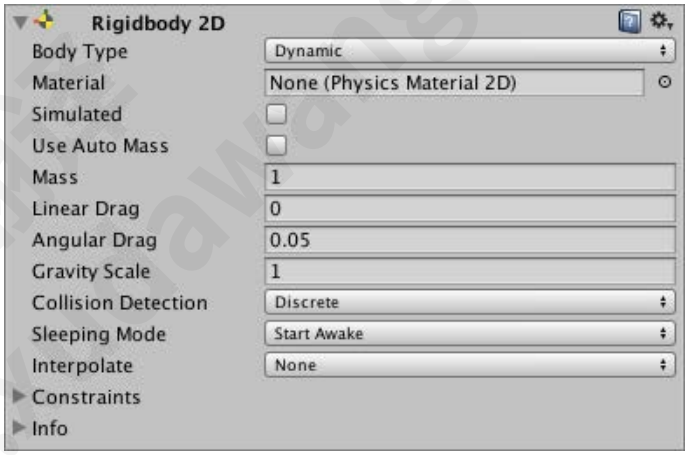
Rigidbody 2D
刚体2D 组件

刚体2D(Rigidbody 2D)组件将游戏物体(GameObject)置于物理引擎的控制下。在标准 Rigidbody 组件中，有许多我们熟悉的概念，这些概念都将在Rigidbody 2D 组件中出现；不同之处在于，在Rigidbody 2D中，物体只能在XY平面(XY轴)内移动，并且只能在垂直于该平面的轴(Z轴)上旋转。



上图是 Rigidbody 2D 组件↑
这个组件在Unity编辑器中显示的方式，可能与上图有所不同，这具体取决于您为 Body Type 字段设置什么值。 请参阅 Body Type，了解更多信息。

Rigidbody 2D如何工作

通常，Unity Editor的Transform组件定义了GameObject（以及它所有的子GameObject）如何在场景中的位置、旋转和缩放。 当Transform被更改时，它会更新其他组件，而这些组件可能会更新它们显示的位置或碰撞对象的位置。
2D物理引擎能够移动碰撞体并使它们相互交互，所以物理引擎需要一种方法将碰撞体的这种移动传回给Transform组件。 这种移动和与碰撞器的连接，便是Rigidbody 2D组件的作用所在。

Rigidbody 2D组件会覆盖Transform 组件，这会让Rigidbody 2D组件来定义的物体的位置/旋转。 请注意，虽然您仍然可以无视Rigidbody 2D组件，通过自己修改Transform组件来改变的物体的位置和旋转，（因为Unity会公开所有组件的所有属性，所以你想修改

这里应该是说：

如果你在一个物体上附加了Rigidbody 2D组件，那么你就不要使用Transform组件去更改此物体的位置和旋转了。
请使用Rigidbody 2D组件去更改，这样可以避免很多错误。

Transform组件的值，我们无法阻止你），但这样做会导致很多问题，比如，GameObjects相互传递，或相互传递以及不可预知的移动等问题。

添加到此GameObject(被附加了Rigidbody 2D组件的GameObject)和 它的子GameObject上的任何Collider 2D组件，都会被隐式的连接到该Rigidbody 2D组件上。
当Collider 2D组件连接到Rigidbody 2D组件时，Collider 2D组件会随Rigidbody 2D组件移动。
在任何时候，都不应该使用Transform组件或任何碰撞器偏移，来直接移动Collider 2D。你应该直接移动Rigidbody 2D组件， 这样做会提供最佳的性能，并确保正确的碰撞检测。
连接到同一个Rigidbody 2D组件的Collider 2D不会相互碰撞。 这意味着您可以创建一组可以有效作为单个复合对撞机的对撞机，它们都与Rigidbody 2D进行同步移动和旋转。

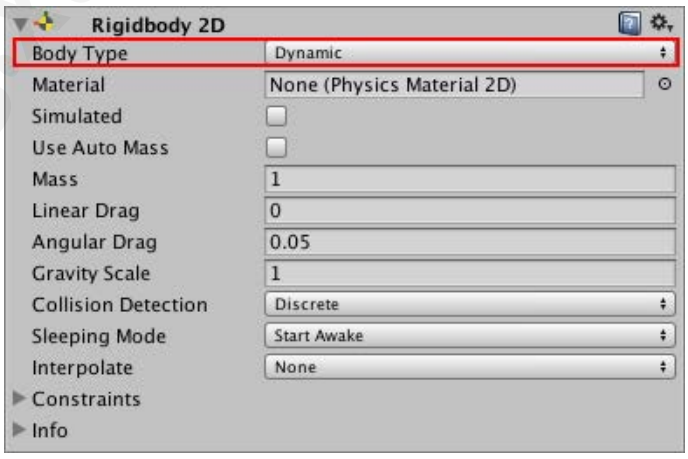
设计场景时，您可以自由使用Unity提供的Rigidbody 2D组件，并在它所在的游戏物体上附加碰撞器。
这些碰撞器会与(附加在其他的Rigidbody 2D上的)任何其他的碰撞器产生碰撞。

提示

通过添加一个**Rigidbody 2D 组件**，然后调用组件提供的API，你可以让一个Sprite(精灵)令人信服(符合物理)地进行移动。
当碰撞器组件也被附加到精灵所在的游戏物体上时，精灵会受到其他移动的游戏物体所产生的碰撞的影响。 使用Unity提供的物理学(物理学组件)可以简化许多常见的游戏机制，并能以最少的代码实现真实的行为。

Body Type 字段　身体类型 字段

Rigidbody 2D组件的顶部有一个Body Type字段。 您可以设置这个字段。而这个字段的值，会影响Rigidbody 2D组件的选项。
如果你无法理解这句话，那么你可以现在修改一下这个字段，就能看到我说的效果。



Body Type 字段有三个选项， 每个选项都有自己的行为。 任何连接到Rigidbody 2D组件的Collider 2D，都会继承Rigidbody 2D组件的Body Type。 这三个选项分别是：

- **Dynamic**　动态
- **Kinematic**　运动学
- **Static**　静态

絮大王备注：

这段的意思大概就是：
如果你要移动一个游戏物体(GameObject)，**并且**这个游戏物体身上有Collider 2D组件，或者这个游戏物体的子物体身上有Collider 2D组件。
那么，你就最好不要使用Transform组件去移动这个物体！

因为Unity的Transform组件在移动的之后，还要去更新Collider 2D组件的位置，然后进行重新的计算。

如果你使用Rigidbody 2D组件去移动这个游戏物体，这个游戏物体(包括它的子物体)身上的Collider 2D组件，都会跟着移动，而不需要进行重新的计算。这样的话，性能更高，而且碰撞检测更加准确。

如何使用Rigibody 2D组件移动游戏物体？

最常见的是使用：
rigidbody2D.position字段
rigidbody2D.rotation字段
rigidbody2D.MovePosition()方法
rigidbody2D.MoveRotation()方法
rigidbody2D.AddForce()方法

[旋转和位移都属于移动]

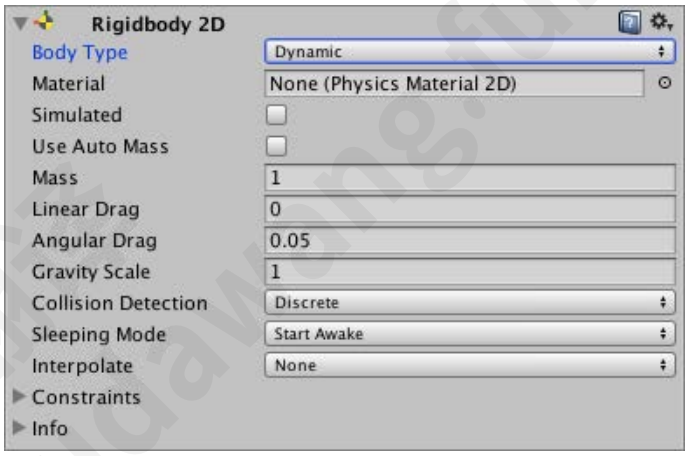
您选择的选项 会决定：

- 移动（位置和旋转）的行为
- 碰撞体的相互作用

请注意，尽管Rigidbody 2D经常被描述为用于实现互相碰撞，但其实，要实现相互碰撞的每个游戏物体身上，都应该附有Collider 2D组件。 没有Collider 2D的话，Rigidbody 2D不会相互碰撞。

在运行状态下，改变Rigidbody 2D的Body Type字段可能是一个棘手的过程。 当Body Type变化时，会立即重新计算各种与质量相关(mass-related)的内部属性，并且在GameObject的下一个FixedUpdate(固定更新)时，需要重新评估附着在Rigidbody 2D上的Collider 2D的所有现有碰撞。
这个过程的棘手程度，取决于Body Type上连接了多少接触点和Collider 2D，所以，更改Body Type可能会导致性能发生变化。

Body Type: Dynamic
Body Type 字段：动态



Dynamic(动态的) 的 Rigidbody2D 主要用于在物理模拟下进行移动。它具有Rigidbody2D组件可用的全部属性，如mass(质量)和drag(阻力)。*Dynamic* 会受重力和力的影响。 Dynamic 的 Body Type(身体类型)将与其他所有的身体类型相撞。并且Dynamic是所有身体类型中，交互性最高的。
Dynamic 是Rigidbody 2D组件的默认身体类型，并且它是最常见的身体类型，经常用于需要进行移动的物体上。 它也是耗费性能最高的身体类型 (开销最大的)，因为它具有动态性和与周围一切的交互性。 该身体类型可以自定义所有的Rigidbody 2D属性。

不要使用Transform组件来更改 **Dynamic** Rigidbody2D(动态刚体)的位置或旋转！ 如果你这样做，它会根据其速度重新定位**Dynamic** Rigidbody2D(动态刚体)。你可以直接通过脚本，用施加力的方法来改变它，或者通过碰撞和重力间接地改变它。

属性:	功能描述：
Body Type	这个值用于，设置RigidBody 2D的组件的模式。以便您可以选择使用哪种运动（位置和旋转）行为，以及选择使用哪种Collider 2D交互。 选项有: **Dynamic**, **Kinematic**, **Static**
Material	使用它可以为所有连接到特定父Rigidbody 2D的

絮大王备注：
也就是说，当你用Dynamic模式的时候，不要使用Transform组件来移动物体！
请使用Rigidbody 2D组件去移动物体。
也可以使用施加力、或者发生碰撞，或者受重力影响的方式去移动物体。

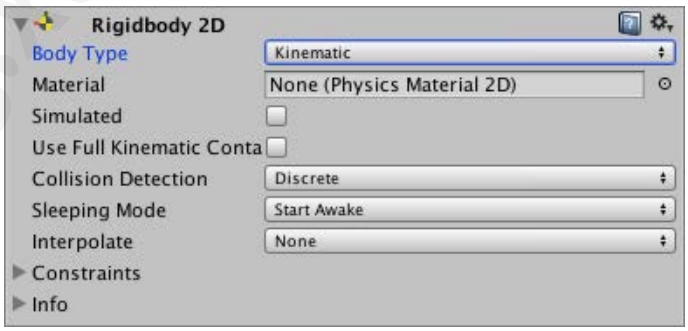
	Collider 2D指定一个通用材质。 **注意:** 如果你为一个Collider 2D组件，在它的Material 字段中指定了材质，那么这个Collider 2D组件，会使用你指定的这个 材质。 如果Rigidbody2D组件或Collider2D组件中没有指定材质，则他们会使用默认选项中的材质。 默认选项的材质为 **None(无) (Physics Material 2D)**。 如果你想修改默认材质，您可以在 Physics 2D Settings 窗口中设置它。 Collider 2D使用以下优先顺序(从高到低)确定要使用的材质(**Material**)： 1. Collider 2D组件本身指定的Physics Material 2D。 2. 在Rigidbody 2D上指定的Physics Material 2D。 3. 在物理2D设置 (Physics 2D Settings)中指定的Physics Material 2D默认材质。 **提示**：使用第3种可以确保所有连接到相同 **Static** Body Type 的Rigidbody 2D都可以使用相同的材质。
Simulated	如果您希望Rigidbody 2D组件和任何附加的Collider 2D和Joint 2D在运行期间与物理模拟进行交互，请启用**Simulated**（勾选这个方框(让这个值为true)）。 如果禁用了该选项（该框未选中），则这些组件不会与物理模拟进行交互。 有关更多详细信息，请参阅下文的 Rigidbody 2D properties: Simulated 部分。此框在默认情况下被选中。
Use Auto Mass	如果您想让Rigidbody 2D组件自动检测Collider 2D所在的GameObject的质量(Mass)，请选中该框(设置值为true)。
Mass	这个值将定义Rigidbody 2D的质量。 如果您勾选了Use Auto Mass字段，则Mass字段将变灰(不可选)。
Linear Drag	影响位置移动的阻力系数。(位移阻力)
Angular Drag	影响旋转运动的阻力系数。(旋转阻力)
Gravity Scale	定义GameObject受重力影响的程度。(重力大小)
Collision Detection	定义如何检测碰撞2D之间的碰撞。 有两个选项：**Discrete**、**Continuous**
Discrete	将 **Collision Detection** (碰撞检测的方式)设置为 **Discrete *(离散的)*** 时，会产生一个问题： 如果物体移动速度足够快，则在physics update(物理更新)期间，具有Rigidbody2D和Collider2D的GameObjects可以重叠或通过彼此。 碰撞接触只在新的位置产生。
Continuous	将 **Collision Detection** (碰撞检测的方式)设置为 **Continuous*(连续的)*** 时，带有Rigidbody2D和Collider2D的GameObjects，在physics update(物理更新)期间不会相互穿透。 相反，Unity会计算任何Collider 2D的第一个影响点，并将GameObject移动到那里。

絮大王备注：

Discrete(离散的)简单的意思就是：如果一个物体的移动速度太快，就会穿过另一个物体，而不会发生应该有的体积碰撞。
比如一个人，如果跑动的速度过快，就会出现穿过墙壁的情况。
如果你不希望这种情况发生，你应该选择**Continuous(连续的)**，这样的话，检测的频率会提高，就更容易在移动速度很快的时候，检测到两个物体之间的体积碰撞。

	请注意，这需要比**Discrete** 选项更多的CPU时间。
Sleeping Mode	这里可以自定义GameObject在休眠时如何“休眠(Sleeping)”。休眠可以节省处理器时间。
Never Sleep	*永不休眠* （应该尽可能的避免这种情况，因为它会影响系统资源）。
Start Awake	*开始时休眠* GameObject最初是清醒的。
Start Asleep	*开始时睡着* GameObject最初是睡着的，但可以被碰撞唤醒。
Interpolate	*插值*：定义如何在物理更新(physics update)之间插入GameObject的运动。 （运动趋于不稳定时很有用）
None	*无平滑* (移动不会有平滑的过渡(不应用插值))
Interpolate	*根据上一帧 进行平滑的过渡* 根据前一帧中此GameObject的位置进行平滑移动。
Extrapolate	*根据下一帧 进行平滑的过渡* 根据下一帧中此GameObject的位置的预估值来平滑移动。
Constraints	定义对Rigidbody 2D运动的限制。 (位移限制 和 旋转限制)
Freeze Position	可以选择停止Rigidbody2D在世界坐标X轴 和 Y轴上的移动。
Freeze Rotation	可以选择停止Rigidbody2D在世界坐标Z轴上的旋转。

Body Type 字段: Kinematic
身体类型：运动学



Kinematic 的Rigidbody2D同样用于在物理模拟下进行移动，但与**Dynamic**不同的是，它只能在非常明确的用户控制下进行。而且，**Dynamic** Rigidbody2D受重力和力的影响，但是**Kinematic** Rigidbody2D不受重力和力的影响。出于这个原因，Kinematic比Dynamic刚体处理得更快，对系统资源的需求更低。 **Kinematic** Rigidbody 2D 被我们设计为通过 Rigidbody2D.MovePosition 或者 Rigidbody2D.MoveRotation方法，来进行位移和旋转。使用physics queries 来检测碰撞，并使用脚本来决定Rigidbody 2D应该在哪里以及如何移动。

Kinematic Rigidbody 2D 仍然通过它的速度(velocity)进行移动，但它的速度不受力或重力的影响；它仅与Dynamic Rigidbody 2D发生体积

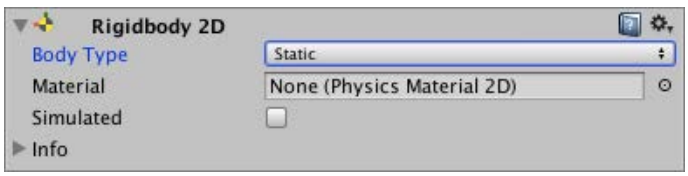
碰撞。 这点比较类似于Static Rigidbody 2D （见下文）。

在碰撞过程中，Dynamic Rigidbody 2D的行为就像一个不可移动的物体（就好像它具有无限的质量(mass)一样）。
这种身体类型不适用于和质量(mass)相关的属性。

Property:	Function:
Body Type	这个值用于，设置RigidBody 2D的组件的模式。以便您可以选择使用哪种运动（位置和旋转）行为，以及选择使用哪种Collider 2D交互。 选项有: **Dynamic**, **Kinematic**, **Static**
Material	使用它可以为所有连接到特定父Rigidbody 2D的Collider 2D指定一个通用材质。 **注意:** 如果你为一个Collider 2D组件，在它的Material 字段中指定了材质，那么这个Collider 2D组件，会使用你指定的这个 材质。 如果Rigidbody2D组件或Collider2D组件中没有指定材质，则他们会使用默认选项中的材质。 默认选项的材质为 **None(无) (Physics Material 2D)**。 如果你想修改默认材质，您可以在 Physics 2D Settings 窗口中设置它。 Collider 2D使用以下优先顺序(从高到低)确定要使用的材质(**Material**)： 1. Collider 2D组件本身指定的Physics Material 2D。 2. 在Rigidbody 2D上指定的Physics Material 2D。 3. 在物理2D设置(Physics 2D Settings)中指定的Physics Material 2D默认材质。 **提示**：使用第3种可以确保所有连接到相同的**Static** 身体类型的Rigidbody 2D都可以使用相同的材质。
Simulated	如果您希望Rigidbody 2D组件和任何附加的Collider 2D和Joint 2D在运行期间与物理模拟进行交互，请启用**Simulated**（勾选这个方框(让这个值为true)）。 如果禁用了该选项（该框未选中），则这些组件不会与物理模拟进行交互。 有关更多详细信息，请参阅下文的 Rigidbody 2D properties: Simulated 部分。此框在默认情况下被选中。
Use Full Kinematic Contacts	如果您想让**Kinematic** Rigidbody2D与所有Rigidbody2D体型相碰撞，请启用此设置（勾选此框）。 这会让**Kinematic**与**Dynamic** Rigidbody2D类似，不同之处在于物理引擎在接触另一个Rigidbody2D组件时，不会移动此Kinematic Rigidbody2D；相反，它作为一个不可移动的物体，具有无限的质量。 当禁用Use Full Kinematic Contacts时，Kinematic Rigidbody2D仅与Dynamic Rigidbody2D相碰撞。 有关更多详细信息，请参阅下面的 Rigidbody 2D properties: Use Full Kinematic Contacts。 此选项默认是不勾选的。

Collision Detection	定义如何检测碰撞2D之间的碰撞。 有两个选项：**Discrete**、**Continuous**
Discrete	将 **Collision Detection** (碰撞检测的方式)设置为 **Discrete** (***离散的***) 时，会产生一个问题：如果物体移动速度足够快，则在physics update(物理更新)期间，具有Rigidbody2D和Collider2D的GameObjects可以重叠或通过彼此。 碰撞接触只在新的位置产生。
Continuous	将 **Collision Detection** (碰撞检测的方式)设置为 **Continuous**(***连续的***)时，带有Rigidbody2D和Collider2D的GameObjects，在physics update(物理更新)期间不会相互穿透。 相反，Unity会计算任何Collider 2D的第一个影响点，并将GameObject移动到那里。 请注意，这需要比**Discrete** 选项更多的CPU时间
Sleeping Mode	这里可以自定义GameObject在休眠时如何“休眠(Sleeping)”。 休眠可以节省处理器时间。
Never Sleep	*永不休眠* （应该尽可能的避免这种情况，因为它会影响系统资源）。
Start Awake	*开始时休眠* GameObject最初是清醒的。
Start Asleep	*开始时睡着* GameObject最初是睡着的，但可以被碰撞唤醒。
Interpolate	*插值：* 定义如何在物理更新(physics update)之间插入GameObject的运动。 （运动趋于不稳定时很有用）
None	*无平滑*（移动不会有平滑的过渡(不应用插值)）
Interpolate	*根据上一帧 进行平滑的过渡* 根据前一帧中此GameObject的位置进行平滑移动。
Extrapolate	*根据下一帧 进行平滑的过渡* 根据下一帧中此GameObject的位置的预估值来平滑移动。
Constraints	定义对Rigidbody 2D运动的限制。 (位移限制 和 旋转限制)
Freeze Position	可以选择停止Rigidbody2D在世界坐标X轴 和 Y轴上的移动。
Freeze Rotation	可以选择停止Rigidbody2D在世界坐标Z轴上的旋转。

Body Type: Static



Static Rigidbody 2D 用于 根本不用在物理模拟下移动的情况。

任何物体与它发生碰撞， **Static** Rigidbody 2D 都会像一个不可移动的物体（好像它有无限的质量(mass)）。
它是使用资源最少的身体类型。
Static 仅与 **Dynamic** Rigidbody 2D相碰撞。 不支持两个 **Static** Rigidbody 2D相碰撞，因为它们两个都不能移动。

这种身体类型只有很少的属性：

Property:	Function:
Body Type	这个值用于，设置RigidBody 2D的组件的模式。以便您可以选择使用哪种运动（位置和旋转）行为，以及选择使用哪种Collider 2D交互。 选项有: **Dynamic**, **Kinematic**, **Static**
Material	使用它可以为所有连接到特定父Rigidbody 2D的Collider 2D指定一个通用材质***(Material)***。 **注意:** 如果你为一个Collider 2D组件，在它的Material 字段中指定了材质，那么这个Collider 2D组件，会使用你指定的这个 材质。 如果Rigidbody2D组件或Collider2D组件中没有指定材质，则他们会使用默认选项中的材质。默认选项的材质为 **None(无) (Physics Material 2D)**。 如果你想修改默认材质，您可以在 Physics 2D Settings 窗口中设置它。 Collider 2D使用以下优先顺序确定要使用的**材质**： 1. Collider 2D组件本身指定的Physics Material 2D。 2. 在Rigidbody 2D上指定的Physics Material 2D。 3. 在物理2D设置（Physics 2D Settings)中指定的Physics Material 2D默认材质。 **提示**：使用第3种可以确保所有连接到相同的 **Static** 身体类型 的Rigidbody 2D都可以使用相同的材质。
Simulated	如果您希望Rigidbody 2D组件和任何附加的Collider 2D和Joint 2D在运行期间与物理模拟进行交互，请启用**Simulated**（勾选这个方框(让这个值为true)）。 如果禁用了该选项（该框未选中），则这些组件不会与物理模拟进行交互。 有关更多详细信息，请参阅下文的 Rigidbody 2D properties: Simulated 部分。此框在默认情况下被选中。

有两种方法将Rigidbody 2D标记为 **Static**:

1. 对于有Collider 2D组件的GameObject来说，如果这个游戏物体完全不具有Rigidbody 2D组件。 那么，所有的这些Collider 2D组件，都被认为是附着在一个隐藏的**Static** Rigidbody 2D组件上的。
2. 如果GameObject身上有一个Rigidbody 2D组件，那么，把那个Rigidbody 2D组件的Body Type设置为**Static**。

第1种方法 是制作**Static** 的Collider 2D的简单方法。在创建大量需要用到Collider 2D组件的游戏物体时，如果这些Collider2D都不需要移动，那么你不必为它们添加Rigidbody2D组件。

第2种方法会让性能变得更好。 **Static** 的Collider2D需要在Start(初始化)或者移动时，重新配置隐藏的Rigidbody2D组件。那么当它具有自己的Rigidbody 2D组件时，会更快性能更好。 第1种方法会在初始化或移动时重新配置一组Collider，将它们全部作为一个父Rigidbody 2D的孩子标记为Static；而第二种方法不需要再内部进行这样的操作，所以更快。

注意：如上所述，Static Rigidbody 2D 用于不需要移动的物体，并且相交的两个Static Rigidbody 2D对象之间不会发生碰撞。 但是，如果其中一个Collider 2D设置为触发器(Trigger)，则Static Rigidbody 2D和Kinematic Rigidbody 2D将与之发生碰撞。 还有一个功能可以改变Kinematic 身体类型的相互作用（请参阅下面的 Use Full Kinematic Contacts）

Rigidbody 2D properties
Rigidbody 2D的属性

Simulated 模拟

使用**Simulated**属性可以停止(不勾选(设置为false)) 或开始(勾选(设置为true)) *一个Rigidbody 2D以及任何附加的Collider 2D和Joint2D* 与2D物理模拟的交互。
与启用或禁用单个Collider 2D和Joint 2D组件相比，更改**Simulated**字段所需要的内存会更少，而CPU的效率也会高得多。

当**Simulated**被勾选时(设置为true时)，会发生以下情况：

- Rigidbody2D通过物理模拟进行移动（应用重力和物理力）
- 任何附加的Collider 2D都会继续创建新的Contacts并不断重新评估它
- 任何附加的Joint2D都可以模拟并约束附加的Rigidbody 2D
- Rigidbody 2D、Collider 2D和Joint 2D的所有内部物理对象都保留(stay)在内存中

当**Simulated** 未勾选时(设置为false时)，会发生以下情况：

- 刚体2D不会被仿真移动(不会应用物理模拟进行移动)（重力和物理力都不适用）
- Rigidbody 2D不会创建新的Contacts，并且任何附加的Collider 2D 中的Contacts都将被销毁
- 任何附加的Joint2D都不会被应用物理模拟，也不会去限制任何附加的Rigidbody2D
- Rigidbody 2D，Collider 2D和Joint 2D的所有内部物理对象都保留(left)在内存中

Why is unchecking Simulated more efficient than individual component controls?
为什么取消选择 Simulated 比控制别的组件更有效?

在2D物理模拟中，Rigidbody2D组件控制附加的Collider2D组件的位置和旋转，并允许Joint2D组件将这些位置和旋转用作锚点(anchor point)。
当它连接的Rigidbod 2D组件进行移动时，Collider2D也会跟着移动。然后会计算此Collider2D组件 与其他Rigidbody2D附加的Collider2D之间的接触。 Joint2Ds也会限制刚体的位置和旋转。

以上这些所有的物理模拟都需要时间。

因此，我们明白了一件事情：您可以通过单独启用/禁用某个组件，来停止/启动这个组件的2D物理模拟。 您可以在Collider 2D和Joint 2D组件上执行此操作。 但是，启用和禁用与物理模拟相关的单个组件，会导致内存使用和CPU能耗成本。 当**Simulated**字段被禁用时，2D物理引擎不会产生任何内部基于物理学的对象来模拟。 当启用**Simulated**字段时，2D物理引擎确实具有内部基于物理学的对象来可以模拟。 而启用和禁用与2D物理模拟相关的组件，意味着必须创建和销毁内部GameObjects和基于物理的组件。
你看出来了吗？禁用**Simulated**字段比禁用单个组件更容易，更高效。

注意：当未勾选Rigidbody 2D的Simulated选项时，任何附加的Collider 2D实际上都是“不可见的”，也就是说， 它不能被任何physics querie(物理检测)检测到，例如 Physics.Raycast.

Use Full Kinematic Contacts

使用完整的Kinematic Contacts

如果你希望Kinematic Rigidbody2D与所有Rigidbody2D身体类型相碰撞，请启用此设置（勾选复选框）。 这会让Kinematic Rigidbody2D 与Dynamic Rigidbody2D非常相似。不同之处在于，在碰撞到另一个Rigidbody2D时，物理引擎不会移动Kinematic Rigidbody2D，它就像是一个拥有无限质量(mass)的不可移动的物体。

当此设置被禁用（未勾选）时，Kinematic Rigidbody2D仅与Dynamic Rigidbody2D碰撞；它不会与其他Kinematic Rigidbody2D或Static Rigidbody2Ds发生碰撞（请注意，触发器(Trigger)不算）。 这意味着不会发生与碰撞相关的脚本回调 (OnCollisionEnter, OnCollisionStay, OnCollisionExit 都会失效)

当您使用物理检测(physics queries)(如 Physics.Raycast) 来检测Rigidbody 2D应该在何处移动、以及何时需要移动多个Kinematic Rigidbody 2D、以及何时需要彼此交互时，这可能会很不方便。 启用**Use Full Kinematic Contacts**可以使Kinematic Rigidbody2D组件以这种方式进行交互。

Use Full Kinematic Contacts允许对Kinematic Rigidbody2D进行明确的位置和旋转控制，但同时也允许发生与碰撞相关的回调(OnCollisionEnter, OnCollisionStay, OnCollisionExit)。
在需要对所有Rigidbody2D进行非常明确的用户控制(explicit control)的情况下，如果你想使用Kinematic Rigidbod 2D代替Dynamic Rigidbody2D(让Kinematic Rigidbod 2D也可以发生与碰撞相关的回调)，你只需要勾选**Use Full Kinematic Contacts**。