Touch System Documentation

TouchEvent Class


```
public static int EventCount { get;
private set; }
```

This keeps track of number of TouchEvents instantiated throughout your game.

(You wouldn't need this, so you can left out it).

```
public delegate bool TouchCondition
(Vector2 inputPos);

public TouchCondition Condition;
```

The touch condition of your controls.

When any of current touches will satisfy the condition then Touch Manager will automatically manage the tracking of that touch with one-to-one mapping to your controls across subsequent frames updates.

The Touch System passes the touch positions one-by-one for all the touch events in this TouchCondition Callback function.

NOTE: *inputPos vector has its origin as the center of screen of your device.*

for example: (0,0) maps to a point in the center of the screen. and (-Screen.width * 0.5f, -Screen.height * 0.5f) maps to the bottom right corner of the screen.

```
public bool
IsCheckConditionEveryFrame;
```

You can set this to true if you want to Remap the touches to their corresponding controls (Touch Events). But this performance intensive and useless, so don't use this ever in your game, it must its value false.

If it has value false. Then the Touch System will automaticaly maps the touches to their corresponding controls when some new touches appears or some touches disappears from the screen. This is not performance intensive.

```
public bool
IsCallOnEndedAfterTouchDataLost;
```

You wouldn't need this, so leave this out.

```
public int LayerID;
```

The layer in which this touch event will occur.

Higher the layer id, The touch event will occur first and then the second touch event, if the two or more touch events are registered for the same condition.

```
public Touch touch;
```

If the Touch Event has occured or occuring then you can get the information about the touch by using this field.

CALLBACKS:

`public Action OnBegan;`	This callback is called when the TouchEvent is Began to occur. It can be used for Button Down Actions (Firing or Buttons).
`public Action OnMoved;`	This callback is called when the TouchEvent is currently occuring and the touch is moving. It can use be used for drawing something or dragging something (Joysticks etc.).
`public Action OnStationary;`	This is callback is called when the TouchEvent is currently occuring and the touch is stationary (not moving). It can be used for grabbing something.
`public Action OnEnded;`	This callback is called when the TouchEvent is just Ended. This is useful for Releasing something (Button Up Actions).

Touch Layer class

You don’t need to explore anything about this class, because this is internally used by the touch system.

But you must have to understand the concept of touch layers,

A touch layer is basically a number (integer) which determines the mapping order of touches to their corresponding controls.

If you don’t understand this, don’t worry you can watch the step-by-step video tutorial on YouTube.

Touch Manager Class

public static void RegisterEvent (TouchEvent touchEvent);

Call this function to register a touch input event.

Example:

```
bool IsOnLeftSide(Vector2 position)
{
    return position.x < 0;
}

TouchEvent touch_event;
void Start()
{
    touch_event = new TouchEvent();
    touch_event.Condition      = InOnLeftSide;
    touch_event.LayerID = 0;
    touch_event.OnBegan = Fire();
    TouchManager.RegisterEvent(touch_event);
}

void Fire()
{
    Debug.Log("Fired");
}
```

public static void UnRegisterEvent (TouchEvent touchEvent);

Call this function to unregister a touch input (i.e. if you don't want to take input more)

public static bool IsTouched (TouchEvent touchEvent);

Returns true when the passed event has occured or occuring.

Example:

```
public float rotation_speed = 90;
TouchEvent square_touch_event;

bool IsInSquare(Vector2 point)
{
    return (-100 < point.x) && (point.x < 100)
        && (point.y < 100) && (point.y > -100);
}

void Start()
{
    square_touch_event = new TouchEvent();
    square_touch_event.LayerID = 0;
    square_touch_event.Condition      = square_touch_event;
    TouchManager.RegisterEvent(square_touch_event);
}

void Update()
{
    if(TouchManager.IsTouched(square_touch_event))
        Rotate();
}

void Rotate()
{
    transform.Rotate(Vector3.up * rotation_speed * Time.deltaTime);
}
```

public static void AddIgnoreLayer (int id);

Use this function to ignore a particular layer it acts like a mask

Example:

```
TouchEvent touch_event1;
TouchEvent touch_event2;

void IsInUnitRadius(Vector2 position)
{
    return position.sqrMagnitude <= 1.0f;
}

void Start()
{
    touch_event1 = new TouchEvent();                //Bottom layer
    touch_event1.LayerID = 0;
    touch_event2.Condition      = IsInUnitRadius;

    touch_event2 = new TouchEvent();
    touch_event2.LayerID = 1;                       //Top layer
    touch_event2.Condition      = IsInUnitRadius;

    /*Both the events will occure for two touches in a unit radius
      but the event which has higher layer will be detected for the
      first touch and then second for touch.
    */
    TouchManager.RegisterTouchEvent(touch_event1);
    TouchManager.RegisterTouchEvent(touch_event2);

    TouchManager.AddIgnoreLayer(0);  /*Only event2 will be detected*/
    TouchManager.RemoveIgnoreLayer(0);
    TouchManager.AddIgnoreLayer(1);  /*Only event1 will be detected*/
}
```

We have given Many Demonstrations along with this package to Learn and Understand how the touch system can be used to create complex touch controls using layers and touch events.

This Touch System is VERY SIMPLE and EASY TO USE.

We have given the full DOCUMENTATION and YOUTUBE VIDEO TUTORIALS so that you can use and understand this system easily. THE LINK TO THE VIDEO TUTORIALS ARE GIVEN IN THEIR RESPECTIVE DEMONSTRATION SOURCE CODES, so check it out.