Unity 工具

一. Terrain Tool

1. 快捷键

① A＋ 鼠标拖动 调整笔刷强度
② S＋ 鼠标拖动 调整笔刷大小
③ D＋ 鼠标拖动 旋转笔刷
④ Shift 使笔刷变得平滑
⑤ Ctrl 反转笔刷功能，地形的升高变为降低

二. Cinemachine

三. Recorder

手册地址：https://docs.unity3d.com/Packages/com.unity.recorder@3.0/manual/index.html

1. 下载

在 Unity 中找到 Window 下的 Package Manager，将 package Manager 界面显示出来。

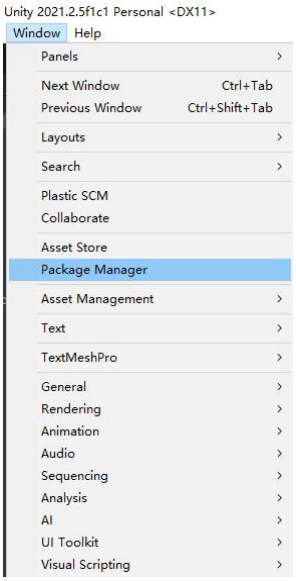


图 1. 图 3.1 打开 PackageManager

打开了 package Manager 界面后，在 Unity Package 界面搜索 Recorder,找到并点击 Install 将其导入到 Unity 中。

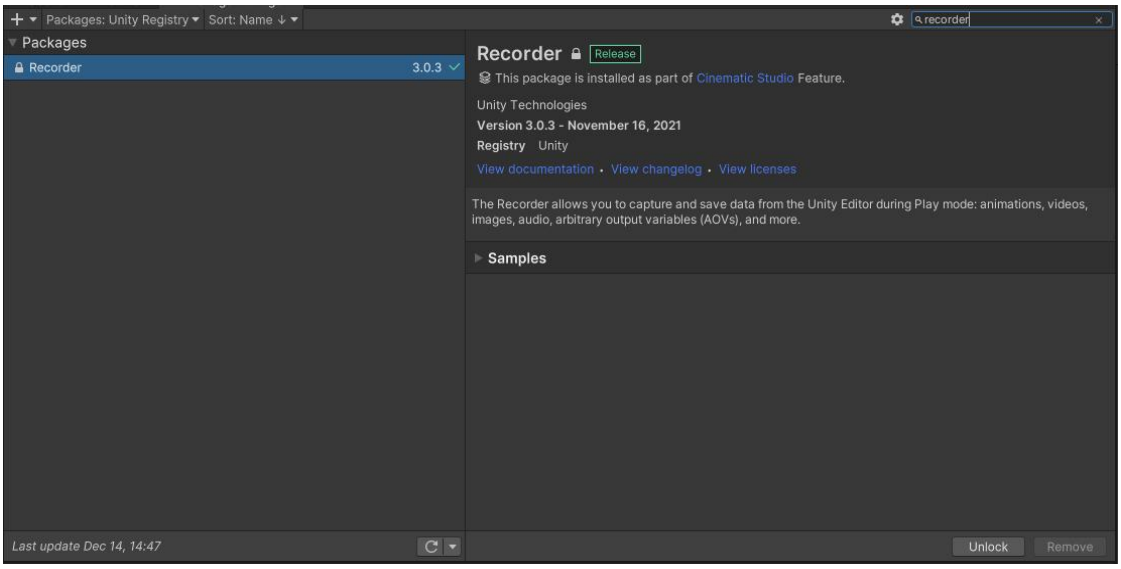


图 2.　　图 3.2 下载 Recorder

2. 打开 Recorder 界面

从 Unity 的主菜单（Window > General > Recorder > Recorder window）打开 Recorder 窗口。

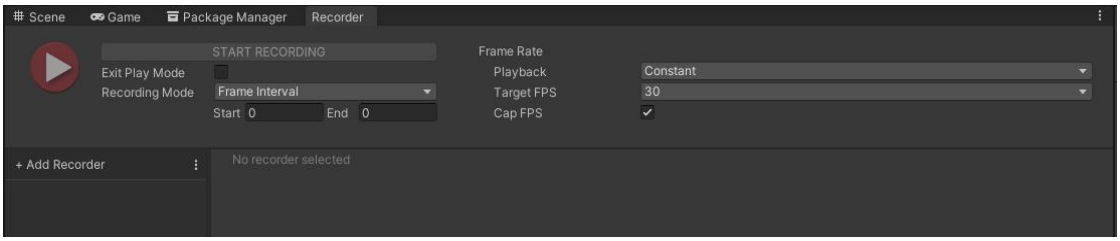


图 3.　　图 3.3 Recorder 界面

可以使用 Recorder 输出以下格式的数据。

Animation Clip Recorder：生成 .anim 格式的动画剪辑。

Movie Recorder：生成 .mp4 或 .webm 格式的视频。

Image Sequence Recorder：生成 .jpeg、.png 或 .exr (OpenEXR) 格式的图像文件序列。

Audio Recorder：生成 .wav 格式的音频剪辑。

3. Recording Mode(记录模式)

① Manual

当您分别手动单击开始录制或停止录制时开始或停止录制。

② Single Frame

记录单帧。使用目标框架属性指定此框架。

③ Frame Interval

在播放模式下记录一组连续的帧。使用 Start 和 End 属性指定开始和停止的时间。

④ Time Interval

在播放模式下记录特定的持续时间（以秒为单位）。使用 Start 和 End 属性指定开始和停止的时间。

4. Frame Rate 帧率属性

使用帧速率属性指定如何在录制期间限制帧速率。帧速率会影响 Recorder 输出的文件的大小和数量。

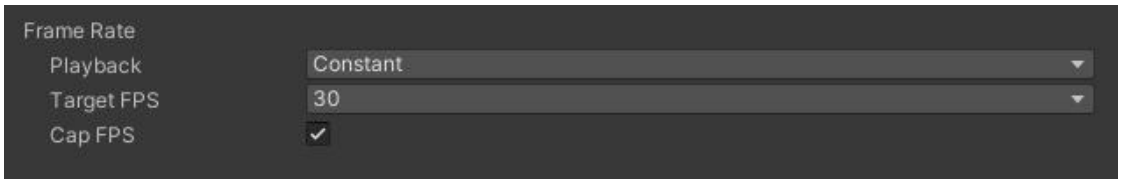


图 4.　　图 3.4 Frame Rate

1)　Playback

指定如何在录制过程中控制帧率。

①　Constant（恒定）

将记录器限制为特定的帧速率。使用 Target 属性指定此速率。

②　Variable（变化）

使用应用程序的帧速率。使用 Max FPS 属性指定录制期间应用程序的速率上限。

注：该记录视频时不支持变帧率。

2)　Target FPS

设置捕获记录的帧速率。当您将 Playback 设置为 Constant 时，会出现此属性。

无论您以更高还是更低的帧速率运行应用程序，Recorder 都会以该速率进行捕获。例如，如果您将目标 FPS 设置为 30 fps 的自定义值，但您以 60 fps 运行应用程序，则记录器以 30 fps 进行捕获。

3)　Max FPS

限制播放模式下的更新速率。当您将 Playback 设置为 Variable 时，此属性可用。为防止您的应用程序超过此帧速率，记录器会在播放期间插入延迟。使用此属性可减小输出的文件大小。

4)　**文件输出**

①　可用占位符

可以在文件名和路径字段中插入占位符，以在文件名和路径字符串中包含自动生成的文本，记录器用于保存输出文件。Recorder 用适当的上下文值替换这些占位符以构建实际字符串。

可以在同一字符串中组合多个占位符。例如，您生成录音的日期和时间。

在+通配符菜单可以帮助您快速添加它们的文件名字符串，还可以手动将字符串中的任何输入。要在路径字符串中使用占位符，必须手动输入它们。

注意：确保在占位符名称中包含 <> 括号。另请注意，占位符名称区分大小写。

占位符	Description-描述	输出示例
<Recorder>	• 在记录器窗口中：记录器列表中的记录器名称。 • 在时间轴轨道中：记录器剪辑的名称。	My Recorder
<Time>	记录生成的时间。使用 00h00m 格式。	16h52m
<Take>	编号值。使用 000 格式。	002
<Date>	记录的生成日期。使用 yyyy-MM-dd 格式。	2020-11-03
<Project>	当前 Unity 项目的名称。	My Project
<Product>	产品名称 Unity 的播放器常	My Product

	规设置中的字段。	
<Scene>	当前 Unity 场景的名称。	My Scene
<Resolution>	以像素为单位的输出图像尺寸（宽 x 高）。使用 WxH 格式。	1920x1080
<Frame>	当前帧数。使用 0000 格式。 这对于将每一帧作为单独文件输出的图像序列记录器很有用。	0154
<Extension>	输出格式的文件扩展名（不带句号）。	png
<GameObject>	正在录制的游戏对象的名称。	My GameObject
<GameObjectScene>	包含正在录制的游戏对象的场景的名称。	My Scene

图 5.　　表 3.1 占位符功能表

5. Capture-捕捉设置

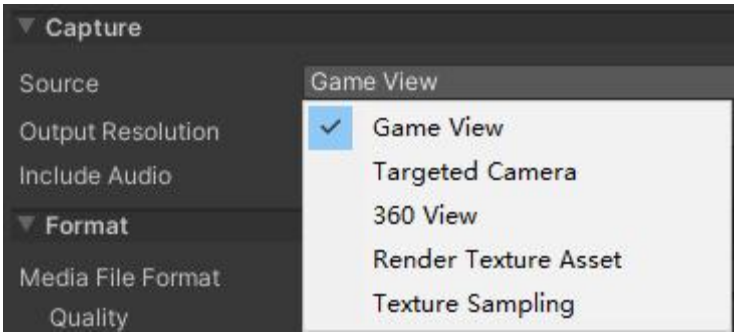


图 6.　　图 3.5 捕捉设置

1) Source-捕捉源

① Game View

记录在游戏视图中渲染的帧。

② Targeted Camera

记录由特定相机捕获的帧，即使游戏视图不使用该相机。

③ 360 View

录制 360 度视频。

④ Render Texture Asset

记录渲染纹理中渲染的帧。

⑤ Texture Sampling

在捕获过程中对源摄像机进行超采样以在记录中生成抗锯齿图像。

2) Output Resolution-输出分辨率

① Match Window Size

匹配当前选择的游戏视图的分辨率和纵横比。

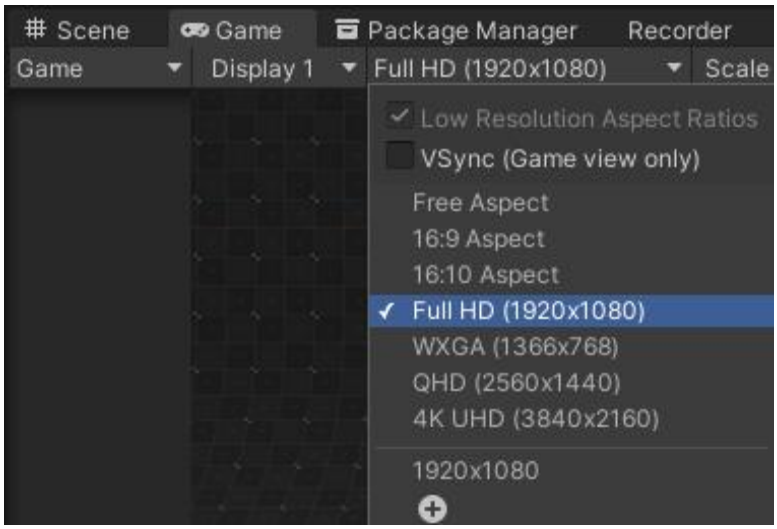


图 7.　　图 3.6 游戏视图的分辨率

② Custom-自定义

使用用户在 W 和 H 字段中提供的自定义宽度和高度值。

四. Post Processing

后处理是全屏图像处理效果的总称，它发生在相机绘制场景之后但在场景渲染到屏幕上之前。后处理可以极大地改善产品的视觉效果，而只需很少的设置时间。

您可以使用后处理效果来模拟物理相机和胶片属性。

1. 开始后处理

1) 后处理层-Post-process Layer

要在场景中启用后期处理，请将 Rendering > Post Process Layer 组件添加到 Main Camera GameObject。此组件允许您为此后处理层配置抗锯齿，选择即将应用后处理的层，并选择哪些游戏对象能够触发此后处理层。

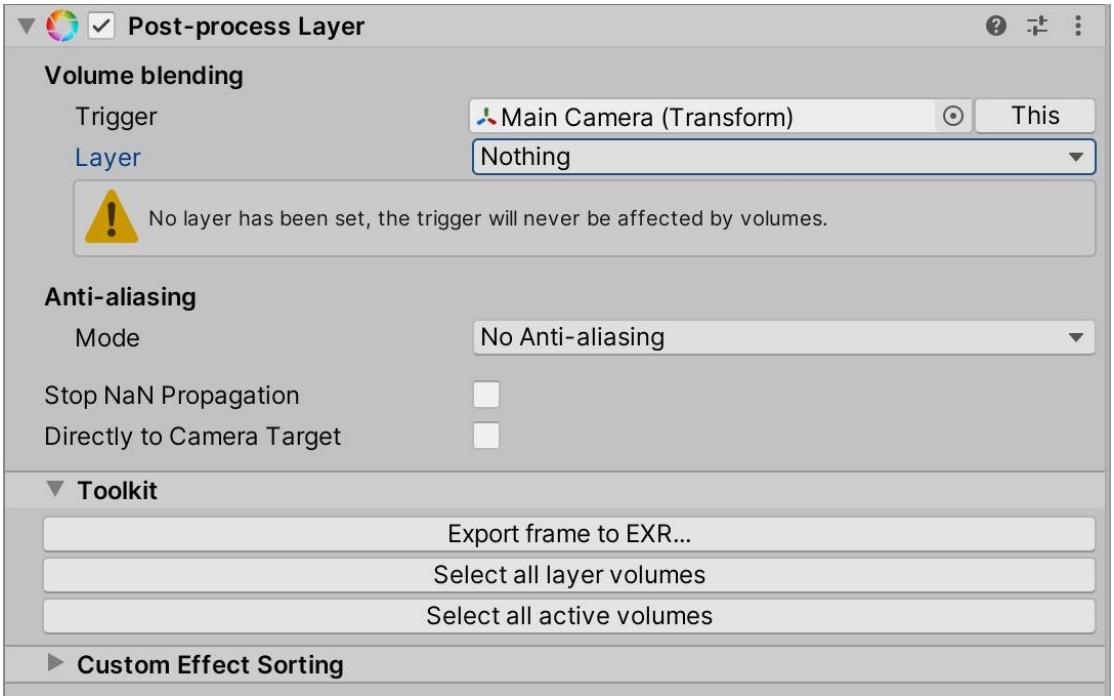


图 8.　　图 12.1 Post Process Layer

Volume blending

2. 泛光-Bloom

1) 作用

Bloom 效果使图像中的明亮区域发光。

Bloom 效果还具有 Dirtiness 功能，您可以使用它来应用全屏的污迹或灰尘层来衍射 Bloom 效果，本质上就是往整个画面上叠加一张纹理。

2) Bloom settings

强度-Intensity	设置泛光的强度
临界值-Threshold	用于过滤亮度小于此数值的像素。也就是说，亮度小于此数值的像素不会参与泛光的计算。
渐变阈值-Soft Knee	为低于或高于阈值之间的过渡设置渐变阈值（0 = 硬阈值，1 = 软阈值）
Clamp	设置钳位像素的值以控制 Bloom 量。该值以伽马空间表示。
扩散-Diffusion	以与屏幕分辨率无关的方式设置遮蔽效果的程度。
镜头畸变比例-Anamorphic Ratio	设置畸变比例以垂直（范围 [-1,0]）或水平（范围 [0,1]）缩放 Bloom 。这模拟了变形镜头的效果。
Color	选择泛光的色调颜色。
快速模式-Fast Mode	启用此复选框可通过降低 Bloom 效果质量来提高性能。

3) Dirtiness settings:

纹理-Texture	为泛光叠加一张纹理，比如包含光晕或者灰尘的纹理
Intensity	用于调整纹理叠加的强度

4) 细节

对于正确曝光的 HDR 场景，Threshold（临界值）应将 设置为 1，以便只有值大于 1 的像素会泄漏到周围的对象中。在 LDR 中工作时降低此值，否则 Bloom 效果将不可见。

5) 表现

降低 Diffusion 参数将使效果更快。Anamorphic Ratio 离 0 越远，速度越慢。如果您正在为移动或低端平台开发想获得显着的性能提升，请启用 Fast Mode。

较低分辨率的镜头 Dirtiness Texture 会导致更快的查找和跨体积混合。

五. 缓动函数

缓动曲线在 UI 动画中的应用十分广泛。缓动曲线可以用来控制动画的运动速率，使其按照我们的意愿模拟真实物体的运动规律。举个例子，当我们往上抛出一个石块时，在不考虑空气阻力的情况下，石块会在重力的作用下，先匀减速上升，直至速度为零。而后，石块的速度又会从零开始匀加速下降。那么，如何模拟这样的运动过程呢？在高中物理课上，我们已经知道，石块的位移和时间之间满足平方关系：

$$h(t) = v_0 t - \frac{1}{2} g t^2 + h_0$$

其中，h(t) 是石块在 t 时刻的高度，v_0 是石块的初速度，g 是重力加速度，h_0 是石块的初始高度。从这个式子可以看出，我们需要一条二次方的运动曲线（抛物线）来模拟这样的运动过程。而类似这样的一条曲线，就是缓动曲线。

总的来说，缓动曲线包含四大类，分别是线性（linear）、缓入（ease in）、缓出（ease out）和缓入缓出（ease in and out）。除了线性类，其余三大类又可以细分出各种子类，比如二次方缓动曲线（Quadratic）就是其中的一个子类。以上文提到的石块为例，石块的运动曲线满足二次方缓动曲线，而上升过程满足缓出的过程（速度先快后慢），下降过程则满足缓入过程（速度先慢后快）。

1. 参数解析

首先说说四个参数的含义。

① t: time 动画执行到当前帧所进过的时间

② b: start 起始位置

③ c: change 总移动距离，即需要变化的量

④ d: duration 动画总时长

对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是 x 轴长度,start 和 change 分别是 y 轴起点和长度，返回值是点 a 对应的 y 轴坐标。

PS：

```
public enum LoopType
    {
        Clamp = 0,
        Loop = 1,
    }

public static float GetTime(float time, float duration , LoopType type = LoopType.Clamp)
        {
            switch (type)
            {
                case LoopType.Clamp:
                    return time > duration ? duration : time;
                case LoopType.Loop:
                    return time % duration;
                default:
                    return time;
            }
        }
```

2. 数学原理

首先要清楚一点，动画中每一帧所经过的时间是相同的，只是由于上一帧与下一帧的位移量不同，因此速度在视觉上感受不同，位移量小，感觉上速度就慢了。

① time 的变化可表达为 0 –> d，提取出常数 d，就变成 d*(0 -> 1)，变化部分为(0->1)，记为 x 轴的变化。

② 动画总的变化量和开始值是已知的，其变化可以表达为 b -> b+c,提取一下变为 b+c*(0 -> 1)，变化部分也是(0->1)，记为 y 轴的变化。

③ t 用来指示事件当前的时间点，将其变为指示动画完成的百分比，即 t/d（当前动画时间除以总动画时间）；

④ 通过上面的变换，我们需要做的事情就是构造 x 轴区间为[0,1],y 轴区间也为[0,1]的线性或者非线性关系了。线性关系多数是 y = x，也就是常用的 linear 了，非线性关系复杂一点。

⑤ 然后我们看看可以构造出哪些非线性关系，并给出函数关系表达式。

利用指数函数(x 的 n 次方)可以构造一大堆 easein 的效果，再根据他们的轴对称或者中心对称做翻转和位移，又可以构造出其对应的 easeout 效果。

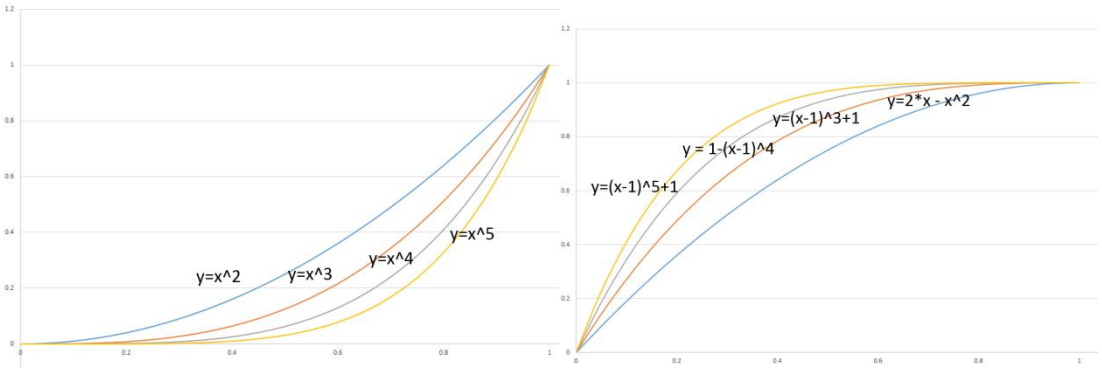


图 9. 指数函数构造的 easein 和 easeout

3. 线性缓动-Linear

线性缓动的计算公式为 f(x) = x。

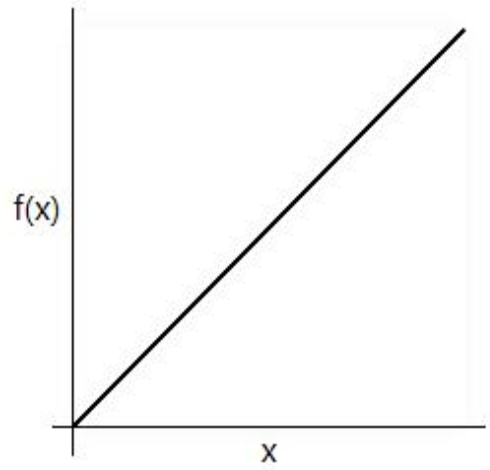


图 10. Linear 缓动图示

PS：

public static float Linear(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)

```
{
    time = GetTime(time, duration, type);
    return change * time / duration + begin;
}
```

4. 正弦缓动-Sine

1) EaseIn-缓入

开始时速度很慢，然后逐渐加快。结尾会突然停止,感觉很生硬，$y = 1 - \cos(0.5\pi x)$。

将 ease-in 映射到横向移动上,可以观察到开始很慢，然后逐渐变快，在结尾时速度最快。

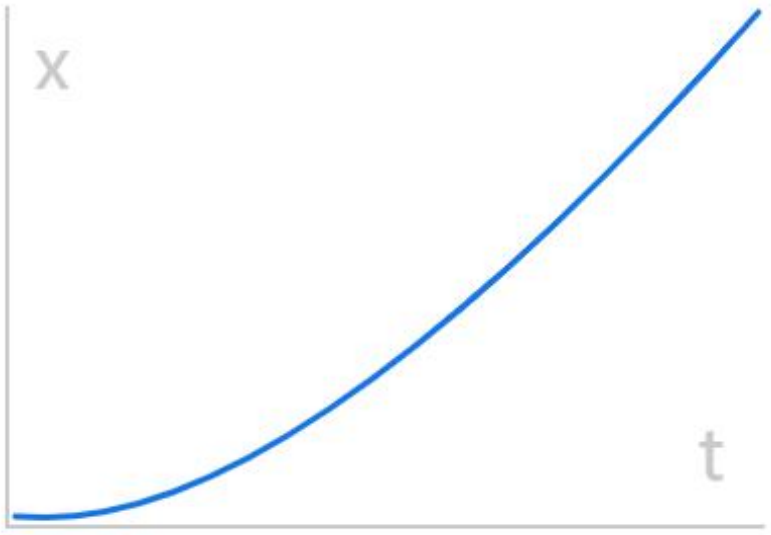


图 11.　图 88.　EaseInSin

图 12.　EaseInSine 移动速度图像

PS：

public static float EaseInSine(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)

{

time = GetTime(time, duration, type);

//1-cos(x * PI/2)

return change - Mathf.Cos((time / duration) * (Mathf.PI/2)) + begin;

}

2)　EaseOut-缓出

常用的变速运动，开始时速度很快。给人一种流畅感，然后逐渐减速，不会让人觉得戛然而止，$y = \sin(0.5\pi x)$。（其实就是 EaseIn 函数的镜像翻转）

将 ease-out 映射到横向移动上,可以观察到开始时速度最快，然后逐渐变慢,在结尾时速度最慢。

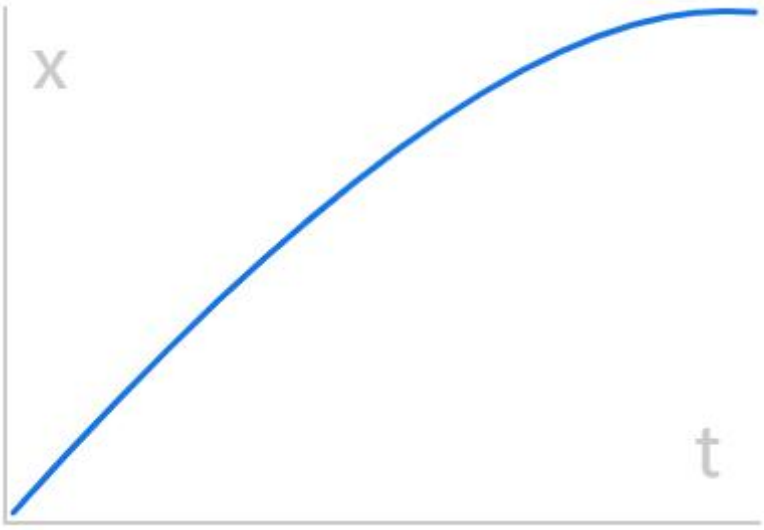


图 13.　EaseOutSine

图 14.　EaseOutSine 移动速度图像

PS：

public static float EaseOutSine(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)

{

time = GetTime(time, duration, type);

//sin(x * PI/2)

return change * Mathf.Sin(time * (Mathf.PI / 2)) + begin;

}

3)　EaseInOut-缓入缓出

开始和结尾时慢，中间快。比缓出更生动，但动画时间不宜过长，最好在 300~500ms 之间。

图 15.　easeInOutSine

ease-in-out 和 ease 非常相似,不同点在于 ease 的开始速度比结束速度更快一些。ease-in-out 是 ease-in 和 ease-out 的结合,前半段用 ease-in 的计算公式,后半段用 ease-out 的计算公式。

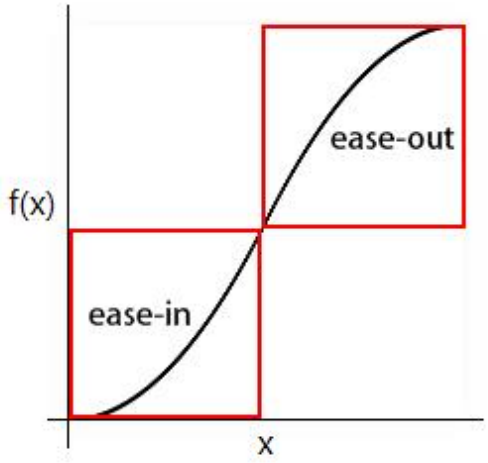


图 16.　EaseInOutSine 的分段计算

将 ease-in-out 映射到横向移动上,可以观察到开始速度慢，然后逐渐变快,到中间时达到最大值，然后又逐渐减慢。

图 17.　图 89.　EaseInOutSine 移动速度图像

PS：

public static float EaseInOutSine(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)

```
{
    time = GetTime(time, duration, type);
    // -(cos(PI * x) - 1) / 2
    return - change * (Mathf.Cos(time * Mathf.PI) - 1) / 2 + begin;
}
```

5. 二次方缓动-Quadratic

1) EaseIn-缓入

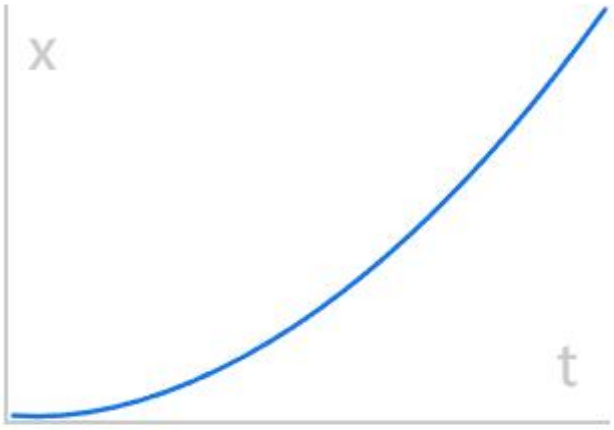


图 18. EaseInQuad

PS：

```
public static float EaseInQuad(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            // x^2
            return change * (time /= duration * time) + begin;
        }
```

2) EaseOut-缓出

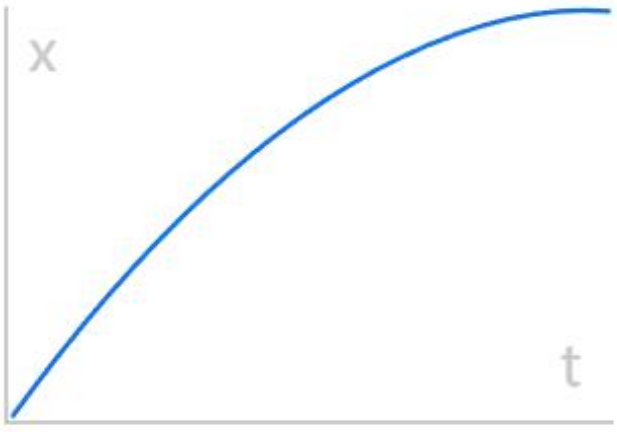


图 19. EaseOutQuad

PS：

```
public static float EaseOutQuad(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            // 1 - (1 - x)*(1 - x)
            return -change * (time /= duration) * (time - 2) + begin;
```

}

3) EaseInOut-缓入缓出

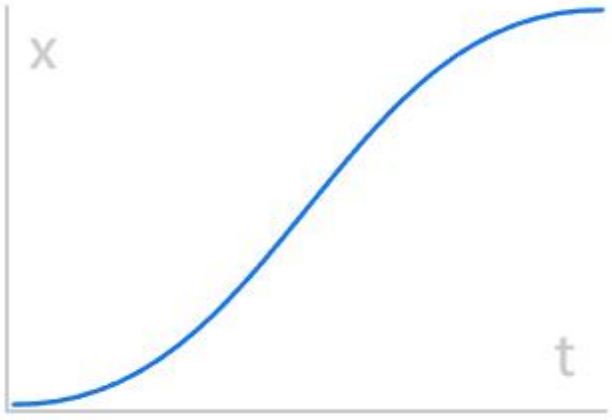


图 20. EaseInOutQuad

PS：

public static float EaseInOutQuad(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)

{

time = GetTime(time, duration, type);

//time < 0.5 ? 2 * time * time : 1 - pow(-2 * time + 2, 2) / 2;

if ((time /= duration / 2) < 1)

{

return change / 2 * time * time + begin;

}

return -change / 2 * ((--time) * (time - 2) - 1) + begin;

}

6. 三次方缓动-Cubic

1) EaseIn-缓入

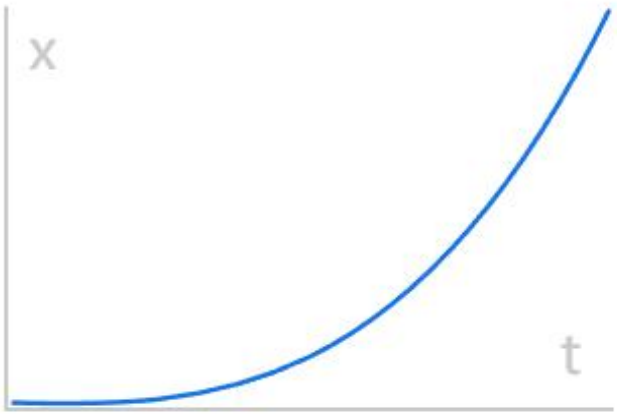


图 21. EaseInCubic

PS：

public static float EaseInCubic(float time, float begin, float change, float duration, LoopType

```
type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //x * x * x
            return change * (time /= duration) * time * time + begin;
        }
```

2) EaseOut-缓出

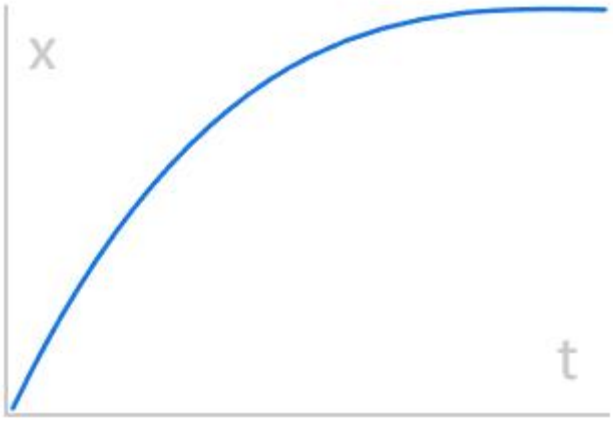


图 22. EaseOutCubic

PS：

```
public static float EaseOutCubic(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //1 - pow(1 - x, 3);
            return change * ((time = time / duration - 1) * time * time + 1) + begin;
        }
```

3) EaseInOut-缓入缓出

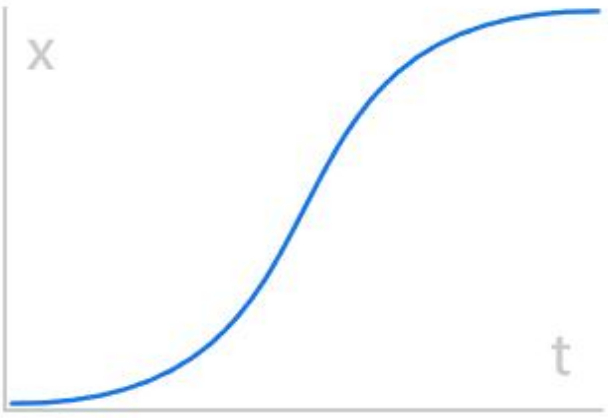


图 23. EaseInOutCubic

PS：

```
public static float EaseInOutCubic(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
```

```
{
    time = GetTime(time, duration, type);
    //x < 0.5 ? 4 * x * x * x : 1 - pow(-2 * x + 2, 3) / 2;
    if ((time /= duration / 2) < 1)
        return change / 2 * time * time * time + begin;
    return change / 2 * ((time -= 2) * time * time + 2) + begin;
}
```

7. 四次方缓动-Quartic

1) EaseIn-缓入

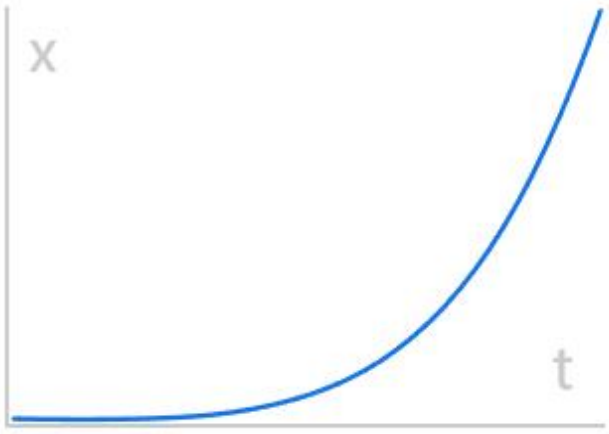


图 24. EaseInQuart

PS：

public static float EaseInQuart(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)

```
{
    time = GetTime(time, duration, type);
    //x * x * x * x
    return change * (time /= duration) * time * time * time + begin;
}
```

2) EaseOut-缓出

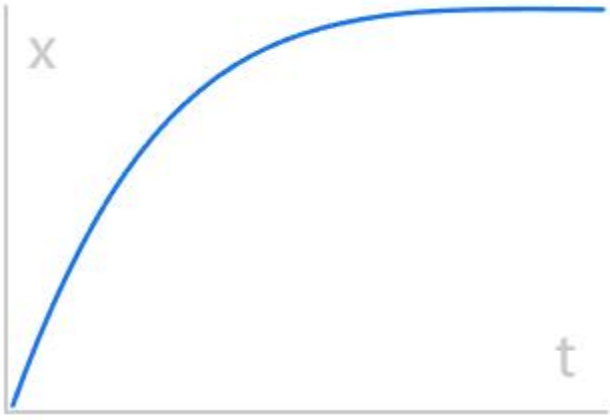


图 25. EaseOutQuart

PS：

public static float EaseOutQuart(float time, float begin, float change, float duration,

```
LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //1 - pow(1 - x, 4);
            return -change * ((time = time / duration - 1) * time * time * time - 1) + begin;
        }
```

3) EaseInOut-缓入缓出

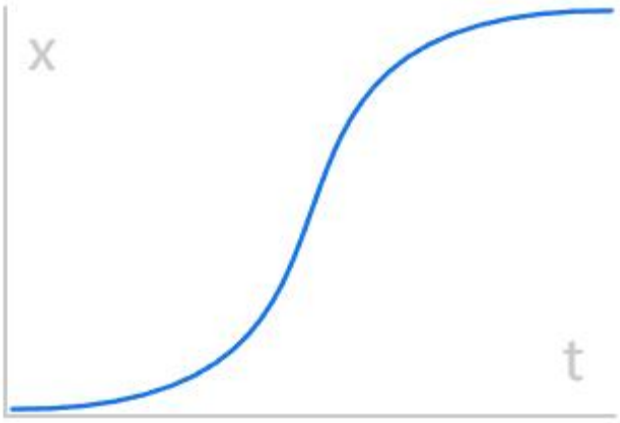


图 26. EaseInOutQuart

PS：

```
public static float EaseInOutQuart(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            // x < 0.5 ? 8 * x * x * x * x : 1 - pow(-2 * x + 2, 4) / 2
            if ((time /= duration / 2) < 1)
                return change / 2 * time * time * time * time + begin;
            return -change / 2 * ((time -= 2) * time * time * time - 2) + begin;
        }
```

8. 五次方缓动-Quintic

1) EaseIn-缓入

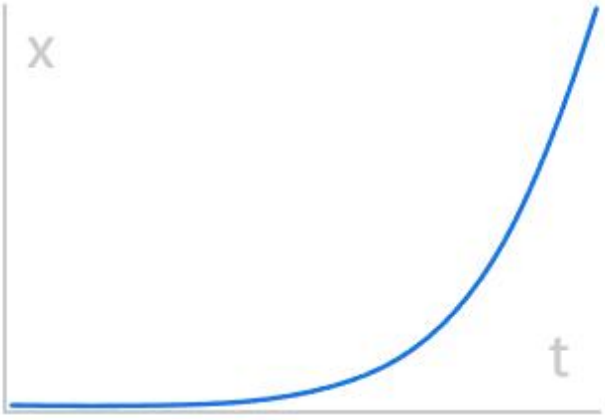


图 27. EaseInQuint

PS：

public static float EaseInQuint(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)

{

time = GetTime(time, duration, type);

//x^5

return change * (time /= duration) * time * time * time * time + begin;

}

2) EaseOut-缓出

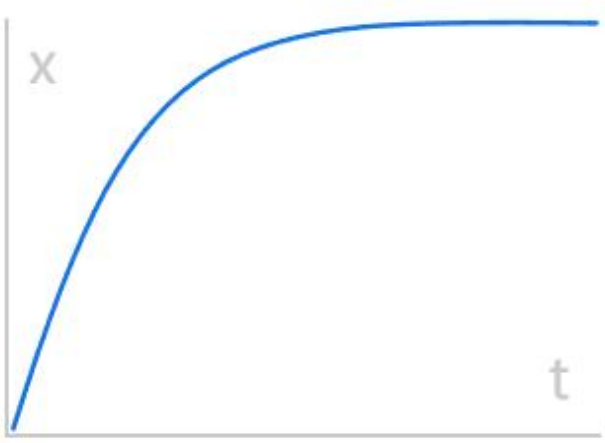


图 28. EaseOutQuint

PS：

public static float EaseOutQuint(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)

{

time = GetTime(time, duration, type);

//1 - pow(1 - x, 5);

return change * ((time = time / duration - 1) * time * time * time * time - 1) + begin;

}

3) EaseInOut-缓入缓出

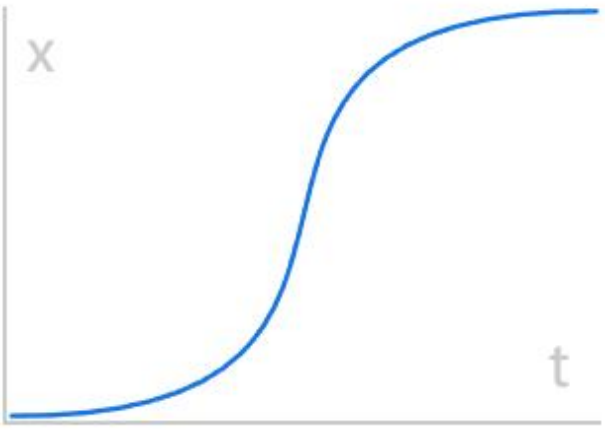


图 29.　EaseInOutQuint

PS：

public static float EaseInOutQuint(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)

{

time = GetTime(time, duration, type);

// x < 0.5 ? 16 * x * x * x * x * x : 1 - pow(-2 * x + 2, 5) / 2

if ((time /= duration / 2) < 1)

return change / 2 * time * time * time * time * time + begin;

return change / 2 * ((time -= 2) * time * time * time * time + 2) + begin;

}

9. 指数缓动-Exponential

1) EaseIn-缓入

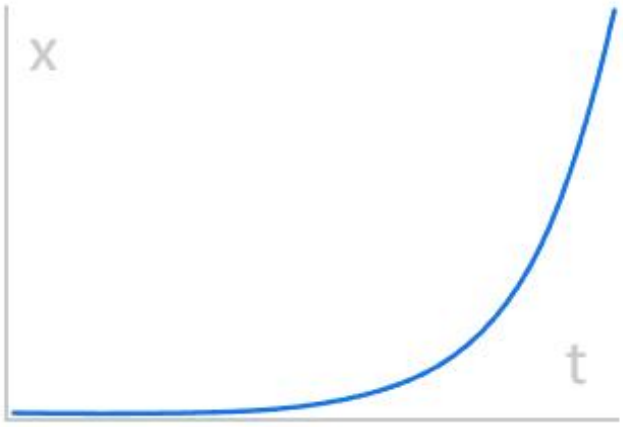


图 30.　EaseInExpo

PS：

public static float EaseInExpo(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)

{

time = GetTime(time, duration, type);

//x === 0 ? 0 : pow(2, 10 * x - 10);

if (time == 0)

{

return begin;

}

else

{

return change * Mathf.Pow(2, 10 * (time / duration - 1)) + begin;

}

}

2) EaseOut-缓出

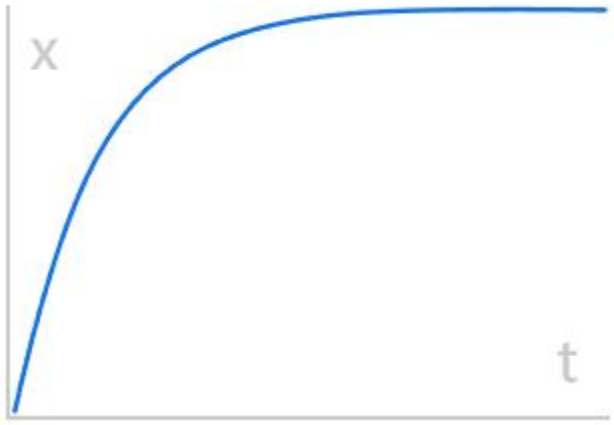


图 31. EaseOutExpo

PS：

```
public static float EaseOutExpo(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //x === 1 ? 1 : 1 - pow(2, -10 * x)
            if (time == duration)
            {
                return begin + change;
            }
            else
            {
                return change * (-Mathf.Pow(2, 10 * time / duration) + 1) + begin;
            }
        }
```

3) EaseInOut-缓入缓出

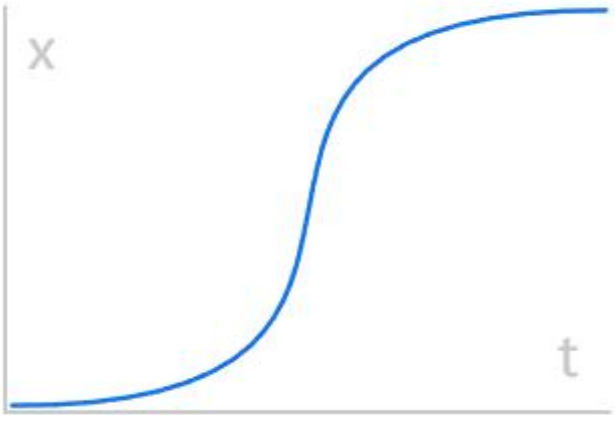


图 32. EaseInOutExpo

PS：

```
public static float EaseInOutExpo(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //当时间为 0 时，返回的 Y 值就是设定的开始位置
            if (time == 0)
                return begin;
            //当时间为动画总时间，返回的 Y 值就是起始值加上 Y 轴的总长度
            if (time == duration)
                return begin + change;
            if ((time /= duration / 2) < 1)
                return change / 2 * Mathf.Pow(2, 10 * (time - 1)) + begin;
            return change / 2 * (-Mathf.Pow(2, -10 * --time) + 2) + begin;
        }
```

10. 圆形曲线缓动-Circular

1) EaseIn-缓入

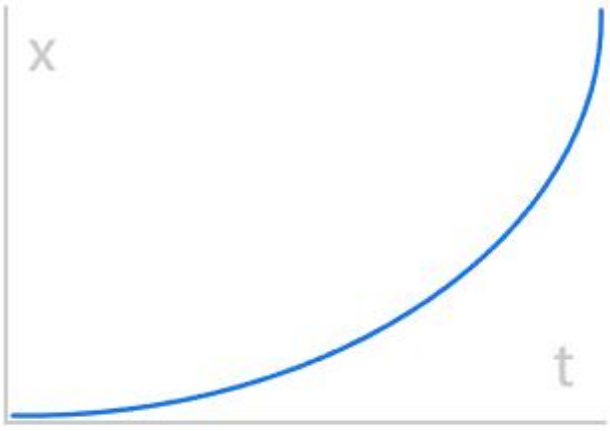


图 33. EaseInCirc

PS：

```
public static float EaseInCirc(float time, float begin, float change, float duration, LoopType
type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //1 - sqrt(1 - pow(x, 2))
            return -change * (Mathf.Sqrt(1 - (time /= duration) * time) - 1) + begin;
        }
```

2) EaseOut-缓出

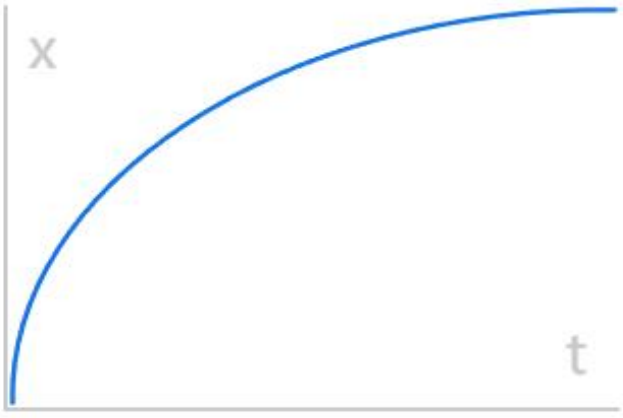


图 34. EaseOutCirc

PS：

```
public static float EaseOutCirc(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //sqrt(1 - pow(x - 1, 2));
            return change * Mathf.Sqrt(1 - (time /= duration - 1) * time) + begin;
        }
```

3) EaseInOut-缓入缓出

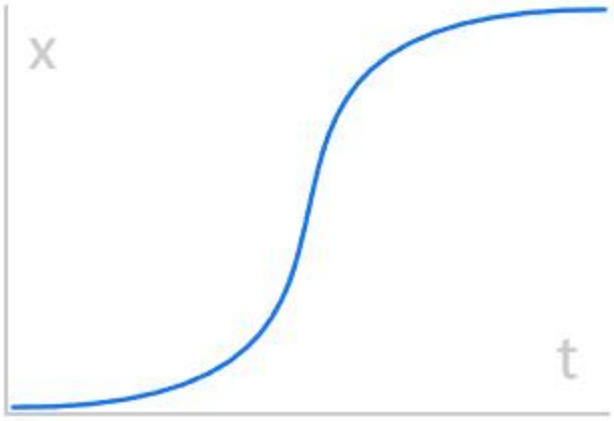


图 35. EaseInOutCirc

PS：

```
public static float EaseInOutCirc(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //x < 0.5    ? (1 - sqrt(1 - pow(2 * x, 2))) / 2    : (sqrt(1 - pow(-2 * x + 2, 2)) + 1) / 2;
            if ((time /= duration / 2) < 1)
```

```
            return -change / 2 * (Mathf.Sqrt(1 - time * time) - 1) + begin;
        return change / 2 * (Mathf.Sqrt(1 - (time -= 2) * time) + 1) + begin;
    }
```

11. 向后缓动-Back

1) EaseIn-缓入

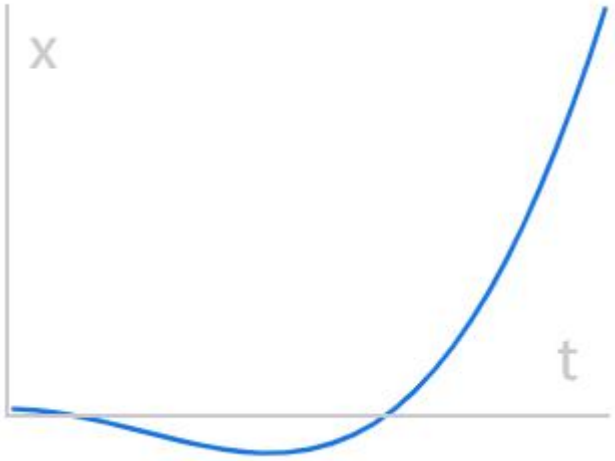


图 36. EaseInBack

PS：

```
public static float EaseInBack(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)
    {
        float s = 0;
        time = GetTime(time, duration, type);
        if (s == 0)
            s = 1.70158f;
        return change * (time /= duration) * time * ((s + 1) * time - s) + begin;
    }
```

2) EaseOut-缓出

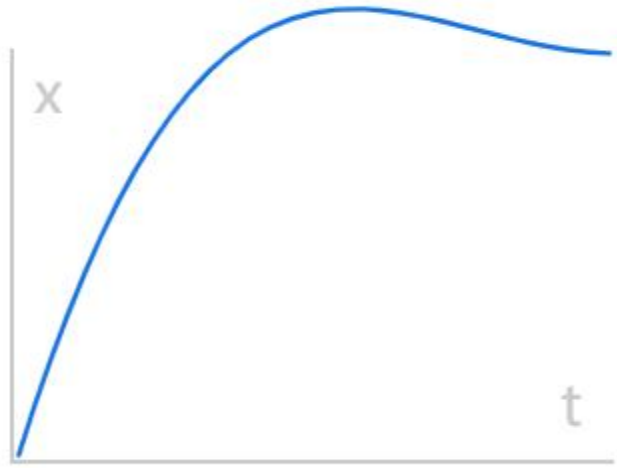


图 37. EaseOutBack

PS：

```
public static float EaseOutBack(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
        {
            float s = 0;
            time = GetTime(time, duration, type);
            if (s == 0)
                s = 1.70158f;
            return change * ((time = time / duration - 1) * time * ((s + 1) * time + s) + 1) +
begin;
        }
```

3) EaseInOut-缓入缓出

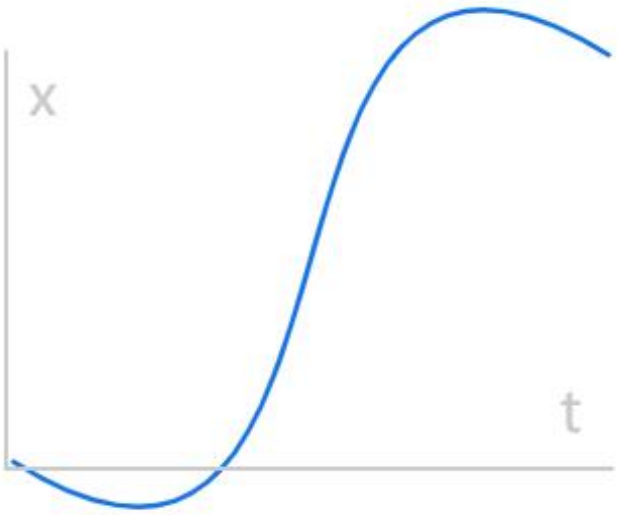


图 38. EaseInOutBack

PS：

```
public static float EaseInOutBack(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
        {
            float s = 0;
            time = GetTime(time, duration, type);
            if (s == 0)
                s = 1.70158f;
            if ((time /= duration / 2) < 1)
                return change / 2 * (time * time * (((s *= (1.525f)) + 1) * time - s)) +
begin;
            return change / 2 * ((time -= 2) * time * (((s *= (1.525f)) + 1) * time + s) + 2)
+ begin;
        }
```

12. 指数衰减正弦曲线缓动-Elastic

1) EaseIn-缓入

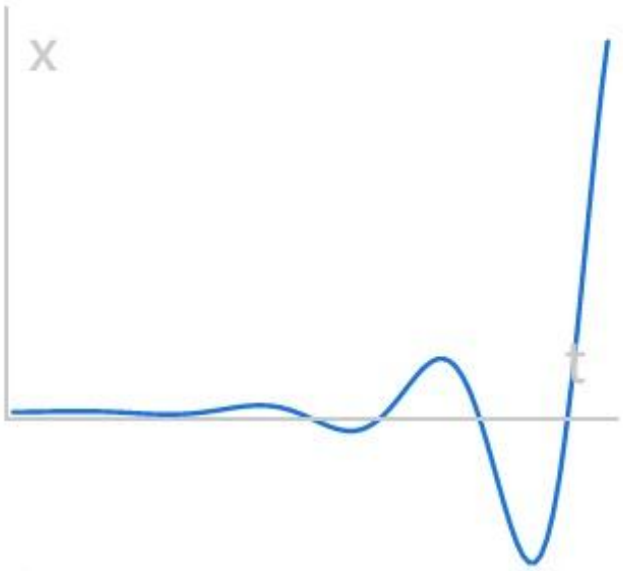


图 39. EaseInElastic

PS：

```
public static float EaseInElastic(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
        {
            float a = 0;
            float p = 0;
            time = GetTime(time, duration, type);
            if (time == 0)
                return begin;
            if ((time /= duration) == 1)
                return begin + change;
            if (p == 0)
                p = duration * 0.3f;

            float s = 0;
            if (a == 0 || a < Mathf.Abs(change))
            {
                a = change;
                s = p / 4;
            }
            else
            {
                s = p / (2 * Mathf.PI) * Mathf.Asin(change / a);
            }
            return -(a * Mathf.Pow(2, 10 * (time -= 1)) * Mathf.Sin((time * duration - s) *
(2 * Mathf.PI) / p)) + begin;
```

}

2) EaseOut-缓出

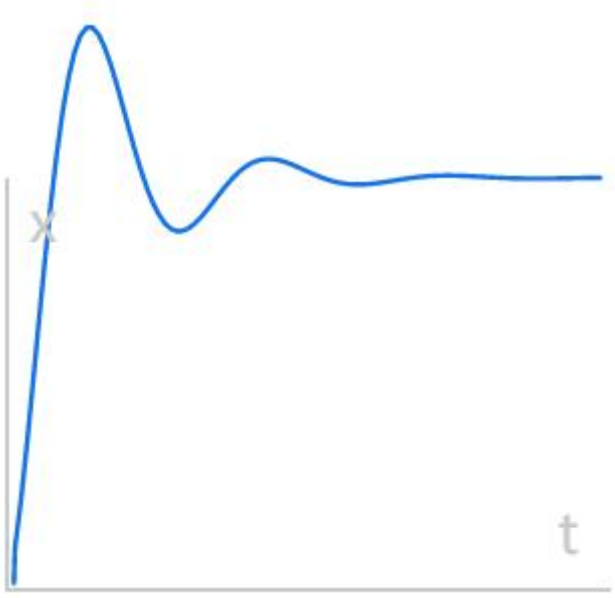


图 40. EaseOutElastic

PS：

public static float EaseOutElastic(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)

{

float a = 0;

float p = 0;

time = GetTime(time, duration, type);

if (time == 0)

return begin;

if ((time /= duration) == 1)

return begin + change;

if (p == 0)

p = duration * 0.3f;

float s = 0;

if (a == 0 || a < Mathf.Abs(change))

{

a = change;

s = p / 4;

}

else

s = p / (2 * Mathf.PI) * Mathf.Asin(change / a);

return (a * Mathf.Pow(2, -10 * time) * Mathf.Sin((time * duration - s) * (2 * Mathf.PI) / p) + change + begin);

}

3) EaseInOut-缓入缓出

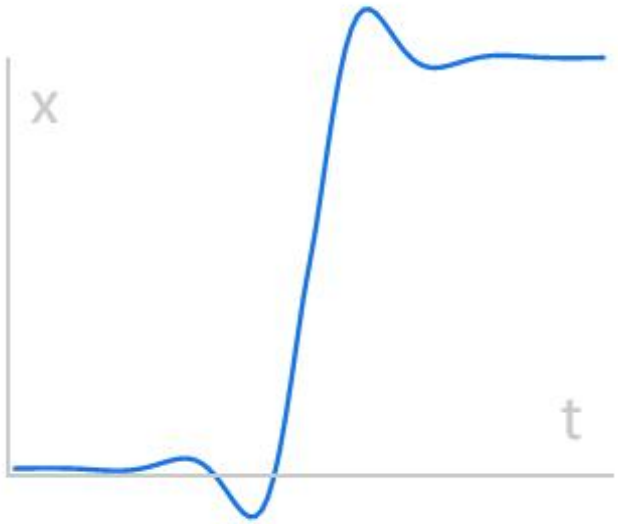


图 41. EaseInOutElastic

PS：

```
public static float EaseInOutElastic(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
        {
            float a = 0;
            float p = 0;
            time = GetTime(time, duration, type);
            if (time == 0)
                return begin;
            if ((time /= duration / 2) == 2)
                return begin + change;
            if (p == 0)
                p = duration * (0.3f * 1.5f);

            float s = 0;
            if (a == 0 || a < Mathf.Abs(change))
            {
                a = change;
                s = p / 4;
            }
            else
                s = p / (2 * Mathf.PI) * Mathf.Asin(change / a);

            if (time < 1)
                return -0.5f * (a * Mathf.Pow(2, 10 * (time -= 1)) * Mathf.Sin((time *
duration - s) * (2 * Mathf.PI) / p)) + begin;
            return a * Mathf.Pow(2, -10 * (time -= 1)) * Mathf.Sin((time * duration - s) *
(2 * Mathf.PI) / p) * 0.5f + change + begin;
```

}

13. 弹跳曲线缓动-Bounce

1) EaseIn-缓入

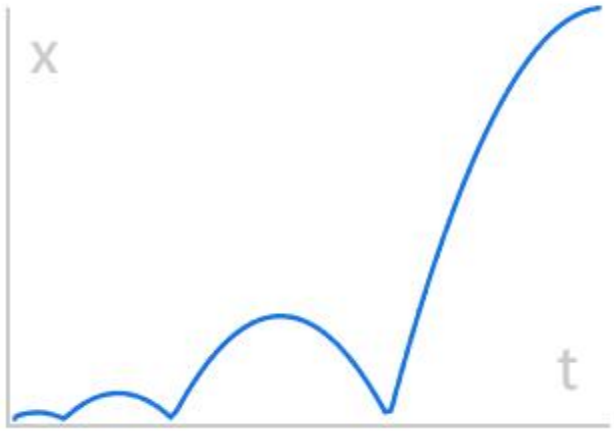


图 42. EaseInBounce

PS：

public static float EaseInBounce(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)

{

time = GetTime(time, duration, type);

//1 - easeOutBounce(1 - x);

return change - EaseOutBounce(duration - time, 0, change, duration, type) + begin;

}

2) EaseOut-缓出

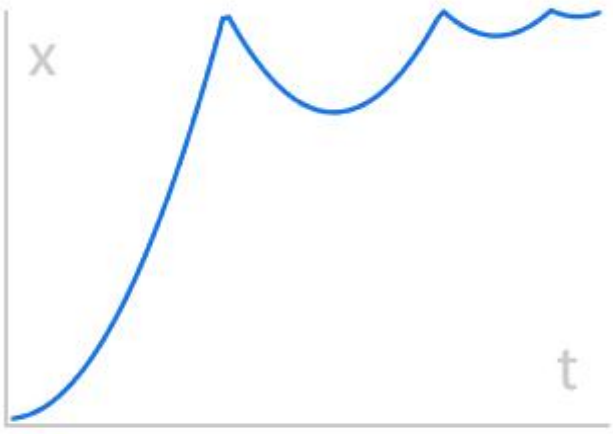


图 43. EaseOutBounce

PS：

public static float EaseOutBounce(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)

{

time = GetTime(time, duration, type);

```
                if ((time /= duration) < (1 / 2.75f))
                {
                    return change * (7.5625f * time * time) + begin;
                }
                else if (time < (2 / 2.75f))
                {
                    return change * (7.5625f * (time -= (1.5f / 2.75f)) * time + 0.75f) +
begin;
                }
                else if (time < (2.5 / 2.75f))
                {
                    return change * (7.5625f * (time -= (2.25f / 2.75f)) * time + 0.9375f) +
begin;
                }
                else
                {
                    return change * (7.5625f * (time -= (2.625f / 2.75f)) * time + 0.984375f)
+ begin;
                }

            }
```

3) EaseInOut-缓入缓出

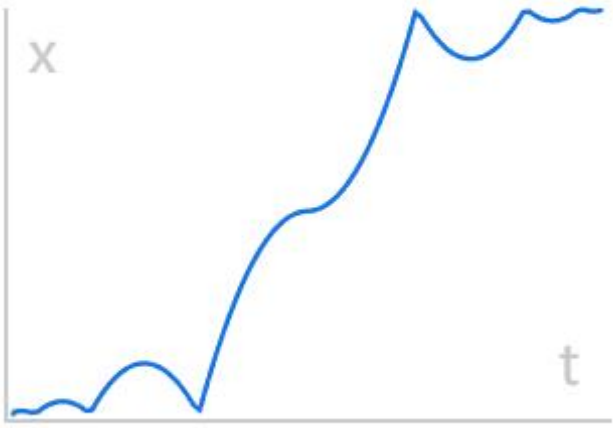


图 44. EaseInOutBounce

PS：

```
    public static float EaseInOutBounce(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
            {
                time = GetTime(time, duration, type);
                //x < 0.5 ? (1 - easeOutBounce(1 - 2 * x)) / 2 : (1 + easeOutBounce(2 * x - 1))
/ 2;
                if (time < duration / 2)
                    return EaseInBounce(time * 2, 0, change, duration, type) * 0.5f + begin;
```

```
                else
                    return EaseOutBounce(time * 2 - duration, 0, change, duration) * 0.5f + change * 0.5f + begin;
            }
```

14. 缓动函数总代码

PS：

```
namespace EaseFunction
{
    public enum LoopType
    {
        Clamp = 0,
        Loop = 1,
    }

    public enum EaseType
    {
        EaseInSine = 0,
        EaseOutSine = 1,
        EaseInOutSine = 2,
        EaseInQuad = 3,
        EaseOutQuad = 4,
        EaseInOutQuad = 5,
        EaseInCubic = 6,
        EaseOutCubic = 7,
        EaseInOutCubic = 8,
        EaseInQuart = 9,
        EaseOutQuart = 10,
        EaseInOutQuart = 11,
        EaseInQuint = 12,
        EaseOutQuint = 13,
        EaseInOutQuint = 14,
        EaseInExpo = 15,
        EaseOutExpo = 16,
        EaseInOutExpo = 17,
        EaseInCirc = 18,
        EaseOutCirc = 19,
        EaseInOutCirc = 20,
        EaseInBack = 21,
        EaseOutBack = 22,
        EaseInOutBack = 23,
        EaseInElastic = 24,
        EaseOutElastic = 25,
        EaseInOutElastic = 26,
        EaseInBounce = 27,
```

```
        EaseOutBounce = 28,
        EaseInOutBounce = 29,
        Linear = 30,
        Custom = 31,
    }

    public static class CurveEase
    {
        /// <summary>
        /// 获取缓动曲线时间
        /// </summary>
        /// <param name="time">当前时间</param>
        /// <param name="duration">总的曲线执行时间</param>
        /// <returns></returns>
        public static float GetTime(float time, float duration , LoopType type =
LoopType.Clamp)
        {
            switch (type)
            {
                case LoopType.Clamp:
                    return time > duration ? duration : time;
                case LoopType.Loop:
                    return time % duration;
                default:
                    return time;
            }
        }
        /// <summary>
        /// 线性缓动，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是 x 轴
长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns>返回值是点 a 对应的 y 轴坐标</returns>
        public static float Linear(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            return change * (time / duration) + begin;
        }
        #region Sine-正弦缓动
```

```
        /// <summary>
        ///  正弦缓入，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是 x 轴长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns>change - cos(time / duration * PI/2) + begin</returns>
        public static float EaseInSine(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //1-cos(x * PI/2)
            return change - Mathf.Cos((time / duration) * (Mathf.PI/2)) + begin;
        }
        /// <summary>
        ///  正弦缓出
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns>change * sin(time / duration * PI/2) + begin</returns>
        public static float EaseOutSine(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //sin(x * PI/2)
            return change * Mathf.Sin(time * (Mathf.PI / 2)) + begin;
        }
        /// <summary>
        ///  正弦缓入缓出
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns>- change * (Mathf.Cos(time / duration * Mathf.PI) - 1) / 2 + begin;</returns>
        public static float EaseInOutSine(float time, float begin, float change, float duration,
```

```
LoopType type = LoopType.Clamp)
            {
                time = GetTime(time, duration, type);
                // -(cos(PI * x) - 1) / 2
                return - change * (Mathf.Cos(time * Mathf.PI) - 1) / 2 + begin;
            }
            #endregion

            #region Quadratic-二次方缓动
            /// <summary>
            ///  二次方缓入，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是 x
轴长度,start 和 change 分别是 y 轴起点和长度
            /// </summary>
            /// <param name="time">动画执行到当前帧所经过的时间</param>
            /// <param name="begin">起始值</param>
            /// <param name="change">总移动距离</param>
            /// <param name="duration">动画的总时间</param>
            /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
            /// <returns>change * (time /= duration * time) + begin</returns>
            public static float EaseInQuad(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
            {
                time = GetTime(time, duration, type);
                // x^2
                return change * (time /= duration * time) + begin;
            }
            /// <summary>
            ///  二次方缓出
            /// </summary>
            /// <param name="time">动画执行到当前帧所经过的时间</param>
            /// <param name="begin">起始值</param>
            /// <param name="change">总移动距离</param>
            /// <param name="duration">动画的总时间</param>
            /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
            /// <returns> -change * (time /= duration) * (time - 2) + begin</returns>
            public static float EaseOutQuad(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
            {
                time = GetTime(time, duration, type);
                // 1 - (1 - x)*(1 - x)
                return -change * (time /= duration) * (time - 2) + begin;
            }
            /// <summary>
            ///  二次方缓入缓出
```

```
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns></returns>
        public static float EaseInOutQuad(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //time < 0.5 ? 2 * time * time : 1 - pow(-2 * time + 2, 2) / 2;
            if ((time /= duration / 2) < 1)
            {
                return change / 2 * time * time + begin;
            }
            return -change / 2 * ((--time) * (time - 2) - 1) + begin;
        }
        #endregion

        #region Cubic-三次方缓动
        /// <summary>
        /// 三次方缓入，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是 x 轴长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns>change * (time /= duration) * time * time + begin</returns>
        public static float EaseInCubic(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //x * x * x
            return change * (time /= duration) * time * time + begin;
        }
        /// <summary>
        /// 三次方缓出，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是 x 轴长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
```

```
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns>change * ((time = time / duration - 1) * time * time + 1) +
begin</returns>
        public static float EaseOutCubic(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //1 - pow(1 - x, 3);
            return change * ((time = time / duration - 1) * time * time + 1) + begin;
        }
        /// <summary>
        /// 三次方缓入缓出，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration
是 x 轴长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns></returns>
        public static float EaseInOutCubic(float time, float begin, float change, float
duration, LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //x < 0.5 ? 4 * x * x * x : 1 - pow(-2 * x + 2, 3) / 2;
            if ((time /= duration / 2) < 1)
                return change / 2 * time * time * time + begin;
            return change / 2 * ((time -= 2) * time * time + 2) + begin;
        }
        #endregion

        #region Quartic-四次方缓动
        /// <summary>
        /// 四次方缓入，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是 x
轴长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns>change * (time /= duration) * time * time * time + begin</returns>
```

```
        public static float EaseInQuart(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //x * x * x * x
            return change * (time /= duration) * time * time * time + begin;
        }
        /// <summary>
        /// 四次方缓出，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是 x
轴长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns>-change * ((time = time / duration - 1) * time * time * time - 1) +
begin</returns>
        public static float EaseOutQuart(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //1 - pow(1 - x, 4);
            return -change * ((time = time / duration - 1) * time * time * time - 1) + begin;
        }
        /// <summary>
        /// 四次方缓入缓出，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration
是 x 轴长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns></returns>
        public static float EaseInOutQuart(float time, float begin, float change, float
duration, LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            // x < 0.5 ? 8 * x * x * x * x : 1 - pow(-2 * x + 2, 4) / 2
            if ((time /= duration / 2) < 1)
                return change / 2 * time * time * time * time + begin;
            return -change / 2 * ((time -= 2) * time * time * time - 2) + begin;
        }
```

```
        #endregion

        #region Quintic-五次方缓动
        /// <summary>
        /// 五次方缓入，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是 x 轴长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns>change * (time /= duration) * time * time * time + begin</returns>
        public static float EaseInQuint(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //x^5
            return change * (time /= duration) * time * time * time * time + begin;
        }
        /// <summary>
        /// 五次方缓出，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是 x 轴长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns>-change * ((time = time / duration - 1) * time * time * time - 1) + begin</returns>
        public static float EaseOutQuint(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //1 - pow(1 - x, 5);
            return change * ((time = time / duration - 1) * time * time * time * time - 1) + begin;
        }
        /// <summary>
        /// 五次方缓入缓出，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是 x 轴长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
```

```
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns></returns>
        public static float EaseInOutQuint(float time, float begin, float change, float
duration, LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            // x < 0.5 ? 16 * x * x * x * x * x : 1 - pow(-2 * x + 2, 5) / 2
            if ((time /= duration / 2) < 1)
                return change / 2 * time * time * time * time * time + begin;
            return change / 2 * ((time -= 2) * time * time * time * time + 2) + begin;
        }
        #endregion

        #region Exponential-指数缓动
        /// <summary>
        /// 指数缓入，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是 x 轴
长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns>(time == 0) ? begin : change * Math.pow(2, 10 * (time / duration - 1))
+ begin</returns>
        public static float EaseInExpo(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //x === 0 ? 0 : pow(2, 10 * x - 10);
            if (time == 0)
            {
                return begin;
            }
            else
            {
                return change * Mathf.Pow(2, 10 * (time / duration - 1)) + begin;
            }
        }
        /// <summary>
        /// 指数缓出，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是 x 轴
```

```
长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns>(time == duration) ? begin + change : change * (-Math.pow(2, -10 *
time / duration) + 1) + begin</returns>
        public static float EaseOutExpo(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //x === 1 ? 1 : 1 - pow(2, -10 * x)
            if (time == duration)
            {
                return begin + change;
            }
            else
            {
                return change * (-Mathf.Pow(2, 10 * time / duration) + 1) + begin;
            }
        }
        /// <summary>
        /// 指数缓入缓出，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是
x 轴长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns></returns>
        public static float EaseInOutExpo(float time, float begin, float change, float
duration, LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //当时间为 0 时，返回的 Y 值就是设定的开始位置
            if (time == 0)
                return begin;
            //当时间为动画总时间，返回的 Y 值就是起始值加上 Y 轴的总长度
            if (time == duration)
                return begin + change;
            if ((time /= duration / 2) < 1)
```

```
                return change / 2 * Mathf.Pow(2, 10 * (time - 1)) + begin;
            return change / 2 * (-Mathf.Pow(2, -10 * --time) + 2) + begin;
        }
        #endregion

        #region Circular-圆形曲线缓动
        /// <summary>
        /// 指数缓入，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是 x 轴长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns>-change * (Mathf.Sqrt(1 - (time /= duration) * time) - 1) + begin</returns>
        public static float EaseInCirc(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //1 - sqrt(1 - pow(x, 2))
            return -change * (Mathf.Sqrt(1 - (time /= duration) * time) - 1) + begin;
        }
        /// <summary>
        /// 指数缓出，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是 x 轴长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns>change * Mathf.Sqrt(1 - (time /= duration - 1) * time) + begin</returns>
        public static float EaseOutCirc(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //sqrt(1 - pow(x - 1, 2));
            return change * Mathf.Sqrt(1 - (time /= duration - 1) * time) + begin;
        }
        /// <summary>
        /// 指数缓入缓出，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是
```

```
x 轴长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns></returns>
        public static float EaseInOutCirc(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            //x < 0.5    ? (1 - sqrt(1 - pow(2 * x, 2))) / 2    : (sqrt(1 - pow(-2 * x + 2, 2)) + 1)
/ 2;
            if ((time /= duration / 2) < 1)
                return -change / 2 * (Mathf.Sqrt(1 - time * time) - 1) + begin;
            return change / 2 * (Mathf.Sqrt(1 - (time -= 2) * time) + 1) + begin;
        }
        #endregion

        #region Back-向后缓动
        /// <summary>
        /// 指数缓入，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是 x 轴
长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns>change - EaseOutBounce(duration - time, 0, change, duration, type) +
begin</returns>
        public static float EaseInBack(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
        {
            float s = 0;
            time = GetTime(time, duration, type);
            if (s == 0)
                s = 1.70158f;
            return change * (time /= duration) * time * ((s + 1) * time - s) + begin;
        }
        /// <summary>
        /// 指数缓出，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是 x 轴
长度,start 和 change 分别是 y 轴起点和长度
```

```
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns></returns>
        public static float EaseOutBack(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
        {
            float s = 0;
            time = GetTime(time, duration, type);
            if (s == 0)
                s = 1.70158f;
            return change * ((time = time / duration - 1) * time * ((s + 1) * time + s) + 1) +
begin;
        }
        /// <summary>
        /// 指数缓入缓出，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是
x 轴长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns></returns>
        public static float EaseInOutBack(float time, float begin, float change, float
duration, LoopType type = LoopType.Clamp)
        {
            float s = 0;
            time = GetTime(time, duration, type);
            if (s == 0)
                s = 1.70158f;
            if ((time /= duration / 2) < 1)
                return change / 2 * (time * time * (((s *= (1.525f)) + 1) * time - s)) +
begin;
            return change / 2 * ((time -= 2) * time * (((s *= (1.525f)) + 1) * time + s) + 2)
+ begin;
        }
        #endregion

        #region Elastic-指数衰减正弦曲线缓动
        /// <summary>
```

```
        /// 指数缓入，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是 x 轴长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns>change - EaseOutBounce(duration - time, 0, change, duration, type) + begin</returns>
        public static float EaseInElastic(float time, float begin, float change, float duration, LoopType type = LoopType.Clamp)
        {
            float a = 0;
            float p = 0;
            time = GetTime(time, duration, type);
            if (time == 0)
                return begin;
            if ((time /= duration) == 1)
                return begin + change;
            if (p == 0)
                p = duration * 0.3f;

            float s = 0;
            if (a == 0 || a < Mathf.Abs(change))
            {
                a = change;
                s = p / 4;
            }
            else
            {
                s = p / (2 * Mathf.PI) * Mathf.Asin(change / a);
            }
            return -(a * Mathf.Pow(2, 10 * (time -= 1)) * Mathf.Sin((time * duration - s) * (2 * Mathf.PI) / p)) + begin;
        }
        /// <summary>
        /// 指数缓出，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是 x 轴长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
```

```
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns></returns>
        public static float EaseOutElastic(float time, float begin, float change, float
duration, LoopType type = LoopType.Clamp)
        {
            float a = 0;
            float p = 0;
            time = GetTime(time, duration, type);
            if (time == 0)
                return begin;
            if ((time /= duration) == 1)
                return begin + change;
            if (p == 0)
                p = duration * 0.3f;

            float s = 0;
            if (a == 0 || a < Mathf.Abs(change))
            {
                a = change;
                s = p / 4;
            }
            else
                s = p / (2 * Mathf.PI) * Mathf.Asin(change / a);
            return (a * Mathf.Pow(2, -10 * time) * Mathf.Sin((time * duration - s) * (2 *
Mathf.PI) / p) + change + begin);
        }
        /// <summary>
        /// 指数缓入缓出，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是
x 轴长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns></returns>
        public static float EaseInOutElastic(float time, float begin, float change, float
duration, LoopType type = LoopType.Clamp)
        {
            float a = 0;
            float p = 0;
            time = GetTime(time, duration, type);
            if (time == 0)
                return begin;
```

```
                if ((time /= duration / 2) == 2)
                    return begin + change;
                if (p == 0)
                    p = duration * (0.3f * 1.5f);

                float s = 0;
                if (a == 0 || a < Mathf.Abs(change))
                {
                    a = change;
                    s = p / 4;
                }
                else
                    s = p / (2 * Mathf.PI) * Mathf.Asin(change / a);

                if (time < 1)
                    return -0.5f * (a * Mathf.Pow(2, 10 * (time -= 1)) * Mathf.Sin((time *
duration - s) * (2 * Mathf.PI) / p)) + begin;
                return a * Mathf.Pow(2, -10 * (time -= 1)) * Mathf.Sin((time * duration - s) *
(2 * Mathf.PI) / p) * 0.5f + change + begin;
            }
            #endregion

            #region Bounce-弹跳曲线缓动
            /// <summary>
            /// 指数缓入，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是 x 轴
长度,start 和 change 分别是 y 轴起点和长度
            /// </summary>
            /// <param name="time">动画执行到当前帧所经过的时间</param>
            /// <param name="begin">起始值</param>
            /// <param name="change">总移动距离</param>
            /// <param name="duration">动画的总时间</param>
            /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
            /// <returns>change - EaseOutBounce(duration - time, 0, change, duration, type) +
begin</returns>
            public static float EaseInBounce(float time, float begin, float change, float duration,
LoopType type = LoopType.Clamp)
            {
                time = GetTime(time, duration, type);
                //1 - easeOutBounce(1 - x);
                return change - EaseOutBounce(duration - time, 0, change, duration, type) +
begin;
            }
            /// <summary>
            /// 指数缓出，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是 x 轴
```

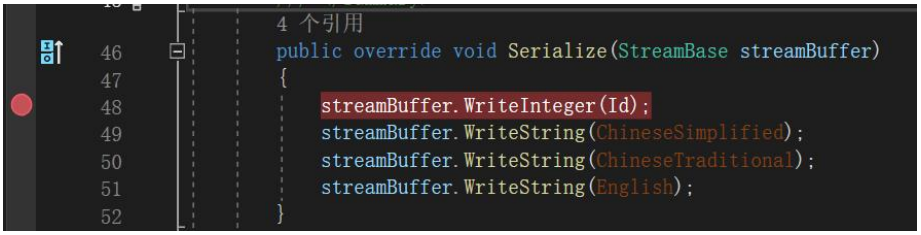
```
长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns></returns>
        public static float EaseOutBounce(float time, float begin, float change, float
duration, LoopType type = LoopType.Clamp)
        {
            time = GetTime(time, duration, type);
            if ((time /= duration) < (1 / 2.75f))
            {
                return change * (7.5625f * time * time) + begin;
            }
            else if (time < (2 / 2.75f))
            {
                return change * (7.5625f * (time -= (1.5f / 2.75f)) * time + 0.75f) +
begin;
            }
            else if (time < (2.5 / 2.75f))
            {
                return change * (7.5625f * (time -= (2.25f / 2.75f)) * time + 0.9375f) +
begin;
            }
            else
            {
                return change * (7.5625f * (time -= (2.625f / 2.75f)) * time + 0.984375f)
+ begin;
            }

        }
        /// <summary>
        /// 指数缓入缓出，对应到坐标轴上, time 是 x 轴上某点(记为点 a), duration 是
x 轴长度,start 和 change 分别是 y 轴起点和长度
        /// </summary>
        /// <param name="time">动画执行到当前帧所经过的时间</param>
        /// <param name="begin">起始值</param>
        /// <param name="change">总移动距离</param>
        /// <param name="duration">动画的总时间</param>
        /// <param name="type">动画曲线的循环类型，默认是 Clamp</param>
        /// <returns></returns>
        public static float EaseInOutBounce(float time, float begin, float change, float
```

```
duration, LoopType type = LoopType.Clamp)
                {
                    time = GetTime(time, duration, type);
                    //x < 0.5 ? (1 - easeOutBounce(1 - 2 * x)) / 2 : (1 + easeOutBounce(2 * x - 1))
/ 2;
                    if (time < duration / 2)
                        return EaseInBounce(time * 2, 0, change, duration, type) * 0.5f + begin;
                    else
                        return EaseOutBounce(time * 2 - duration, 0, change, duration) * 0.5f +
change * 0.5f + begin;
                }
                #endregion
            }
        }
```

六. Visual Studio 断点

1. 普通断点

设置普通断点的方法很简单，就是在代码行的左边栏灰色区域点击或者把光标放在某代码行，按下 F9。



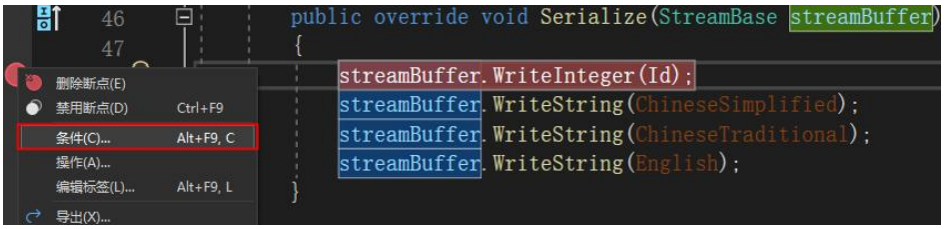
如上图所示，此时左边栏出现的红色圆点就代表了一个断点。

按下 F5， Debugger 就会在此处停下来。

2. 条件断点

不像是普通的断点，程序每次执行都会停下来。条件断点的意义是，只有在条件被满足时，Debugger 才会在此处停下来。

设置条件断点一般的方法是在红点处右键选择条件或者把光标放在红点处，等待齿轮图标出现并点击。

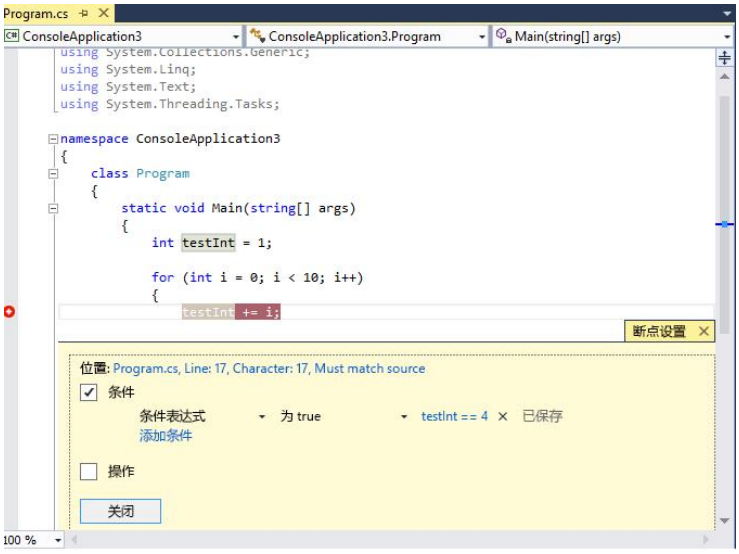


条件断点设置窗口如下。

总共有三种类型的条件断点：
条件表达式 （Conditional Expression）
命中次数 （Hit Count）
筛选器 （Filter）

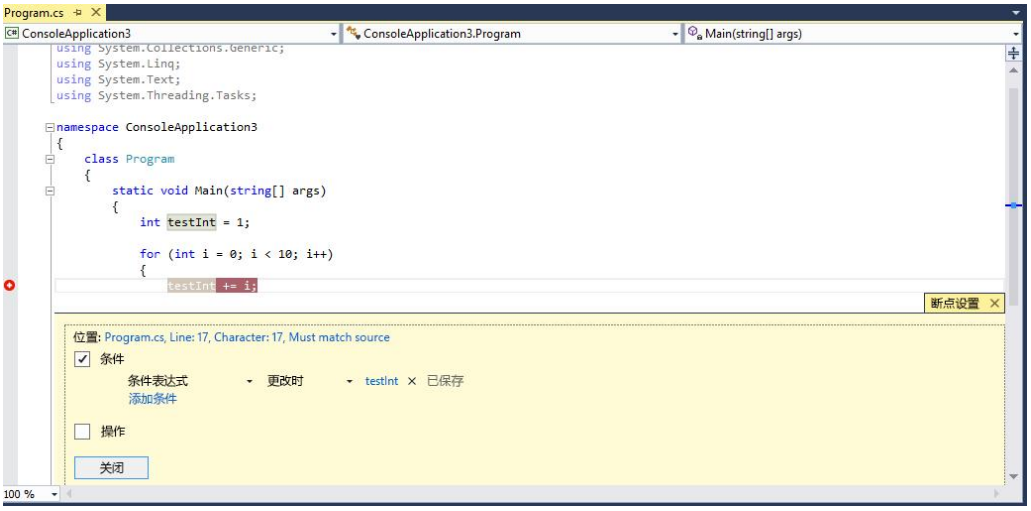
1) 条件表达式

条件表达式也有两种类型：
为 ture （Is ture）
如下图示，此处设置的条件为： 当 testInt == 4 is ture 时命中断点。



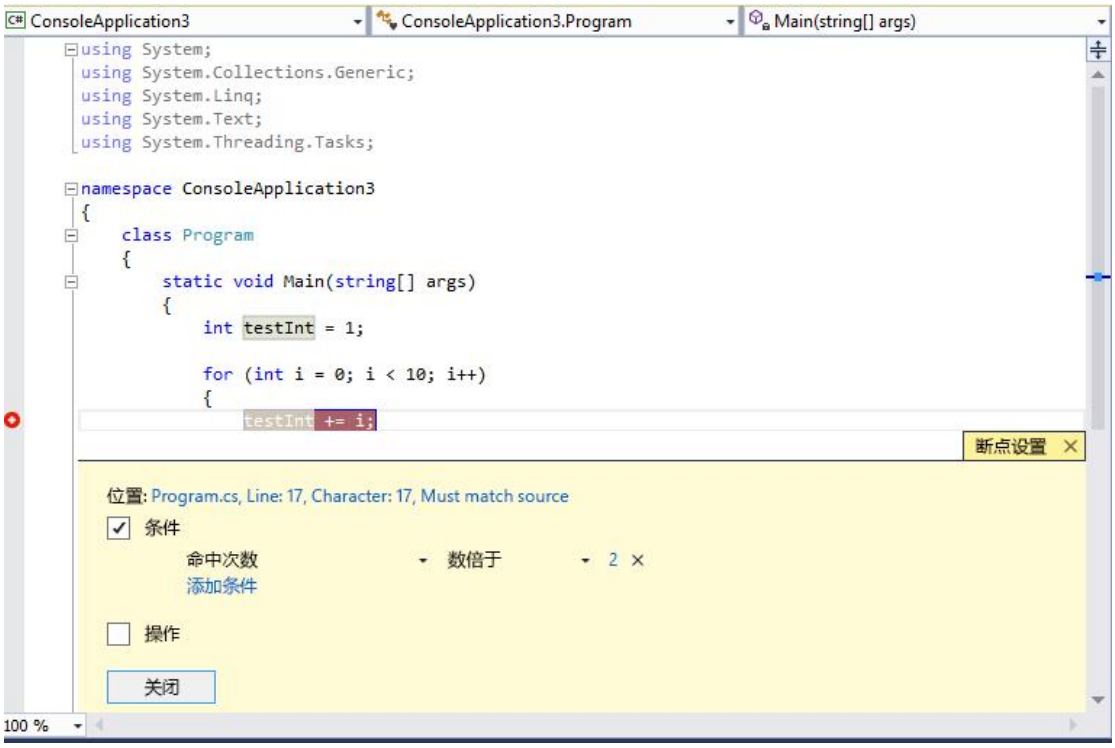
2)更改时

如下图示，此处设置的条件为： 当 testInt 被更改的时候命中断点。



3)命中次数

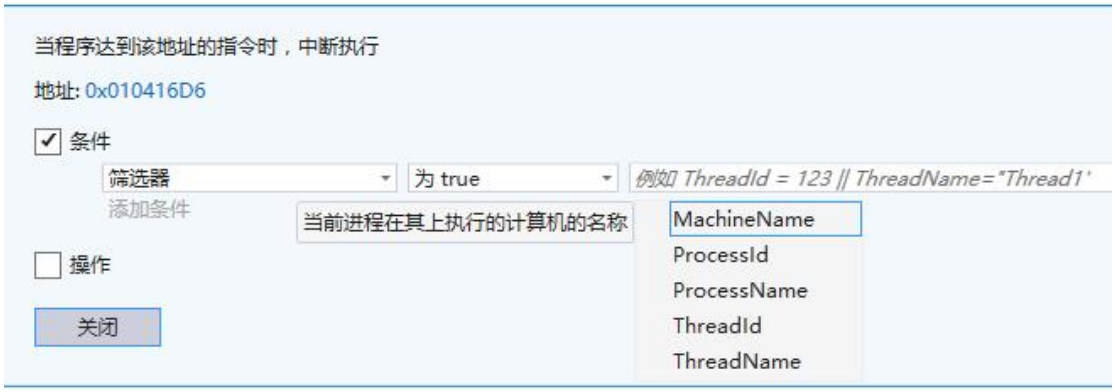
命中次数在 Debug 循环语句时非常有用。
比如你怀疑程序出错点发生在 N 次循环之后或者某些次循环，那么可以设置循环语句内的命中次数，让它在某些次或者某次循环停下来，而不是每次都停下来。
如下图，设置的条件为：让它每隔一次循环断下来。



4)筛选器

筛选器是用来限制断点命中时所在的设备、进程和线程。

比如，在一个多线程的程序中，我们不必手动 GetThreadId，可以通过筛选器，设置让它只在某线程中执行到此处时停下来。



筛选器表达式用法为：

MachineName = "name"

ProcessId = value

ProcessName = "name"

ThreadId = value

ThreadName = "name"

多个表达式之间可以使用运算符 & (AND), || (OR), ! (NOT) 连接 。

3. 追踪断点

在断点设置窗口，除了条件，我们还可以看到操作（Action）选项框。

操作（Action）的意义是设置追踪点（Tracepoint）。

Tracepoint 相当于是一种临时的有 Trace 功能的断点，它会把消息打印到 Output 窗口。

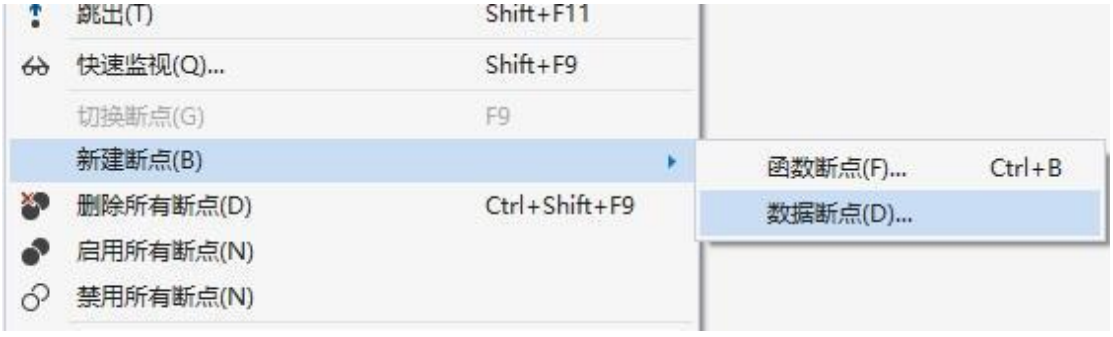
勾选后面的继续执行（Continue Execution），代表 Tracepoint 命中时，Debugger 不会停下来，否则将会在此处停下来。两种情况下，消息都会打印出来。

可以使用下面的关键字作为消息。

$ADDRESS	Current instruction
$CALLER	Calling function name
$CALLSTACK	Call stack
$FUNCTION	Current function name
$PID	Process id
$PNAME	Process name
$TID	Thread id
$TNAME	Thread name
$TICK	
$TNAME	

4. 函数断点

在 Debug 菜单下，点击新建断点，可以新建两种类型的断点：函数断点（Function Breakpoint）和数据断点（Data Breakpoint）。



函数断点是通过函数名设置断点，当程序执行到该函数的时候断点断下来。

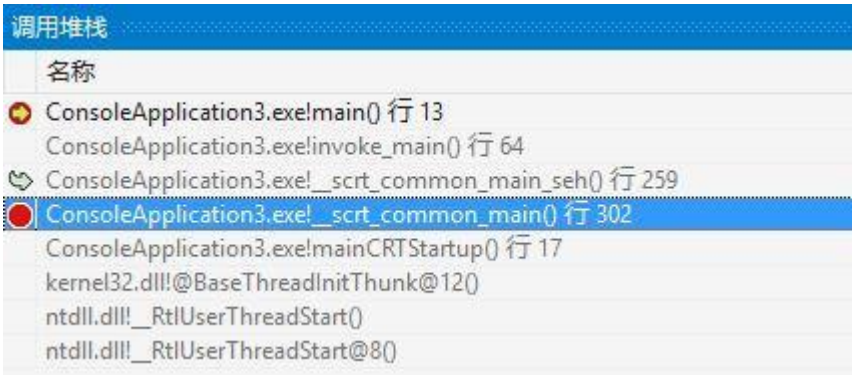
5. 数据断点

紧接上面，这里讲下数据断点。

数据断点的意义是，让程序在当某处地址指定字节发生改变的时候中断下来。它只有在 Break Mode 下才可以设置。

地址栏可以用具体的内存地址，也可以用表达式来代表内存地址。例如，可以使用 &avar，让程序在当变量 avar 内容发生改变的时候中断下来。

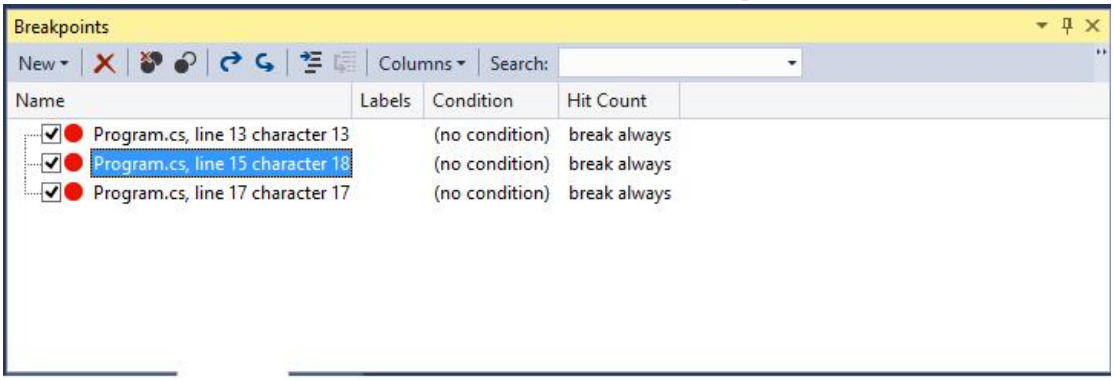
6. 在 Call Stack Window 设置断点



当我们在查看函数的调用关系的时候，想在这一系列的函数中快速设置断点，那么其实很简单：在 Call Stack 窗口，点击想要设置断点的函数所在的行，按下 F9 即可。

7. 断点管理

我们前面所设置的所有断点都会出现在断点窗口（Breakpoints Windows）。



在这里，可以对所有的断点进行管理，比如批量启用、禁用、删除等，还可以在程序运行过程中，查看断点的命中状态。

当工程非常大，Debug 时设置了很多的断点的时候，它会非常有用。