УДК 004

РАЗРАБОТКА СИСТЕМЫ ШИРОКОВЕЩАТЕЛЬНЫХ РАССЫЛКИ СООБЩЕНИЙ НА ОСНОВЕ UNITY3D

В.А. Коровкин, В.А. Лисьев
Научный руководитель: Л.И. Ямпольская, к. филос. н., начальник ОРОТ ИК ТПУ
Национальный исследовательский Томский политехнический университет
E-mail: alcasar@tpu.ru

Abstracts. *Thisarticle describes simple method for creating broadcast event system with using Unity3D platform. This method can be used for standalone (GNU/Linux, Windows, OS X) and mobile platforms.*

Key words: unity3d, broadcast event system, game developing, GNU/Linux, Windows, OS X.
Ключевые слова: unity3d, проектирование, широковещательная рассылка сообщений.

При разработке игр, довольно часто возникает необходимость в построении системы широковещательной рассылки сообщений. Предположим необходимо сделать так, чтобы в тот момент, когда персонаж, управляемый игроком, вошел в определенную зону, или выполнил определенное действие, все заинтересованные в этом другие объекты получили уведомление. По возможности это уведомление должно нести в себе информацию о произошедшем событии. В данной статье предлагается один из возможных способов построения подобной системы на базе Unity3D EventSystem.

Начиная с версии 4.6 в состав Unity3D включена UI System, значительно упрощающая процесс создания UI. К тому же она является Open Source проектом. В основе этой системы лежат два очень важных компонента – EventSystem и InputModules, позволяющие принимать и обрабатывать события. InputModules являются наследниками UIBehaviour, который в свою очередь наследует MonoBehaviour и содержат в себе логику обработки событий, поступающих от EventSystem.

Отправка события определенному GameObject осуществляется посредством вызова метода ExecuteEvents.Execute(). Определение метода имеет следующий вид:

```
    public static bool Execute<T>(GameObject target, BaseEventData
data, EventFunction<T> functor) where T : IEventSystemHandler;
```

Вызывая этот метод, в качестве параметров должно передаваться ссылка на GameObject в списке ассоциированных компонентов которого, должен присутствовать InpuModule реализующий интерфейс T или интерфейс-наследник T. Если таких компонентов окажется несколько, все они будут вызваны по очереди.

Для отправки события необходимо иметь ссылку на целевой GameObject, что не подходит для широковещательной рассылки. Решением этой проблемы может послужить коллекция содержащая список GameObjects с прикрепленными InputModules способными обрабатывать широковещательные события.

```
    public abstract class BroadcastInputModule<TEventType> :
BaseInputModule
            where TeventType : IEventSystemHandler
    {
        protected override void Awake()
        {
            base.Awake();
            BroadcastReceivers.RegisterBroadcastReceiver<TEventType
>(gameObject);
        }
```

```
        protected override void OnDestroy()
        {
            base.OnDestroy();
            BroadcastReceivers.UnregisterBroadcastReceiver<TEventTy
pe>(gameObject);
        }
    }
```

Класс BroadcastInputModule служит базовым для модулей обработчиков широковещательных событий. Его основной задачей является регистрация модуля в этой коллекции. Создадим модуль, который будет реагировать на глобальное событие, которое условно будет называться «SomethingHappened». Этот компонент должен быть добавлен ко всем GameObjects заинтересованным в получении события *SomethingHappened*. Интерфейс IsomethingHappenedEventHandler должен выглядить следующим образом:

```
public interface IsomethingHappenedEventHandler :
IEventSystemHandler
{
    void OnSomethigHappened(SomethingHappenedEventData data);
}
```

Коллекция хранящая обработчики может быть довольно простой и сделана с помощью Dictionary. Последним элементом является класс BroadcastExecuteEvents:

```
public static class BroadcastExecuteEvents
{
    public   static   void   Execute<T>(BaseEventData   eventData,
    ExecuteEvents.EventFunction<T> functor)
        where T : IEventSystemHandler
    {
        var handlers = BroadcastReceivers.GetHandlersForEvent<T>();
        if (handlers == null) return;
        foreach (var handler in handlers)
        {
            ExecuteEvents.Execute<T>(handler, eventData, functor);
        }
    }
}
```

Как видно из его определения, он является всего лишь обёрткой над ExecuteEvents и выполняет всего одну задачу – выбор подходящих обработчиков для указанного события и их вызов. Теперь произвести широковещательную посылку события можно так:

```
BroadcastExecuteEvents.Execute<ISomethingHappenedEventHandler>(nul
l, (i, d) => i.OnSomethigHappened(new
SomethingHappenedEventData()));
```

Список литературы

1. Unity3D Documentation [Электронный ресурс]. URL: // http://docs.unity3d.com/Manual/index.html. (Дата обращения: 10.03.2015).

2. Ахо А.В., Хопкорфт Д.Э., Ульман Д.Д. Структуры данных и алгоритмы: пер. с англ.: М.: Издательский дом «Вильямс», 2003 г., 384 с.