事件

本页介绍了 TopDown 引擎中包含的事件系统。

介绍

自顶向下引擎包括它自己的**事件管理器**，并依赖它在类之间进行通信。事件是由应用程序中的类广播的消息，可以被任何**侦听**类捕获，以便它可以采取行动。例如，让我们考虑您关卡中的敌人。当你的角色杀死它时，它可能会增加分数，但也会算作成就的进度，你需要以某种方式更新 GUI。你当然可以让你的 Kill 方法直接调用所有这些，但它会产生**耦合**和**依赖关系**。另外，有时您不知道需要通知多少其他类才能通知敌人死亡。这就是事件发生的地方。

事件是您**传播**某事刚刚发生的事实的一种方式。**任何班级都可以听它**，并采取适当的行动。它们对于扩展引擎和实现您自己的功能非常有用，而无需修改基本代码。

MMEventManager

MMEventManager 类是一个静态类（意味着您不需要将其添加为场景中的组件），它负责事件广播，并让侦听器知道事件已被触发。它是您需要用来触发或侦听事件的唯一类。

从任何类触发事件都非常简单，您只需要调用 YOUR_EVENT.Trigger(YOUR_EVENT_PARAMETERS); 例如，这里我们向所有侦听器广播一个名为 GameStart 的 MMGameEvent：

```
MMGameEvent.Trigger("GameStart");
```

侦听事件可能有点棘手，因为您需要将一些内容添加到类中，以便它可以正确侦听。确保您不要忘记这些步骤之一！

第 1 步：指定您的类为此类事件实现 MMEventListener 接口。这是在班级的顶部完成的。例如，这是 MMAchievementDisplayer 的声明：

```
public class MMAchievementDisplayer : MonoBehaviour,
MMEventListener<MMAchievementUnlockedEvent>
{
  // ... the content of the class goes here
}
```

如您所见，这里我们说这个类将侦听 MMAchievementUnlockedEvents。

第 2 步：我们需要在 OnEnable 上开始监听事件，并在 OnDisable 上停止监听它们。这实际上让 MMEventManager 知道这个实例希望被通知任何被触发的那种类型的事件。确保你也停止监听 OnDisable 事件，否则 MMEventManager 可能会尝试"联系"这个实例，即使它被禁用或销毁，这会导致错误。这很简单：

```
void OnEnable()
{
  this.MMEventStartListening<MMAchievementUnlockedEvent>();
}
void OnDisable()
{
  this.MMEventStopListening<MMAchievementUnlockedEvent>();
}
```

当然，您会希望为要收听的每种类型的事件执行此操作。

第 3 步：

剩下要做的就是为该事件实现 MMEventListener 接口：

```
public virtual void OnMMEvent(MMAchievementUnlockedEvent
achievementUnlockedEvent)
{
  // here we start a coroutine that will display our achievement
  StartCoroutine(DisplayAchievement
(achievementUnlockedEvent.Achievement));
}
```

该引擎包含大量事件使用示例，检查它们，并利用事件的潜力来改进您自己的游戏！

事件类型

资产带有许多预定义的事件类型，但它可以与任何类型的事件一起使用，只要它们是结构体，因此可以随意创建您的事件类型。您可以使用以下示例。与资产捆绑在一起的是以下事件类型：

MMGameEvent :

MMGameEvents 是简单的多用途事件，仅由字符串名称组成。您可以使用这些来触发不需要比该名称更多信息的事件（游戏开始，诸如此类）。

```
public struct MMGameEvent
{
  public string EventName;
  public MMGameEvent(string newName)
  {
    EventName = newName;
  }
}

// trigger one like this (here we're broadcasting a save event):
MMGameEvent.Trigger("Save");
```

TopDownEngineEvent：

TopDownEngineEvents 是 MMGameEvents 的替代方案。它们由 TopDownEngineEventTypes 而不是字符串定义。TopDownEngineEventTypes 在枚举中定义。这可以防止您在使用字符串时出现拼写错误。

```
public enum TopDownEngineEventTypes
{
  LevelStart,
  LevelEnd,
  PlayerDeath
}

// trigger one like this
TopDownEngineEvent.Trigger(TopDownEngineEventTypes.LevelStart);
```

MMCharacterEvent :

通常从能力内部触发，您可以使用这些来让游戏的其余部分知道角色执行了特定动作。所述动作在枚举中定义。大多数情况下，您应该对状态机事件没问题，但如果您需要更多，也有此选项。

```
public enum MMCharacterEventTypes
{
  ButtonActivation,
  Jump
}

// trigger one like this:
MMCharacterEvent.Trigger(_character, MMCharacterEventTypes.Jump);
```

PickableItemEvent :

当它们被拾取时由可拾取物品触发。

MMDamageTakenEvent：

每次受到伤害时触发。

CheckpointEvent：

每次到达检查点时触发。

MMAchievementUnlockedEvent：

MMAchievementUnlockedEvents 通常在成就解锁时触发。然后它可以被 GUI 捕获并用于显示成就，但也可以用于增加分数等。

MMSfxEvent：

MMSfxEvent 是允许您请求 SoundManager 播放特定声音的事件。诚然，你也可以直接做 SoundManager.Instance.PlaySound(clipToPlay,transform.position); 两者都会起作用。事件允许您这样做，而不会在缺少声音管理器的情况下冒错误的风险。事件只会被广播，没有任何东西会拦截它（并且声音不会播放，但至少你不会收到错误）。

```
// trigger one like this (here we're asking for the playing of a sound
set in the achievement displayer's inspector):
MMSfxEvent.Trigger(achievement.UnlockedSound);
```

同样，事件可以是任何类型的结构，也可以是非常复杂的结构，因此可以随意创建自己的结构并将它们与该系统一起使用。

状态机

StateMachines（例如在角色系统中使用）也可以在**每次更改状态时**自动触发事件。这特别有用，因为它可以为您省去手动触发它们的麻烦，您可以专注于您的听众。这些特定事件是 MMStateChangeEvent，可以与任何类型的状态机一起使用的通用事件。获取事件将使您知道状态机的目标、它处于什么新状态以及它以前处于什么状态。如果您不打算使用事件，您可

以通过 Character 组件的检查器为 Character 关闭它们。

```
public struct MMStateChangeEvent<T> where T: struct, IComparable,
IConvertible, IFormattable
{
  public GameObject Target;
  public MMStateMachine<T> TargetStateMachine;
  public T NewState;
  public T PreviousState;

  public MMStateChangeEvent(MMStateMachine<T> stateMachine)
  {
    Target = stateMachine.Target;
    TargetStateMachine = stateMachine;
    NewState = stateMachine.CurrentState;
    PreviousState = stateMachine.PreviousState;
  }
}
```