扩展对象对其所处矩形的图像使用高斯模糊, 通过控制模糊系数来控制模糊程度



###### 属性

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 说明 | 读写方法 |
| gaussiantype | string | Default  可选值: Default, IIR, FIR | 对象使用的模糊算法.  IIR的性能不随模糊系数和模糊半径变化;  FIR的性能随着模糊系数和半径递增;  Default根据模糊系数择优选择FIR或IIR; | SetGaussianType(string)  string GetGaussianType() |
| sigma | float | 1 | 模糊系数. 系数越大越模糊 | SetSigma(float)  float GetSigma() |
| radius | int | 3 | 模糊半径.  在FIR模式下, 建议取模糊系数的3倍取整, 性能随半径递增;  在IIR模式下, 这个属性没有意义. | SetRadius(int)  int GetRadius() |

###### 使用方法

扩展元对象像内置元对象一样使用. 可以指定的属性包括LayoutObject的所有属性和上文介绍的3个属性. 高斯模糊对象没有自定义事件, 如常使用LayoutObject的所有事件.

属性gaussiantype指定了高斯模糊对象使用的模糊算法. 对使用者来说, 需要注意以下三点:

* 在使用FIR算法时, 需要同时指定sigma和radius. radius=3\*sigma时可以达到预期的模糊效果, 与radius > 3\*sigma的效果差异微弱. 使用者当然有让radius < 3\*sigma的自由, ]可能面临模糊系数设置的很大, 但是模糊效果甚微的窘境.

由于FIR算法的计算量随模糊半径线性增长, **使用radius = 3\*sigma当然是性价比最高的策略**. 从下面的对比图左上和右上两组文字对比中可以看出, 模糊半径设大了没有用, 徒增CPU消耗; 从左下和右下两组文字对比中看出, 模糊半径设小了模糊效果不好.



图1 – 模糊半径, 模糊系数, 对模糊效果的影响

* 在使用IIR算法时, 算法和模糊效果都与模糊半径无关, 使用者只需按需设定模糊系数sigma即可.
* FIR算法在模糊系数小的时候效率高, IIR算法在模糊系数大的时候效率比FIR高. 所以高斯模糊对象提供”Default”的模糊类型混用两种算法. 在模糊系数变化的时候, 自动将模糊半径设为模糊系数的3倍; 在模糊半径小于等于1的时候, 使用模糊半径等于3倍模糊系数的FIR算法; 在模糊半径大于1的时候使用与半径无关的IIR算法以达到最优性能.
* 剪裁敏感. 高斯模糊对象是剪裁敏感的, 所以需要放在剪裁敏感的层对象LayerObject中使用. 这意味着同一层中的任何对象发生更新, 高斯模糊对象都会跟着更新.

###### 性能

Bolt对图像实时性能要求是800x600的图片20ms内处理完. 下面是目前提供的两种算法的性能数据:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 模糊系数= 0.33  模糊半径= 1 | 模糊系数=1  模糊半径=3 | 模糊系数=8  模糊半径=24 |
| FIR算法, MMX 实现 | <15 ms | < 15 ms | <50 ms |
| IIR 算法, SSE实现 | < 20 ms | | |

可以看到使用IIR算法性能较稳定, 使用Default模式结合两种算法可以达到性能要求.