01

Vetor direção

Transcrição

Vimos como capturar o evento do mouse, mas isso ainda não resolve o nosso problema. Queremos com que a nave, quando em movimento, se dirija para onde o mouse estiver. O método do objeto `event` no método `mudarDirecao` que vai nos ajudar é o `getLocation` .

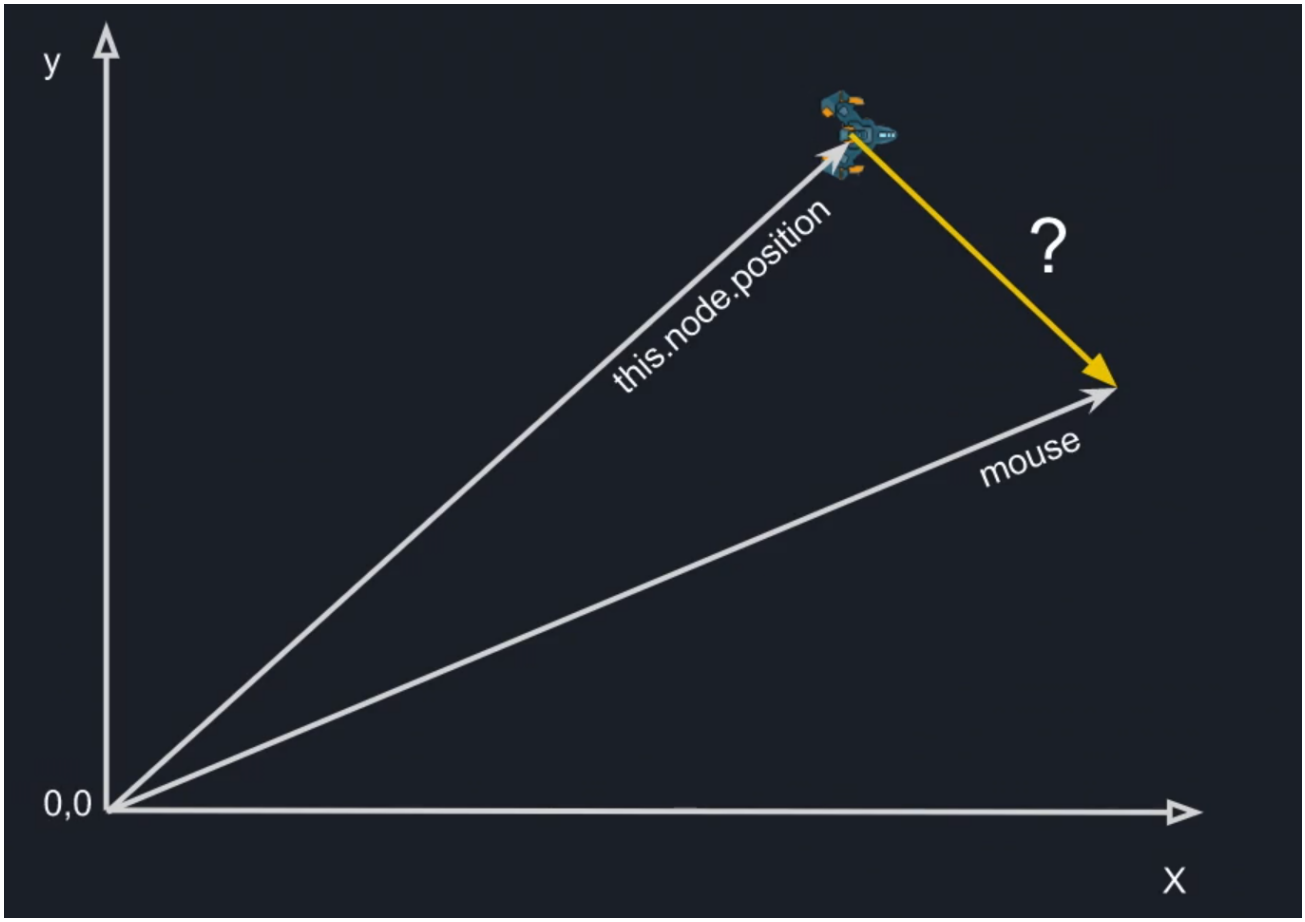
```
mudarDirecao: function(event){
    console.log(event.getLocation());
}
```

Esse método retorna um outro objeto que possui duas propriedade: X e Y, ou seja, as coordenadas do mouse dentro do *canvas*. Mas como a gente relaciona as propriedades de posicionamento da nossa nave com a localização do mouse?

Pensando em vetores

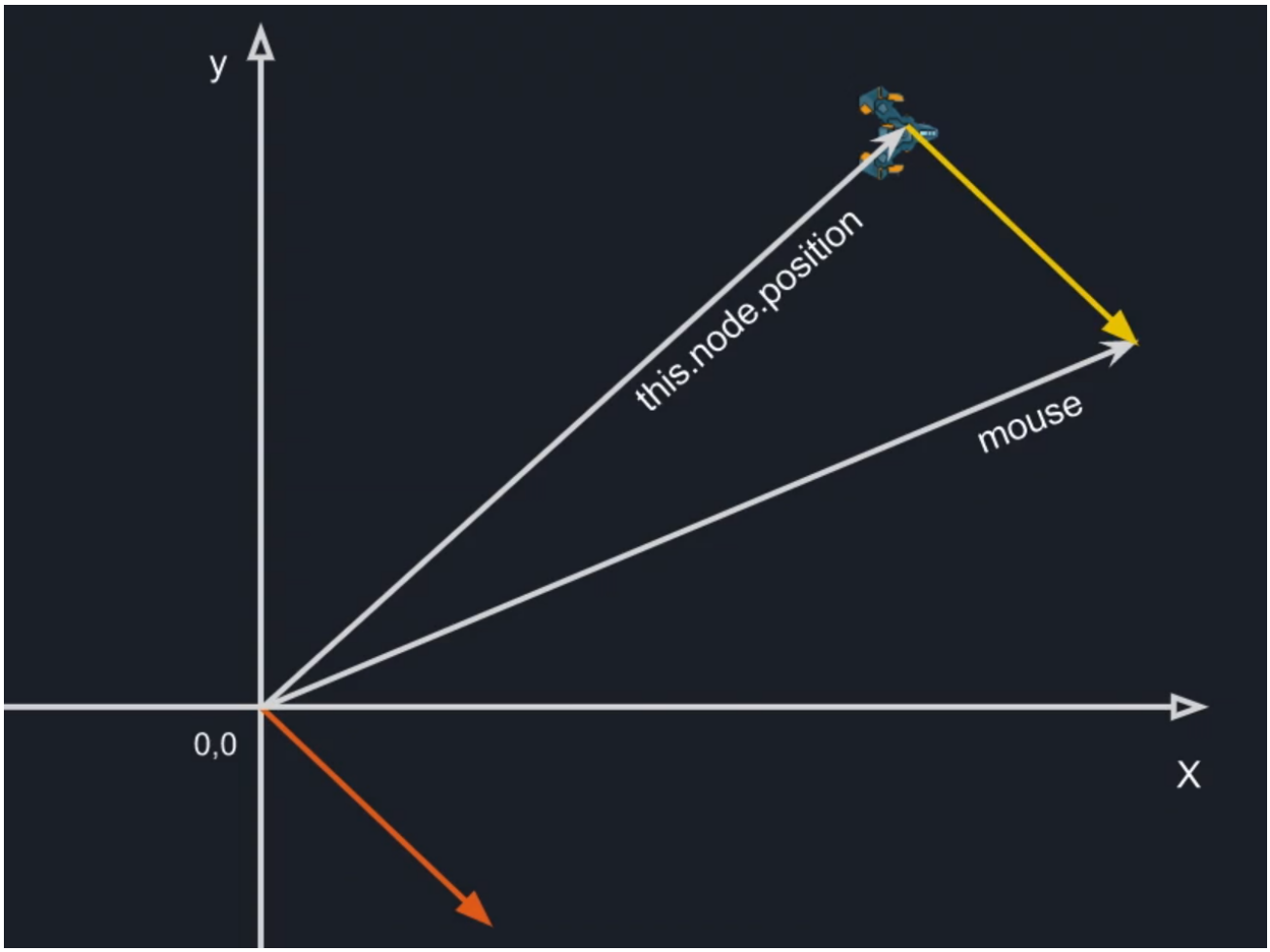
Vamos imaginar que o *canvas* em nosso jogo nada mais é que um plano cartesiano. Ou seja, teremos os eixos `x` e `y` iniciando nas coordenadas `0` e `0` e variando seus valores positivamente para a direita e para cima.

Neste plano, temos a posição do mouse como uma coordenada e considerando a origem do plano, podemos traçar uma reta da origem até o ponto atual do mouse. O mesmo vale para a nave. Então teremos dois vetores.



O que precisamos descobrir é o terceiro vetor que pode ser formado considerando a posição da nave e a posição do mouse. Neste caso, a linha amarela em nosso gráfico.

Uma alternativa seria pensar na substração dos vetores, porém, a substração dos dois vetores resultaria em um outro vetor de origem `0` no `x` e um valor negativo no `y`. Porém, se aproximássemos o vetor resultante do ponto de origem da nave de forma proporcional, teríamos o vetor que queremos. O que nos indicaria a *direção* do movimento.



Com o vetor *direção* em mãos, podemos somá-lo ao vetor *posição* da nave para que além da inversão da subtração dos vetores, a gente tenha o comportamento da nave partir do ponto em que ela está, ao invés de considerar as coordenadas fixas da origem. Vamos codificar isso?

Aplicando no código

No método `mudaDirecao`, capturamos a posição atual do mouse usando o método `getLocation` que nos devolve um objeto com as coordenadas `x` e `y`. Esse objeto não é um vetor, mas vamos transformá-lo em um usando a classe `Vec2` da Cocos da seguinte forma:

```
mudarDirecao: function(event){
    let posicaoMouse = event.getLocation();
    posicaoMouse = new cc.Vec2(posicaoMouse.x, posicaoMouse.y);
}
```

Agora podemos subtrair da posição do mouse o vetor da posição da nave obtendo como resultado um novo vetor `direcao` da seguinte forma:

```
mudarDirecao: function(event){
    let posicaoMouse = event.getLocation();
    posicaoMouse = new cc.Vec2(posicaoMouse.x, posicaoMouse.y);
    let direcao = posicaoMouse.sub(this.node.position);
}
```

Com o vetor `direcao` nas mãos, precisamos usá-lo no método `update` para somá-lo à posição atual da nave. Para isso criaremos uma nova propriedade em nosso componente chamado `direcao` da mesma forma que fizemos com o `acelerando`, essa propriedade será do tipo `Vec2`.

```
properties: {
    // foo: {
    //    default: null,      // The default value will be used only when the component attachi
    //                           to a node for the first time
    //    url: cc.Texture2D,  // optional, default is typeof default
    //    serializable: true, // optional, default is true
    //    visible: true,      // optional, default is true
    //    displayName: 'Foo', // optional
    //    readonly: false,    // optional, default is false
    // },
    // ...
    _acelerando: false,
    _direcao: Vec2,
},
```

E no método `mudarDirecao`, alterar o valor dessa propriedade sempre que o evento do mouse for disparado.

```
mudarDirecao: function(event){
    let posicaoMouse = event.getLocation();
    posicaoMouse = new cc.Vec2(posicaoMouse.x, posicaoMouse.y);
    let direcao = posicaoMouse.sub(this.node.position);

    this._direcao = direcao;
}
```

Por fim, atualizamos o método `update` para realizar a soma dos vetores e reposicionar a nave conforme calculamos.

```
update: function (dt) {
    if(this._acelerando){
        this.node.position = this.node.position.add(this._direcao);
    }
},
```

Podemos testar! Você vai notar que inicialmente a nave não se move mais, justamente porque ela só se move se pressionarmos a letra `a`, mas o mais bizarro é que caso a tecla se mantenha pressionada, a nave segue exatamente a posição do mouse, mas caso o mouse esteja mais distante e a tecla seja pressionada apenas uma vez, a nave dará um salto. Estranho isso, em vez de mover a nave na direção do mouse, nós estamos teletransportando ela.