# 1.概念

|  |
| --- |
|  |

# 一.图像粒子,ImageParticle

## 2.简单使用

### 使用步骤

|  |
| --- |
| 1.绑定所有元素到一个模拟的粒子系统（ ParticleSystem）  2. 一个向系统发射粒子的发射器（ Emitter） 。  3. 一个ParticlePainter派生元素， 用来实现粒子的可视化。 |

### 项目结构

|  |
| --- |
|  |

**需要导入库: import QtQuick.Particles 2.0**

|  |  |
| --- | --- |
| **//main.qml**  import QtQuick 2.15  import QtQuick.Window 2.15  import QtQuick.Particles 2.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Particle Demo")  //粒子系统的简单使用  Rectangle{  id:*root*  x:10  y:10  width: 480  height: 160  color: "#1f1f1f"  //实例化一个粒子系统  ParticleSystem{  id:*particleSystem*  }  //实例化一个粒子发射器  Emitter{  id:*emitter*  anchors.centerIn: *parent*  width: 160;height: 80  system: *particleSystem*  //每一次发射粒子数  emitRate: 30  //粒子的生命周期  lifeSpan: 1000  //粒子变化周期  lifeSpanVariation: 500  //粒子大小  size:16  //粒子最终大小  endSize: 32  Tracer{  color:'green'  }  }  //用来做粒子的图片,需要指明source和system  ImageParticle{  source:"images/particle.png"  system: *particleSystem*  }  }  } |  |

### 简单改进,使他充满这个矩形区域

|  |  |
| --- | --- |
| //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtQuick.Particles 2.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Particle Demo")  //粒子系统的简单使用  Rectangle{  id:*root*  x:10  y:10  width: 480  height: 160  color: "#1f1f1f"  //实例化一个粒子系统  ParticleSystem{  id:*particleSystem*  }  //实例化一个粒子发射器  Emitter{  id:*emitter*  anchors.centerIn: *parent*  anchors.fill: *parent*  system: *particleSystem*  //每一次发射粒子数  emitRate: 30  //粒子的生命周期  lifeSpan: 1000  //粒子变化周期  lifeSpanVariation: 500  //粒子大小  size:16  //粒子最终大小  endSize: 32  Tracer{  color:'yellow'  }  }  //用来做粒子的图片,需要指明source和system  ImageParticle{  source:"images/particle.png"  system: *particleSystem*  }  }  } |  |

## 3.星星闪烁的效果

|  |  |
| --- | --- |
| //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtQuick.Particles 2.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Particle Demo")  //星星闪烁的效果  Rectangle{  id:*root*  x:10  y:10  width: 480  height: 160  color: "#1f1f1f"  //实例化一个粒子系统  ParticleSystem{  id:*particleSystem*  }  //实例化一个粒子发射器  Emitter{  id:*emitter*  anchors.fill: *parent*  system: *particleSystem*  //每一次发射粒子数  // emitRate: 30  //粒子的生命周期  lifeSpan: 1000  //粒子大小  size:16  //粒子最终大小  endSize: 32  }  //用来做粒子的图片,需要指明source和system  ImageParticle{  source:"images/star.png"  system: *particleSystem*  //设置颜色  color: '#FFD700'  //设置颜色变化  colorVariation: 0.2  //旋转  rotation: 0  //旋转角度变化  rotationVariation: 45  //旋转速度  rotationVelocity: 15  //旋转速度变化  rotationVelocityVariation: 15  //进入效果  entryEffect: ImageParticle.Scale  }  }  } |  |

## 4.旋转的吊扇

|  |  |
| --- | --- |
| //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtQuick.Particles 2.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Particle Demo")  //旋转的吊扇  Rectangle {  id: *root*  width: 400; height: 400  color: "#333333"  ParticleSystem {  id: *particleSystem*  }  ImageParticle {  source: "images/star.png"  system: *particleSystem*  colorVariation: 1.0  rotation: 45  rotationVariation: 30  rotationVelocity: 0  rotationVelocityVariation: 180  entryEffect: ImageParticle.Scale  alpha: 0.5  alphaVariation: 0.4  }  Emitter {  id: *emitter*  anchors.fill: *parent*  system: *particleSystem*  lifeSpan: 4000  size: 64  endSize: 256  sizeVariation: 32  }  }  } |  |

## 5. 方向粒子

|  |
| --- |
| 我们已经看到了粒子的旋转， 但是我们的粒子需要一个轨迹。 轨迹由速度 随机方向的加速度指定， 也可以叫做矢量空间。 有多种可用矢量空间用来定义粒子的速度或加速度： 角度方向（ AngleDirection） - 使用角度的方向变化。 点方向（ PointDirection） - 使用x,y组件组成的方向变化。 目标方向（ TargetDirection） - 朝着目标点的方向变化。 |

### 实例朝着指定方向移动的粒子

|  |  |
| --- | --- |
| //main.qml  import QtQuick.Window 2.15  import QtQuick.Particles 2.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Particle Demo")  //粒子方向  Rectangle{  id: *root*  width: 400;height: 400  color: "#333333"  ParticleSystem{  id: *particleSystem*  }  ImageParticle{  source:"images/star.png"  system: *particleSystem*  color: '#FFD700'  colorVariation: 0.2  rotation: 0  rotationVariation :45  rotationVelocity: 15  rotationVelocityVariation:15  entryEffect: ImageParticle.Scale  }  Emitter{  system: *particleSystem*  anchors.left: *parent*.left  anchors.horizontalCenter: *parent*.horizontalCenter  width: 1;height: 1  lifeSpan: 6400  lifeSpanVariation: 400  size:32  velocity: AngleDirection{  angle: 0  angleVariation: 15  magnitude: 100  magnitudeVariation: 50  }  }  }  } |  |

## 6.抛物线粒子效果

|  |  |
| --- | --- |
| //main.qml  import QtQuick.Window 2.15  import QtQuick.Particles 2.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Particle Demo")  //抛物线粒子效果  Rectangle{  id:*root*  width: 400; height: 400  color: "#333333"  ParticleSystem{  id:*particleSystem*  }  ImageParticle{  source:"images/star.png"  system: *particleSystem*  color: '#FFD700'  colorVariation: 0.2  rotation: 0  rotationVariation: 45  rotationVelocity: 15  rotationVelocityVariation: 15  entryEffect: ImageParticle.Scale  }  Emitter{  system: *particleSystem*  anchors.left: *parent*.left  anchors.verticalCenter: *parent*.verticalCenter  height: 1 ; width:1 //修改这里可以获得不同宽度的抛物线  lifeSpan: 6400  lifeSpanVariation: 400  size:32  velocity: AngleDirection{  angle: -45  angleVariation: 0  magnitude: 100  }  //加速度  acceleration: AngleDirection{  angle: 90  magnitude: 25  }  }  }  } |  |

### 小改进,宽抛物线

|  |  |
| --- | --- |
| //main.qml  import QtQuick.Window 2.15  import QtQuick.Particles 2.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Particle Demo")  //抛物线粒子效果  Rectangle{  id:*root*  width: 400; height: 400  color: "#333333"  ParticleSystem{  id:*particleSystem*  }  ImageParticle{  source:"images/star.png"  system: *particleSystem*  color: '#FFD700'  colorVariation: 0.2  rotation: 0  rotationVariation: 45  rotationVelocity: 15  rotationVelocityVariation: 15  entryEffect: ImageParticle.Scale  }  Emitter{  system: *particleSystem*  anchors.left: *parent*.left  anchors.verticalCenter: *parent*.verticalCenter  height: 50 ; width:50  lifeSpan: 6400  lifeSpanVariation: 400  size:32  velocity: AngleDirection{  angle: -45  angleVariation: 0  magnitude: 100  }  //加速度  acceleration: AngleDirection{  angle: 90  magnitude: 25  }  }  }  } |  |

## 7.点方向粒子系统

|  |  |
| --- | --- |
| //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtQuick.Particles 2.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Particle Demo")    //点方向PointDirection  Rectangle {  id: *root*  width: 480; height: 240  color: "#1F1F1F"  ParticleSystem {  id: *particleSystem*  }  ImageParticle {  source: "images/star.png"  system: *particleSystem*  color: '#FFD700'  colorVariation: 0.2  rotation: 0  rotationVariation: 45  rotationVelocity: 15  rotationVelocityVariation: 15  entryEffect: ImageParticle.Scale  }  Emitter {  id: *emitter*  anchors.left: *parent*.left  anchors.verticalCenter: *parent*.verticalCenter  width: 1; height: 1  system: *particleSystem*  emitRate: 10  lifeSpan: 6400  lifeSpanVariation: 400  size: 32  velocity: PointDirection {  x: 100  y: 0  xVariation: 0  yVariation: 100/6  }  }  }  } |  |

### 小改进,添加一个简单方向的加速度

|  |  |
| --- | --- |
| import QtQuick 2.15  import QtQuick.Window 2.15  import QtQuick.Particles 2.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Particle Demo")  //点方向PointDirection  Rectangle {  id: *root*  width: 480; height: 240  color: "#1F1F1F"  ParticleSystem {  id: *particleSystem*  }  ImageParticle {  source: "images/star.png"  system: *particleSystem*  color: '#FFD700'  colorVariation: 0.2  rotation: 0  rotationVariation: 45  rotationVelocity: 15  rotationVelocityVariation: 15  entryEffect: ImageParticle.Scale  }  Emitter {  id: *emitter*  anchors.left: *parent*.left  anchors.verticalCenter: *parent*.verticalCenter  width: 1; height: 1  system: *particleSystem*  emitRate: 10  lifeSpan: 6400  lifeSpanVariation: 400  size: 32  velocity: PointDirection {  x: 100  y: 0  xVariation: 0  yVariation: 100/6  }  acceleration: AngleDirection {  angle: 90  magnitude: 25  }  }  }  } |  |

## 8.目标方向,”毛毛虫效果”

|  |  |
| --- | --- |
| //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtQuick.Particles 2.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Particle Demo")  //目标方向  Rectangle {  id: *root*  width: 480; height: 240  color: "#1F1F1F"  ParticleSystem {  id: *particleSystem*  }  ImageParticle {  source: "images/star.png"  system: *particleSystem*  color: '#FFD700'  colorVariation: 0.2  rotation: 0  rotationVariation: 45  rotationVelocity: 15  rotationVelocityVariation: 15  entryEffect: ImageParticle.Scale  }  Emitter {  id: *emitter*  anchors.left: *parent*.left  anchors.verticalCenter: *parent*.verticalCenter  width: 1; height: 1  system: *particleSystem*  emitRate: 10  lifeSpan: 6400  lifeSpanVariation: 400  size: 32  velocity: TargetDirection {  targetX: 100  targetY: 0  targetVariation: 0  magnitude: 100/6  }  }  }  } |  |

## 9.目标方向,改进版

|  |  |
| --- | --- |
| //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtQuick.Particles 2.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Particle Demo")  //目标方向  Rectangle {  id: *root*  width: 480; height: 240  color: "#1F1F1F"  ParticleSystem {  id: *particleSystem*  }  ImageParticle {  source: "images/star.png"  system: *particleSystem*  color: '#FFD700'  colorVariation: 0.2  rotation: 0  rotationVariation: 45  rotationVelocity: 15  rotationVelocityVariation: 15  entryEffect: ImageParticle.Scale  }  Emitter {  id: *emitter*  anchors.left: *parent*.left  anchors.verticalCenter: *parent*.verticalCenter  width: 500; height: 1  system: *particleSystem*  emitRate: 10  lifeSpan: 6400  lifeSpanVariation: 400  size: 32  velocity: TargetDirection {  targetX: 20  targetY: 30  targetVariation: 20  magnitude: 100/6  }  }  }  } |  |

## 10.瀑布效果,添加了加速度的效果

|  |  |
| --- | --- |
| //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtQuick.Particles 2.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Particle Demo")  //目标方向  Rectangle {  id: *root*  width: 480; height: 240  color: "#1F1F1F"  ParticleSystem {  id: *particleSystem*  }  ImageParticle {  source: "images/star.png"  system: *particleSystem*  color: '#FFD700'  colorVariation: 0.2  rotation: 0  rotationVariation: 45  rotationVelocity: 15  rotationVelocityVariation: 15  entryEffect: ImageParticle.Scale  }  Emitter {  id: *emitter*  anchors.left: *parent*.left  anchors.verticalCenter: *parent*.verticalCenter  width: 500; height: 1  system: *particleSystem*  emitRate: 10  lifeSpan: 6400  lifeSpanVariation: 400  size: 32  velocity: TargetDirection {  targetX: 100  targetY: 30  targetVariation: 20  magnitude: 100/6  }  acceleration: AngleDirection {  angle: 90  magnitude: 25  }  }  }  } |  |

# 二.粒子画笔

|  |
| --- |
| 到目前为止我们只使用了基于粒子画笔的图像来实现粒子可视化。 Qt也提供了一些 其它的粒子画笔： 粒子项（ ItemParticle） ： 基于粒子画笔的代理 自定义粒子（ CustomParticle） ： 基于粒子画笔的着色器 粒子项可以将QML元素项作为粒子发射。 你需要制定自己的粒子代理 |

### 11.ItemParticle

以下的实例的粒子系统可以从一个图片数组中随机挑选几个图片作为粒子来发射

|  |  |
| --- | --- |
| //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtQuick.Particles 2.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Particle Demo")  //粒子项系统  Rectangle{  id:*root*  width: 400; height: 400  color: '#1F1F1F'  //定义图像数组  property var images: [  "images/box\_blue.png",  "images/box\_red.png",  "images/box\_green.png",  "images/circle\_blue.png",  "images/circle\_red.png",  "images/circle\_green.png",  "images/triangle\_blue.png",  "images/triangle\_red.png",  "images/triangle\_green.png" ]  ParticleSystem{  id:*particleSystem*  }  Emitter{  id:*emitter*  system: *particleSystem*  anchors.margins: 32  anchors.fill: *parent*  emitRate: 4  lifeSpan: 2000  }  ItemParticle{  system: *particleSystem*  delegate: *itemParticleDelegate*  }  Component{  id: *itemParticleDelegate*  Item {  id: *container*  width: 32\**Math*.ceil(*Math*.random()\*3); height: *width*  Image {  anchors.fill: *parent*  anchors.margins: 4  source: *root*.images[*Math*.floor(*Math*.random()\*9)]  }  }  }  }  } |  |

# 三粒子控制

|  |
| --- |
| 粒子由粒子发射器发出。 在粒子发射出后， 发射器无法再改变粒子。 粒子控制器允 许你控制发射后的粒子参数。 控制器的每个类型使用不同的方法来影响粒子： 生命周期（ Age） - 修改粒子的生命周期 吸引（ Attractor） - 吸引粒子朝向指定点 摩擦（ Friction） - 按当前粒子速度成正比减慢运动 重力（ Gravity） - 设置一个角度的加速度 紊流（ Turbulence） - 强制基于噪声图像方式的流动 漂移（ Wander） - 随机变化的轨迹 组目标（ GroupGoal） - 改变一组粒子群的状态 子粒子（ SpriteGoal） - 改变一个子粒子的状态 |

### 12.Age

允许粒子老得更快， lifeLeft属性指定了粒子的有多少的生命周期。

|  |  |
| --- | --- |
| //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtQuick.Particles 2.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Particle Demo")  //粒子控制1.Age  Rectangle{  id:*root*  x:10  y:10  width: 480  height: 240  anchors.horizontalCenter: *parent*.horizontalCenter  color: "#1f1f1f"  //实例化一个粒子系统  ParticleSystem{  id:*particleSystem*  }  //实例化一个粒子发射器  Emitter{  id:*emitter*  anchors.centerIn: *parent*  //width: 160;height: 80  anchors.fill: *parent*  system: *particleSystem*  //每一次发射粒子数  emitRate: 30  //粒子的生命周期  lifeSpan: 6400  //粒子变化周期  lifeSpanVariation: 400  //粒子大小  size:32  //粒子最终大小  // endSize: 32  }  //用来做粒子的图片,需要指明source和system  ImageParticle{  source:"images/particle.png"  system: *particleSystem*  color: '#FFD700'  colorVariation: 0.2  rotation: 0  rotationVariation: 45  // rotationVelocity: 15  rotationVelocityVariation: 15  entryEffect: ImageParticle.Scale  }  //粒子控制:Age  Age{  y:10  anchors.horizontalCenter: *parent*.horizontalCenter  width: 240; height: 120  lifeLeft: 1200  system: *particleSystem*  advancePosition: true  once: true  Tracer{}  }  }    } |  |

### 13. 吸引（ Attractor）

|  |
| --- |
| 吸引会将粒子朝指定的点上吸引。 这个点使用pointX与pointY来指定， 它是与吸引 区域的几何形状相对的。 strength指定了吸引的力度。 在我们的例子中， 我们让粒 子从左向右运动， 吸引放在顶部， 有一半运动的粒子会穿过吸引区域。 控制器只会 影响在它们几何形状内的粒子。 这种分离让我们可以同步看到正常的流动与受影响 的流动。 |

|  |  |
| --- | --- |
| //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtQuick.Particles 2.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Particle Demo")  //粒子控制1.Age  Rectangle{  id:*root*  x:10  y:10  width: 480  height: 240  anchors.horizontalCenter: *parent*.horizontalCenter  color: "#1f1f1f"  //实例化一个粒子系统  ParticleSystem{  id:*particleSystem*  }  //实例化一个粒子发射器  Emitter{  id:*emitter*  anchors.centerIn: *parent*  //width: 160;height: 80  anchors.fill: *parent*  system: *particleSystem*  //每一次发射粒子数  emitRate: 30  //粒子的生命周期  lifeSpan: 6400  //粒子变化周期  lifeSpanVariation: 400  //粒子大小  size:32  //粒子最终大小  // endSize: 32  }  //用来做粒子的图片,需要指明source和system  ImageParticle{  source:"images/particle.png"  system: *particleSystem*  color: '#FFD700'  colorVariation: 0.2  rotation: 0  rotationVariation: 45  // rotationVelocity: 15  rotationVelocityVariation: 15  entryEffect: ImageParticle.Scale  }  //粒子控制Attractor  Attractor {  anchors.horizontalCenter: *parent*.horizontalCenter  width: 160; height: 120  system: *particleSystem*  pointX: 0  pointY: 0  strength: 1.0  Tracer {}  }  }    } |  |

### 14.摩擦

摩擦控制器使用一个参数（ factor） 减慢粒子运动， 直到达到一个阈值。

|  |  |
| --- | --- |
| import QtQuick 2.15  import QtQuick.Window 2.15  import QtQuick.Particles 2.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Particle Demo")  //粒子控制1.Age  Rectangle{  id:*root*  x:10  y:10  width: 480  height: 240  anchors.horizontalCenter: *parent*.horizontalCenter  color: "#1f1f1f"  //实例化一个粒子系统  ParticleSystem{  id:*particleSystem*  }  //实例化一个粒子发射器  Emitter{  id:*emitter*  anchors.centerIn: *parent*  //width: 160;height: 80  anchors.fill: *parent*  system: *particleSystem*  //每一次发射粒子数  emitRate: 30  //粒子的生命周期  lifeSpan: 6400  //粒子变化周期  lifeSpanVariation: 400  //粒子大小  size:32  velocity: AngleDirection {  angle: 0  angleVariation: 15  magnitude: 100  magnitudeVariation: 100  }  }  //用来做粒子的图片,需要指明source和system  ImageParticle{  source:"images/particle.png"  system: *particleSystem*  color: '#FFD700'  colorVariation: 0.2  rotation: 0  rotationVariation: 45  // rotationVelocity: 15  rotationVelocityVariation: 15  entryEffect: ImageParticle.Scale  }  //粒子控制摩擦（ Friction）  Friction{  anchors.horizontalCenter: *parent*.horizontalCenter  width: 240; height: 150  system: *particleSystem*  //摩擦系数  factor: 0.8  //阈值  threshold: 25  Tracer{  }  }  }  } |  |

### 15.重力gravity

|  |
| --- |
| 重力控制器应用在加速度上， 在我们的例子中， 我们使用一个角度方向将粒子从底 部发射到顶部。 右边是为控制区域， 左边使用重力控制器控制， 重力方向为90度方 向（ 垂直向下） ， 梯度值为50。 |

|  |  |
| --- | --- |
| //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtQuick.Particles 2.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Particle Demo")  //粒子控制1.Age  Rectangle{  id:*root*  x:10  y:10  width: 480  height: 240  anchors.horizontalCenter: *parent*.horizontalCenter  color: "#1f1f1f"  //实例化一个粒子系统  ParticleSystem{  id:*particleSystem*  }  //实例化一个粒子发射器  Emitter{  id:*emitter*  anchors.centerIn: *parent*  //width: 160;height: 80  anchors.fill: *parent*  system: *particleSystem*  //每一次发射粒子数  emitRate: 30  //粒子的生命周期  lifeSpan: 6400  //粒子变化周期  lifeSpanVariation: 400  //粒子大小  size:32  //粒子最终大小  // endSize: 32  velocity: AngleDirection {  angle: 0  angleVariation: 15  magnitude: 100  magnitudeVariation: 100  }  }  //用来做粒子的图片,需要指明source和system  ImageParticle{  source:"images/particle.png"  system: *particleSystem*  color: '#FFD700'  colorVariation: 0.2  rotation: 0  rotationVariation: 45  // rotationVelocity: 15  rotationVelocityVariation: 15  entryEffect: ImageParticle.Scale  }    //粒子控制Gravity  Gravity{  anchors.horizontalCenter: *parent*.horizontalCenter  width: 240; height: 240  system: *particleSystem*  angle: 90  magnitude: 50  Tracer{}  }  }  } |  |

### 16. 紊流（ Turbulence）

|  |
| --- |
| 紊流控制器， 对粒子应用了一个混乱映射方向力的矢量。 这个混乱映射是由一个噪 声图像定义的。 可以使用noiseSource属性来定义噪声图像。 strength定义了矢量对 于粒子运动的影响有多大。 |

|  |  |
| --- | --- |
| //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtQuick.Particles 2.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Particle Demo")  //粒子控制1.Age  Rectangle{  id:*root*  x:10  y:10  width: 480  height: 240  anchors.horizontalCenter: *parent*.horizontalCenter  color: "#1f1f1f"  //实例化一个粒子系统  ParticleSystem{  id:*particleSystem*  }  //实例化一个粒子发射器  Emitter{  id:*emitter*  anchors.centerIn: *parent*  //width: 160;height: 80  anchors.fill: *parent*  system: *particleSystem*  //每一次发射粒子数  emitRate: 30  //粒子的生命周期  lifeSpan: 6400  //粒子变化周期  lifeSpanVariation: 400  //粒子大小  size:32  //粒子最终大小  // endSize: 32  velocity: AngleDirection {  angle: 0  angleVariation: 15  magnitude: 100  magnitudeVariation: 100  }  }  //用来做粒子的图片,需要指明source和system  ImageParticle{  source:"images/particle.png"  system: *particleSystem*  color: '#FFD700'  colorVariation: 0.2  rotation: 0  rotationVariation: 45  // rotationVelocity: 15  rotationVelocityVariation: 15  entryEffect: ImageParticle.Scale  }    //粒子控制Turbulence  Turbulence{  anchors.horizontalCenter: *parent*.horizontalCenter  width: 240; height: 240  system: *particleSystem*  strength: 100  Tracer{}  }  }  } |  |

### 17. 漂移（ Wander）,其实就是抖动效果

|  |
| --- |
| 漂移控制器控制了轨迹。 affectedParameter属性可以指定哪个参数控制了漂移（ 速 度， 位置或者加速度） 。 pace属性制定了每秒最多改变的属性。 yVariance指定了y 组件对粒子轨迹的影响。 |

实例

|  |  |
| --- | --- |
| import QtQuick 2.15  import QtQuick.Window 2.15  import QtQuick.Particles 2.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Particle Demo")  //粒子控制1.Age  Rectangle{  id:*root*  x:10  y:10  width: 480  height: 240  anchors.horizontalCenter: *parent*.horizontalCenter  color: "#1f1f1f"  //实例化一个粒子系统  ParticleSystem{  id:*particleSystem*  }  //实例化一个粒子发射器  Emitter{  id:*emitter*  anchors.centerIn: *parent*  //width: 160;height: 80  anchors.fill: *parent*  system: *particleSystem*  //每一次发射粒子数  emitRate: 30  //粒子的生命周期  lifeSpan: 6400  //粒子变化周期  lifeSpanVariation: 400  //粒子大小  size:32  //粒子最终大小  // endSize: 32  velocity: AngleDirection {  angle: 0  angleVariation: 15  magnitude: 100  magnitudeVariation: 100  }  }  //用来做粒子的图片,需要指明source和system  ImageParticle{  source:"images/particle.png"  system: *particleSystem*  color: '#FFD700'  colorVariation: 0.2  rotation: 0  rotationVariation: 45  // rotationVelocity: 15  rotationVelocityVariation: 15  entryEffect: ImageParticle.Scale  }    //粒子控制Wander  Wander{  anchors.horizontalCenter: *parent*.horizontalCenter  width: 240; height: 240  system: *particleSystem*  //设置受影响的参数  affectedParameter: Wander.Position  pace: 200//每秒改变粒子y方向的位置200次。  yVariance: 240  //xVariance: 100  Tracer{  }  }  }  } |  |

# 四.粒子组ParticleGroup

|  |
| --- |
| 默认下， 粒子都属于空组（ ""） 。 使用GroupGoal控制器可以改变粒子组。 为了实现可视化， 我们创建了一个烟花示例，火箭进入， 在空中爆炸形成壮观的烟火。这个例子分为两部分。 第一部分叫做“发射时间（ Launch Time） ”连接场景， 加入粒子组， 第二部分叫做“爆炸烟花（ Let there be firework） ”， 专注于粒子组的变化。 |

## 火箭烟花实例

|  |  |
| --- | --- |
| //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtQuick.Particles 2.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Particle Demo")  //粒子组ParticleGroup  Rectangle{  id:*root*  x:10  y:10  width: 480  height: 240  property bool tracer:false  anchors.horizontalCenter: *parent*.horizontalCenter  color: "#1f1f1f"  //实例化一个粒子系统  ParticleSystem{  id:*particleSystem*  }  //rocket组  ImageParticle{  id:*rockPainter*  groups: ['rocket']  system: *particleSystem*  source: "images/rocket.png"  entryEffect: ImageParticle.Fade  }  //smoke组  ImageParticle{  id:*smokePainter*  groups:[ 'smoke']  system: *particleSystem*  source: "images/particle.png"  alpha: 0.3  }  //sparkl组  ImageParticle{  id:*sparklePainter*  groups: ['sparkle']  system: *particleSystem*  source: "images/star.png"  color: "red"  colorVariation: 0.6 //这个非常重要,如果没有这个设置,就不会有色彩缤纷的效果,只是红色非常难看  alpha: 0.3  }  //发射器,方式rocket,在下方  Emitter{  id:*rocketEmitter*  anchors.bottom: *parent*.bottom  width: *parent*.width; height: 40  system: *particleSystem*  group: 'rocket' //属于rocket组  emitRate:2  maximumEmitted: 8  lifeSpan: 4800  lifeSpanVariation: 400  size: 128  velocity: AngleDirection{angle: 270; magnitude: 150; magnitudeVariation: 10}  acceleration: AngleDirection{angle:90; magnitude: 50} //重力加速度梯度magnitude是什么意思?  Tracer{color:'green'; visible: *root*.tracer}  }  //追踪发射器,追踪火箭组,根据一个逻辑particle来发射粒子  TrailEmitter{  id:*someEmitter*  system: *particleSystem*  group: 'smoke'  follow: 'rocket'  size: 16  sizeVariation: 8  emitRatePerParticle: 16  velocity: AngleDirection{angle: 90; angleVariation: 15; magnitude: 100}  lifeSpan: 200  Tracer { color: 'blue'; visible: *root*.tracer }  }  //粒子控制,摩擦friction,控制的是rocket组  Friction{  groups: ['rocket']  factor: 0.9  anchors.top: *parent*.top  system: *particleSystem*  width: *parent*.width; height: 80  threshold: 5  }  //粒子控制,紊流,控制的是rocket组  Turbulence{  groups: ['rocket']  system: *particleSystem*  anchors.bottom: *parent*.bottom  width: *parent*.width; height: 160  strength: 25  Tracer { color: 'green'; visible: *root*.tracer }  }  //组目标,设置的是rocket组  GroupGoal{  id:*rocketChanger*  anchors.top: *parent*.top  width: *parent*.width; height: 80  system: *particleSystem*  groups: ['rocket']  //设置效果  goalState: 'explosion'  //是否跳跃  jump: true  Tracer { color: 'blue'; visible: *root*.tracer }  }  //粒子组  ParticleGroup{  name: 'explosion'  system: *particleSystem*  //需要两个追踪发射器,追踪的是rocket组  TrailEmitter{ //第一种爆炸效果  id:*explosionEmitter*  anchors.fill: *parent*  group: 'sparkle'  follow: 'rocket'  lifeSpan: 750  emitRatePerParticle: 200  size: 32  velocity: AngleDirection{angle:90; angleVariation: 180; magnitude: 50}  }  TrailEmitter{ //第二种爆炸效果  id:*explosion2Emitter*  anchors.fill: *parent*  group: 'sparkle'  follow: 'rocket'  lifeSpan: 250  emitRatePerParticle: 100  size: 32  velocity: AngleDirection{angle:90; angleVariation: 15; magnitude: 400}  }  }  }  } |  |

# 粒子系统总结

粒子是一个非常强大且有趣的方法， 用来表达图像现象的一种方式， 比如烟， 火花， 随机可视元素。 Qt5的扩展API非常强大， 我们仅仅只使用了一些浅显的。 有一些元素我们还没有使用过， 比如精灵（ spirites） ， 尺寸表（ size tables） ， 颜色表 color tables 。 粒子看起来非常有趣， 它在界面上创建引人注目的东西是非常有潜力的。 在一个用户界面中使用非常多的粒子效果将会导致用户对它产生这是一个游戏的印象。 粒子的真正力量也是用来创建游戏。