

2020

BGE + PYTHON

```
177         ),
178         default='v',
179     )
180     global_scale_setting = FloatProperty(
181         name="Scale",
182         min=0.01, max=1000.0,
183         default=1.0,
184     )
185
186     def execute(self, context):
```

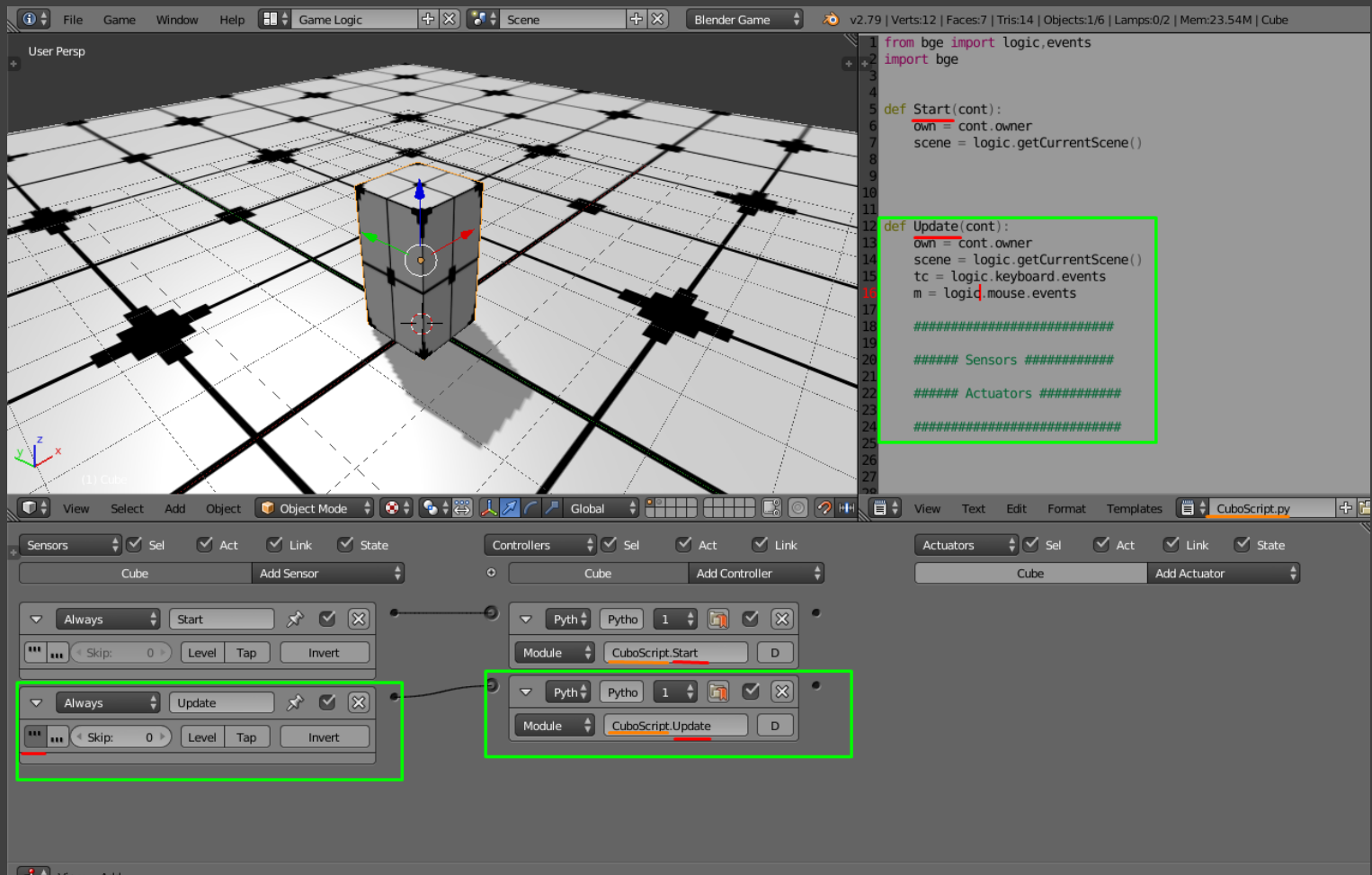
```
90     # get objects selected in the viewport
91     viewport_selection = bpy.context.selected_objects
92
93     # get export objects
94     obj_export_list = viewport_selection
95     if self.use_selection_setting == False:
96         obj_export_list = [i for i in bpy.context.scene.objects]
97
98     # deselect all objects
99     bpy.ops.object.select_all(action='DESELECT')
100
101     for item in obj_export_list:
102         item.select = True
103         if item.type == 'MESH':
104             file_path = os.path.join(folder_path, "{}.obj".format(item.name))
105             bpy.ops.export_scene.obj(filepath=file_path, use_selection=True,
106                                     axis_forward=self.axis_forward_setting,
107                                     axis_up=self.axis_up_setting,
108                                     use_animation=self.use_animation_setting,
109                                     use_mesh_modifiers=self.use_mesh_modifiers_setting,
110                                     use_edges=self.use_edges_setting,
111                                     use_smooth_groups=self.use_smooth_groups_setting,
112                                     use_smooth_groups_bitflags=self.use_smooth_groups_bitflags_setting,
113                                     use_normals=self.use_normals_setting,
114                                     use_uv=self.use_uv_setting,
115                                     use_material=self.use_material_setting,
```

Edinaldo Cicero
ÁtomoGames.
28/07/2020

BGE + PYTHON

Começando com Funções para Módulos:

Ao criar uma Função Start ou Update você há usa como Modulo no objeto que deseja usar no script, ou seja, o dono do script o próprio objeto.



Para chamar o script pelo Modulo você pega o nome do (script. Nome da Função) como visto acima no controlador python.

Lá na função Update (cont), esse (cont) corresponde á (controller) para quer serve, para chamar o logicbricks de controlador do objeto, ou seja, o que está em verde com isso você tem acesso aos sensores e actuators dos objetos que estiverem ligados nesse controlador.

Own ou (owner = cont.owner) aqui corresponde ao próprio objeto proprietário do script com isso em mão você pode manipular como quiser o próprio objeto exemplo:

Para movimentar o objeto own:

```
owner.applyMovement([x,y,z],True)
```

Veja como eu aplico o movimento para o dono do próprio script(owner.) depois de colocar o ponto você deve chamar um função da biblioteca, da class(KX_GameObjects) que pertence á todos os objetos do blender com isso tem varias funções para todos os objetos.

KX_GameObject (SCA_IObject) ¶

classe base - `SCA_IObject`

classe `bge.types.KX_GameObject(SCA_IObject)`

Todos os objetos do jogo são derivados dessa classe.

As propriedades atribuídas aos objetos do jogo são acessíveis como atributos desta classe.

Nota: Chamar QUALQUER método ou atributo em um objeto que foi removido de uma cena gerará um `SystemError`, se um objeto puder ter sido removido desde o último acesso a ele, use o `invalid` atributo para verificar.

https://docs.blender.org/api/2.79/bge.types.KX_GameObject.html#bge.types.KX_GameObject

getAxisVect(vect)

Retorna o vetor do eixo que gira pela orientação do espaço no mundo do objeto. Isso é o equivalente a multiplicar o vetor pela matriz de orientação.

Parâmetros: **vect** (*vetor 3D*) - um vetor para alinhar o eixo.

Devoluções: O vetor em relação à rotação dos objetos.

Tipo de retorno: Vetor 3d

applyMovement(movimento , local = Falso)

Define o movimento do objeto do jogo.

Parâmetros: • **movimento** (*vetor 3D*) - vetor de movimento.

• **local** -

◦ Falso: você obtém o movimento "global", isto é, relativo à orientação mundial.

◦ Verdadeiro: você obtém o movimento "local", isto é, relativo à orientação do objeto.

• **local** - booleano

applyRotation(rotação , local = Falso)

Define a rotação do objeto do jogo.

Parâmetros: • **rotação** (*vetor 3D*) - vetor de rotação.

• **local** -

◦ Falso: você obtém a rotação "global", isto é, relativa à orientação mundial.

◦ Verdadeiro: você obtém a rotação "local", isto é, relativa à orientação do objeto.

• **local** - booleano

Tento isso em mente você poderá perceber que o que estiver dentro da classe `Kx_GameObjects` poderá ser utilizado após o (`owner.`) .

Uma coisa importante, se lembre que para cada função que for feita para um objeto e for chamada em Modulo no objeto que desejas manipular o (`owner`) se passa a ser o próprio objeto certo, ou seja, cada função poderá ser usada em cada objeto porém variáveis que estiverem dentro do escopo de uma função não poderá ser acessada dentro de outra !.

Como Movimentar e Rotacionar um objeto e o que é Vetor2 E Vetor3:

Para movimentar ou rotacionar um objeto no universo do blender existe duas classes Vetoriais:

Para Movimentar:

1 = `owner.applyMovent([vetor3],Bool)`

`owner.applyMovent([X,Y,Z],Bool)`

Para rotacionar:

2 = `owner.applyRotation([vetor3],Bool)`

`owner.applyRotation([X,Y,Z] ,Bool)`

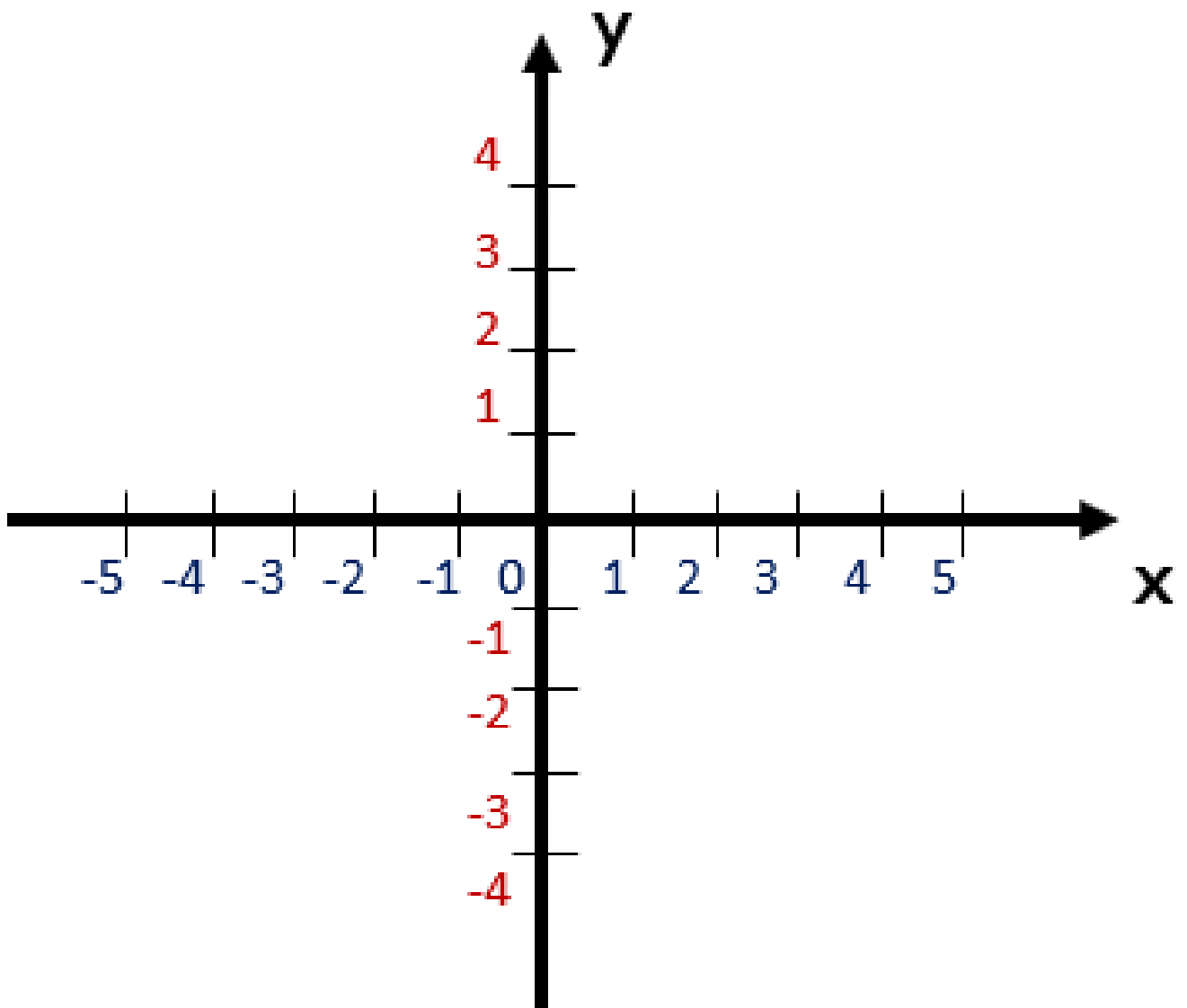
Vetor2 ou Vetor3 nada mais é na realidade um tipo de variavel que guarda posições geográficas, vector3 (X,Y,Z) se for vector2 seria (X,Y)

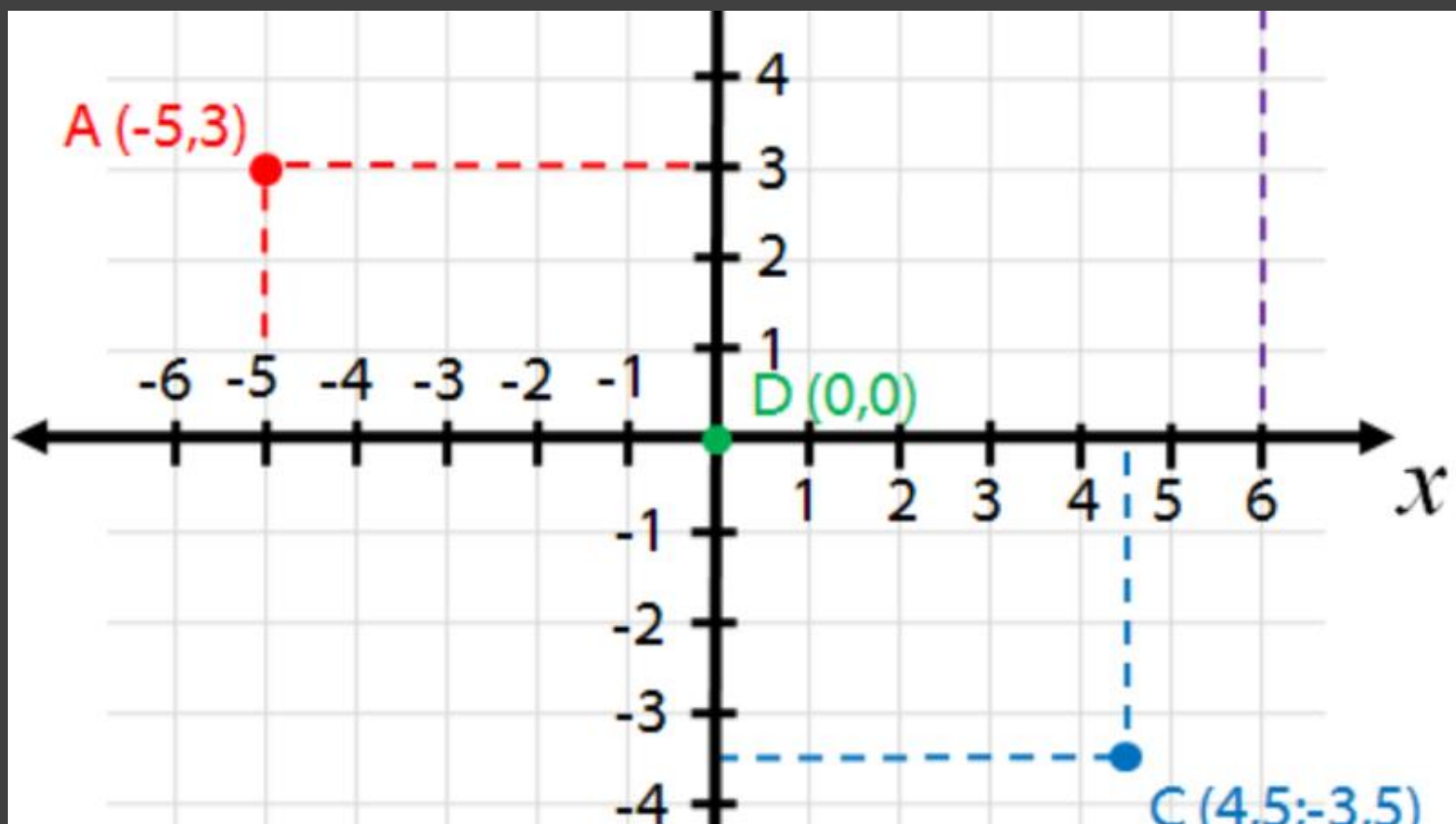
Com isso `own.applyMovement([x,y,z],True)` no lugar dos `([X,Y,Z])` coloque o valor que deseja que o objeto ande ou gire, tenha em mente que se você colocar (- ou +) na frente do valor desejado isso fará o objeto mudar de direção pois isso está baseado nos parâmetros geométricos do mundo 3D ou seja positivo para frente, negativo para trás e os valores tende á ser do tipo (Float) números quebrados ou numero seguidos de virgulas certo!, Pode sim usar números do tipo (Int) mais saiba que números inteiros são números exatos e corresponde a mais espaços para o objeto se deslocar pois o objeto não andar realmente ,claro! , mais sim se deslocar de acordo com o valor digitado, lembrou as aulas de física muito bem!

X = Largura , Y = Altura, Z = Profundidade

Vector2 pra Games 2d.

Vector3 pra Games 3d.





Como usar Inputs de Teclado e Mouse:

Para acessar teclas do teclado se deve chamar Chaves de Constantes, para isso chame todas as teclas do teclado:

Antes de chamar certifique-se de ter importado as bibliotecas de Logicas e eventos do blender assim:

```
From bge import logic, events
```

Feito isso agora crie uma variável com o nome que quiser mais que se torne fácil de lembrar claro e faça desse jeito:

```
Teclado = logic.keyboard.events
```

Se pode usar também os inputs de mouse basta chamar do mesmo jeito do teclado só que para o mouse:

```
Mouse = logic.mouse.events
```

```
If Mouse[events.LEFTMOUSE]:
```

```
    pass
```

Com isso pronto vamos usar cada tecla, e para isso vamos para a API com todas as teclas para usar: <https://docs.blender.org/api/2.79/bge.events.html>

E para usar a tecla escolhida basta criar uma condição de Input:

```
def Update(cont):
    own = cont.owner
    scene = logic.getCurrentScene()
    teclado = logic.keyboard.events
    m = logic.mouse.events
    #####
    ##### Sensors #####
    ##### Actuators #####
    #####

    if teclado[events.WKEY]:
        print("Tecla W precionada !!")
```

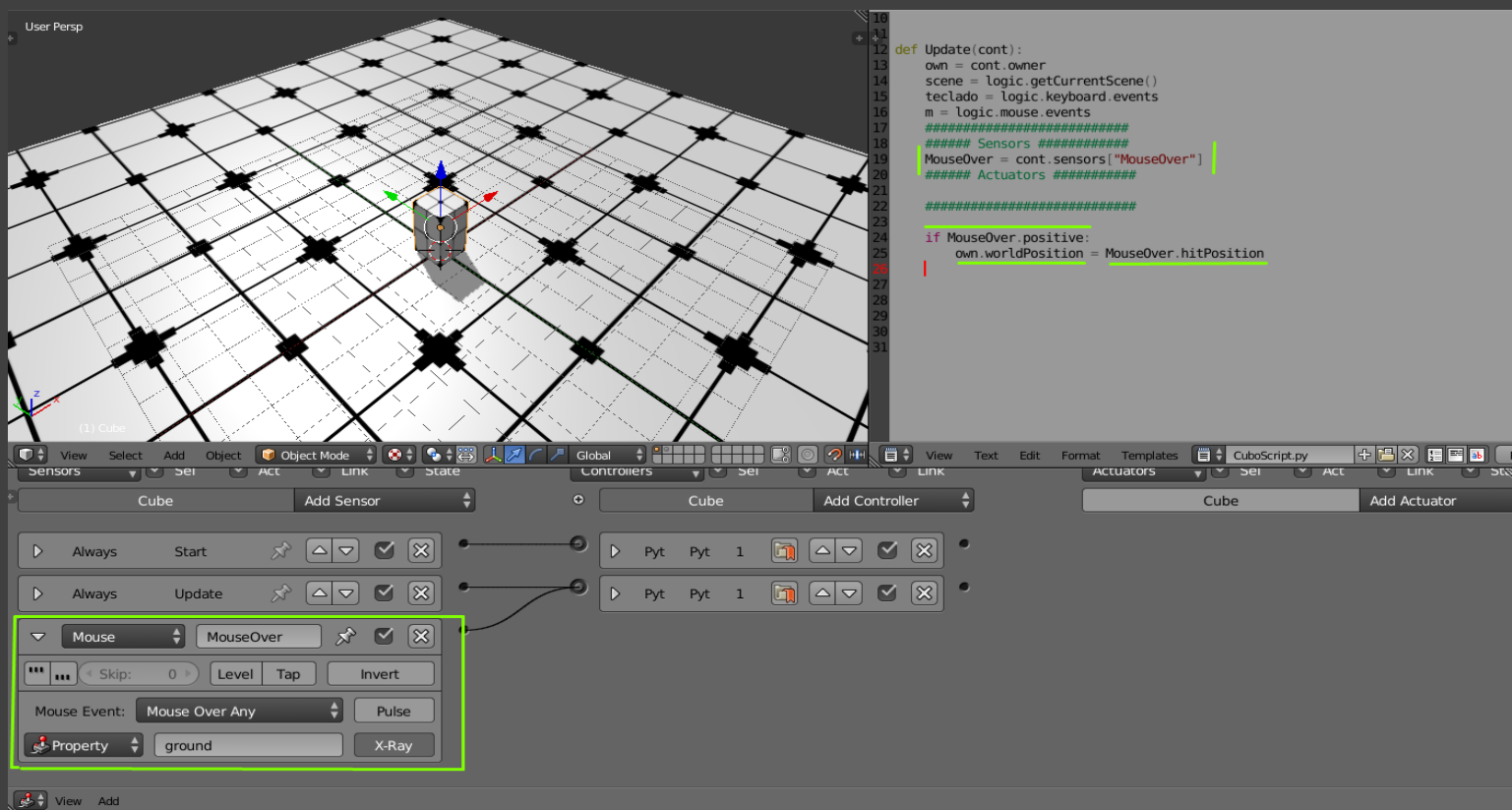
(If teclado[biblioteca de eventos do BGE. Tecla escolhida terminando com KEY no final])

Com isso temos uma condição que quando a tecla W for apertada, pois os dois pontos você aperta Inter e entrara no escopo da condição que será abaixo com isso você poderá fazer ações com o objeto colocando até o próprio own.applyMovement([x,y,z],True) e no lugar dos ([X,Y,Z]) coloque o valor que deseja que o objeto ande ou rotacione!

Como fazer um objeto seguir a posição do mouse:

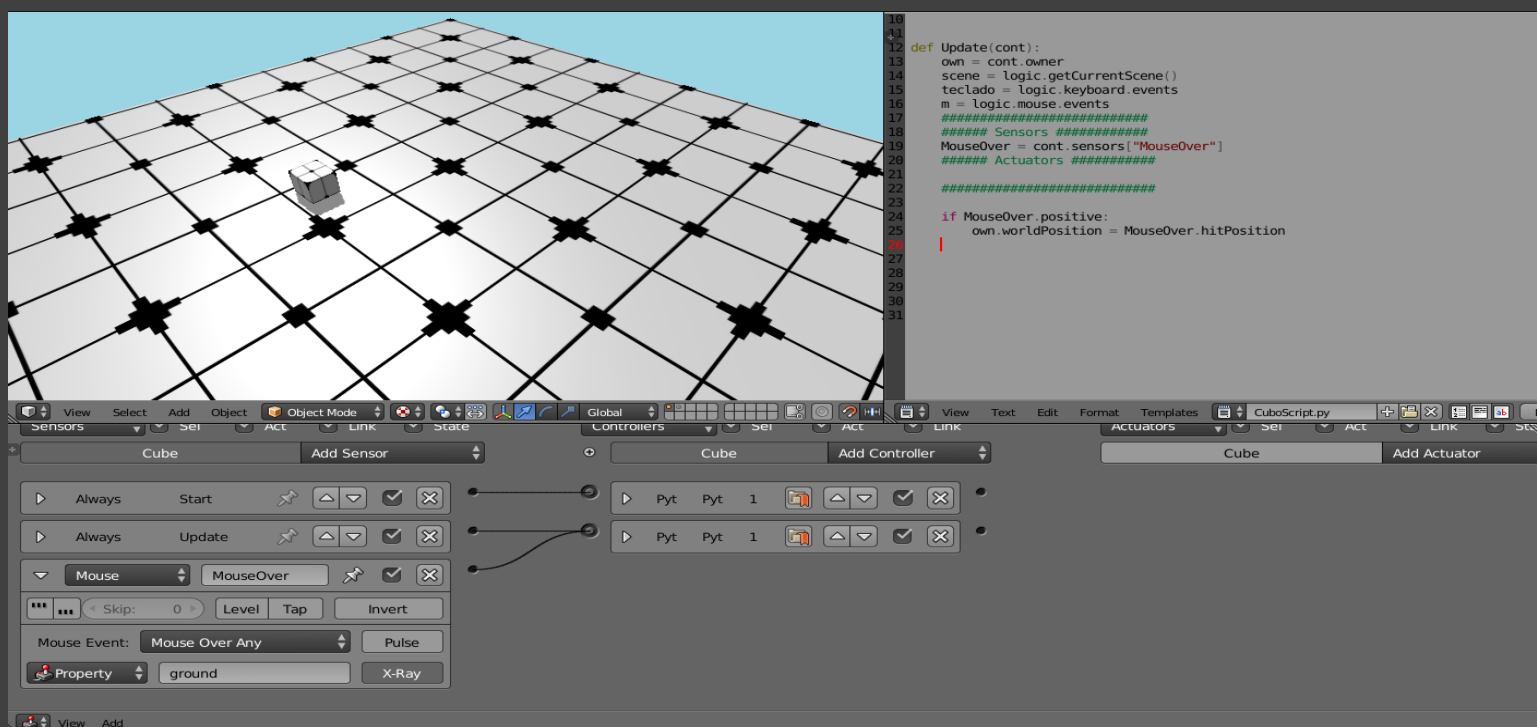


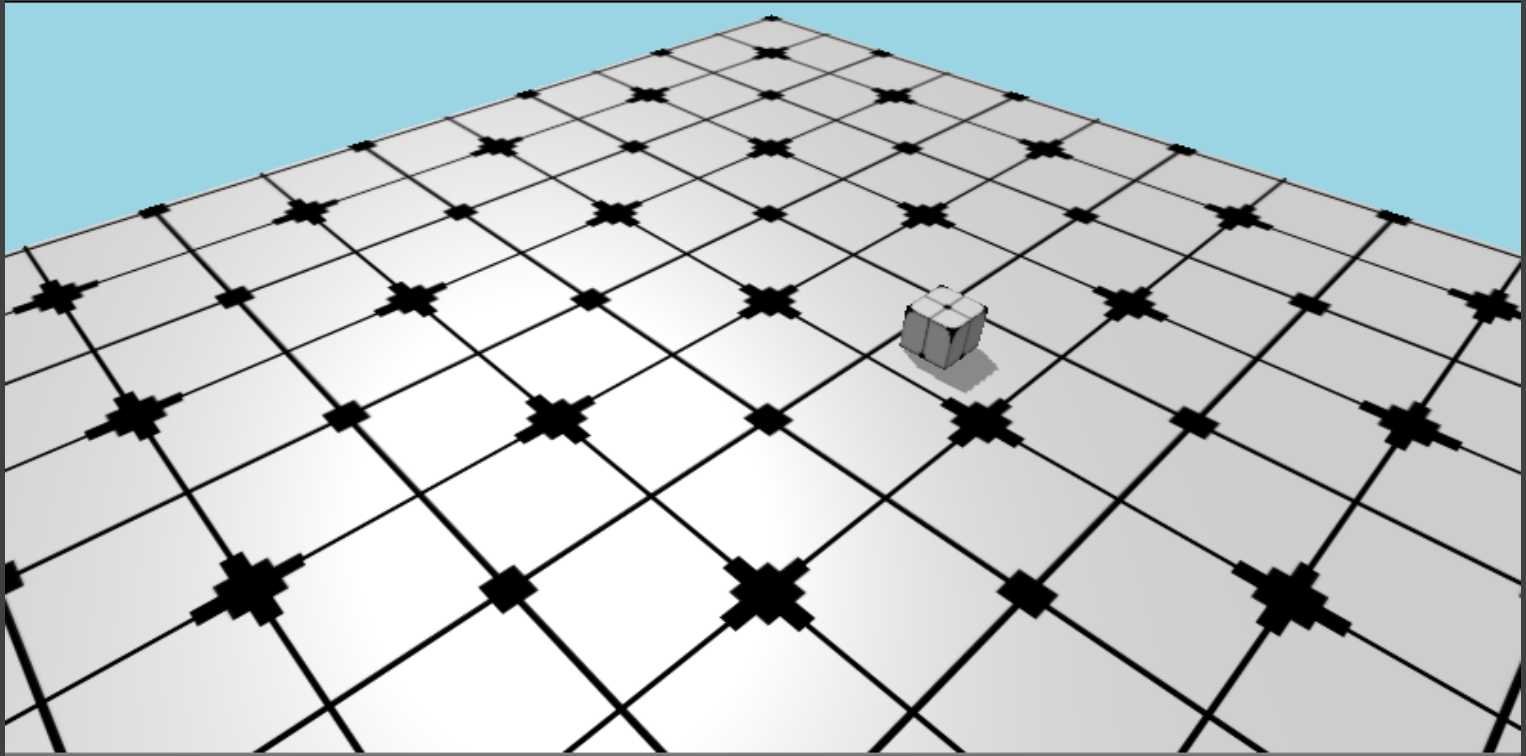
A propriedade "ground" foi colocada como exemplo mais pode ser qualquer uma.



Para um objeto seguir a posição do mouse é preciso ter um sensor do tipo (MouseOverAny), com alguma propriedade que seja um plano ou o chão onde o personagem vai pisar isso é necessário para o objeto não vim em direção á câmera sem ordem.

Sabendo disso agora vamos precisar de duas como posso dizer classes de posição geométricas que também trabalham com vetor3 que é (worldPosition) essa variável contém todas as posições que o objeto pode ter em relação ao universo 3D do blender/(x,y,z) e a outra é uma das funções do sensor (MouseOverAny) que é (MouseOverAny.hitPosition) e contém todos os hit que o mouse detecta em sua posição e isso é bom pois tem base de vetor3 também /(x,y,z). Veja o exemplo com o sistema pronto:



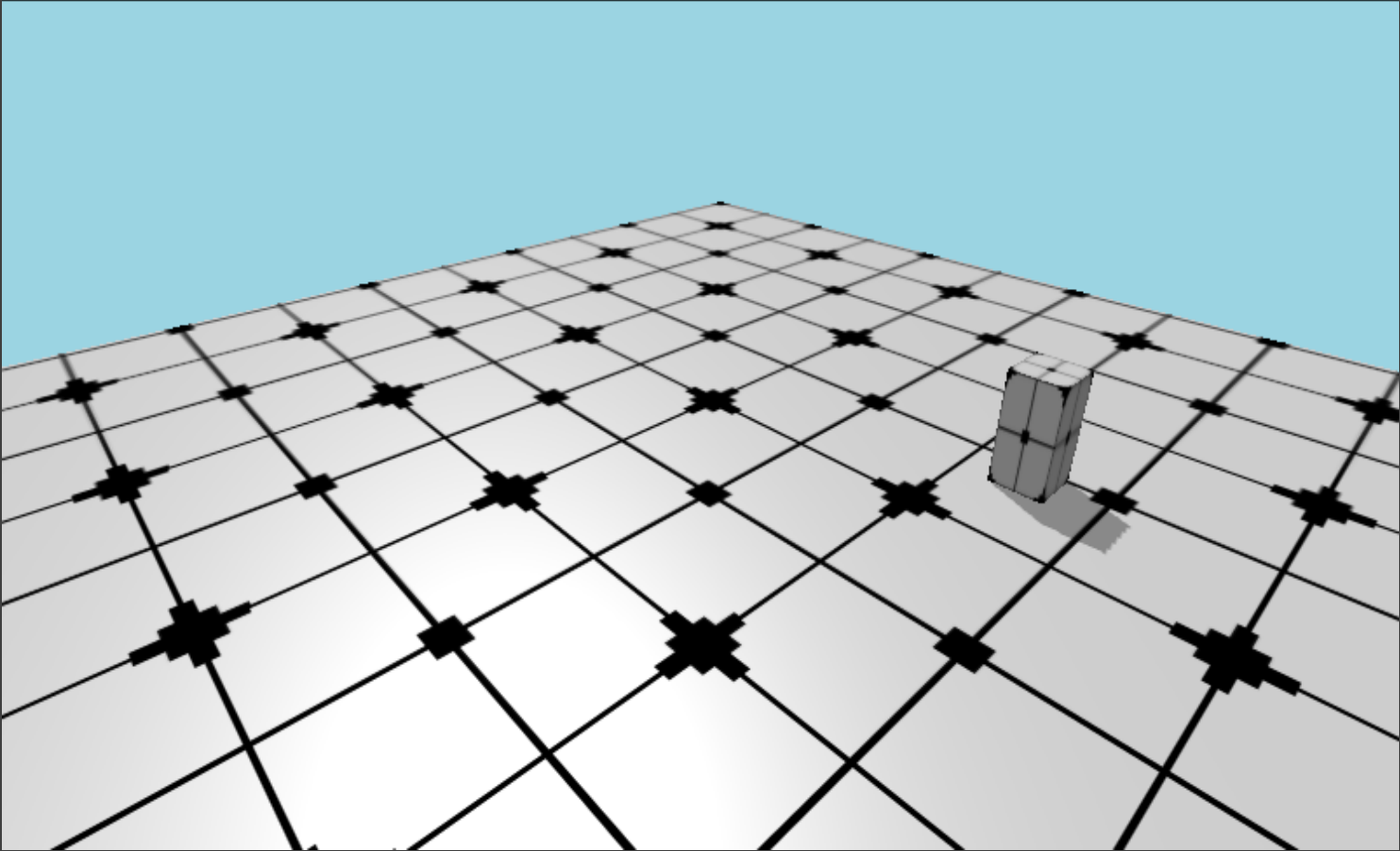


Mais note que o objeto está pela metade, pois, o resto está entrando no plano para resolver isso é muito simples você só precisar usar dois vetores em vez dos três, ou seja, (X, Y) e como fazer isso é simples:

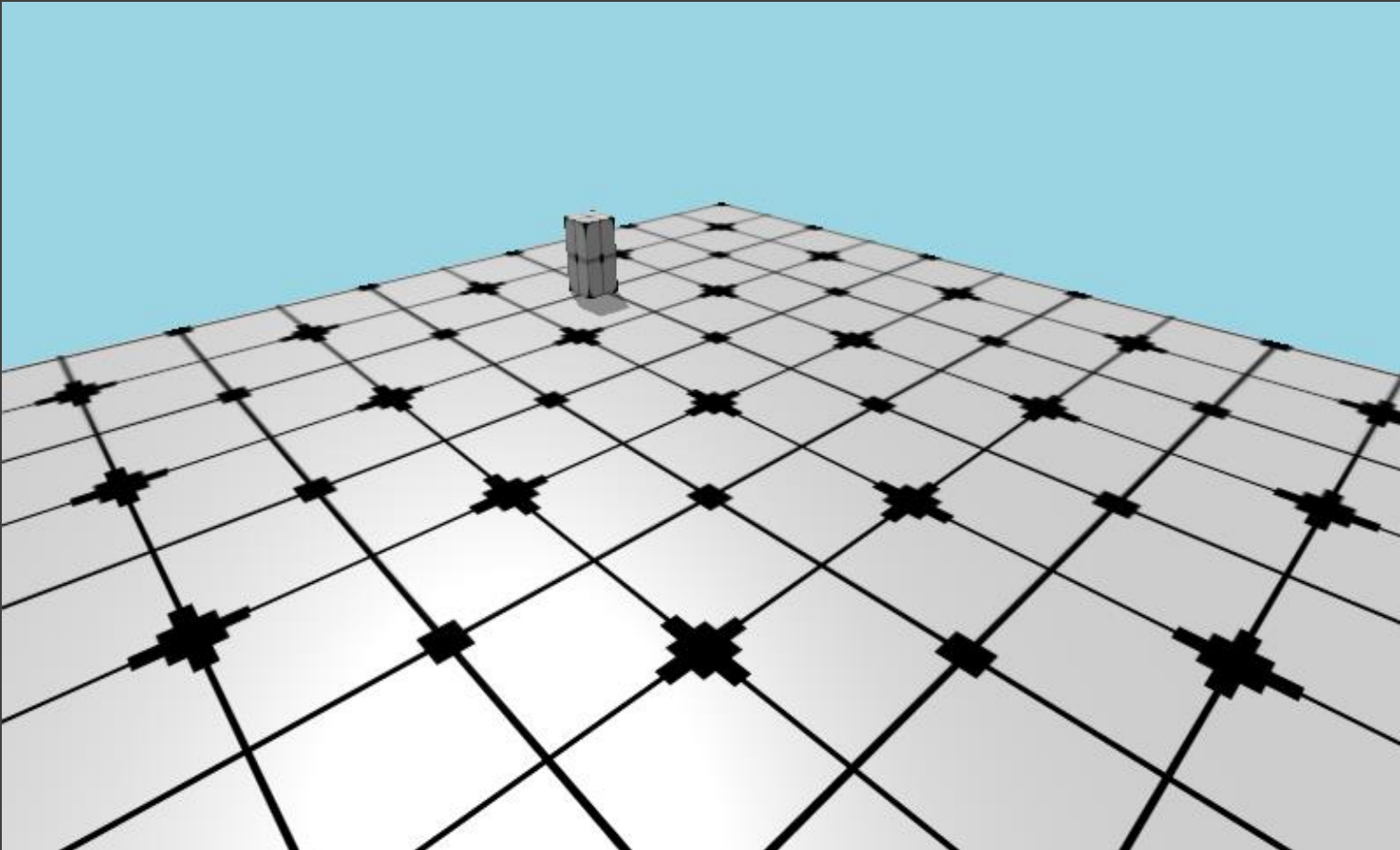
```
10
11
12 def Update(cont):
13     own = cont.owner
14     scene = logic.getCurrentScene()
15     teclado = logic.keyboard.events
16     m = logic.mouse.events
17     #####
18     ##### Sensors #####
19     MouseOver = cont.sensors["MouseOver"]
20     ##### Actuators #####
21
22     #####
23
24     if MouseOver.positive:
25         own.worldPosition.x = MouseOver.hitPosition.x
26         own.worldPosition.y = MouseOver.hitPosition.y
```

Ao colocar (.x , .y) em seus respectivos lugares você está a usar de forma direta do vetor desejado tanto no (worldPosition como no hitPosition) como o exemplo acima .

Posição 1:



Posição 2:



Se por acaso você está a usar o método em um personagem jogável mais não sabe como poder fazer para o personagem voltar a se movimentar com as teclas que você definiu no teclado basta criar uma condição (else:) logo abaixo do (if) pois é assim que funciona as estruturas de condição (if , elif e else:).

Estruturando condições if elif e else:

If , elif e else são as estruturas de condição ,praticamente a peça principal da programação, o if sempre está a cima pois ele é como posso dizer, o peça chave da estrutura ele sempre vai começar primeiro ele é o iniciante da estrutura, quer uma teste?, se você colocar um (elif) antes do if veja o que acontece, sim o console dá um erro e com razão pois o (elif) não pode vim antes assim como o (else), e a logica da estrutura é simples passando para o português ficaria assim:

If = Se

Elif = Ou

Else = outro /outra

Como usar funções dos sensors no python:

Ao criar um sensor seja qual ele for você pode acessar quase todas as funcionalidades do próprio para usar ao seu favor como o exemplo acima que usamos o sensor MouseOver no tipo MouseOverAny e isso é sim possível com quase todos os sensores por que quase pois temos que ver o que sim podemos acessar na verdade para isso procure na API o sensor que deseja e o abra e veja o que se pode usar:

```
# Para conectar um sensor a este controlador.  
# "sensorname" é o nome do sensor conforme definido na interface do Blender.  
# + ----- + + ----- +  
# | Sensor "nome do sensor" + - + Python +  
# + ----- + + ----- +  
sens = cont . sensores [ "sensorname" ]  
  
# Para obter uma sequência de todos os sensores:  
sensores = co . sensores
```

Consulte a referência do sensor para obter os métodos disponíveis:

- [KX_MouseFocusSensor](#)
- [KX_NearSensor](#)
- [KX_NetworkMessageSensor](#)
- [KX_RadarSensor](#)
- [KX_RaySensor](#)
- [KX_TouchSensor](#)
- [SCA_DelaySensor](#)
- [SCA_JoystickSensor](#)
- [SCA_KeyboardSensor](#)
- [SCA_MouseSensor](#)
- [SCA_PropertySensor](#)
- [SCA_RandomSensor](#)

No caso do MouserOver mostrado acima está localizado em (KX_MouseFocusSensor) ao abria ele no nome azulado você verá tudo o que pode usar.

KX_MouseFocusSensor (SCA_MouseSensor) ¶

classe base - SCA_MouseSensor

classe bge.types.KX_MouseFocusSensor(SCA_MouseSensor)

O sensor de foco do mouse detecta quando o mouse está sobre o objeto de jogo atual.

O sensor de foco do mouse funciona transformando as coordenadas do mouse do espaço do dispositivo 2D para o espaço 3D e depois transmitindo o raio para longe da câmera.

raySource

A fonte espacial do raio no mundo (a posição da vista).

Tipo: lista (vetor de 3 carros alegóricos)

rayTarget

O alvo do raio no espaço mundial.

Tipo: lista (vetor de 3 carros alegóricos)

rayDirection

O $\text{rayTarget} - \text{raySource}$ normalizado.

Tipo: lista (vetor normalizado de 3 carros alegóricos)

hitObject

o último objeto em que o mouse acabou.

Tipo: KX_GameObject ou nenhum

hitPosition

A posição do espaço no mundo do raio intersecton.

Tipo: lista (vetor de 3 carros alegóricos)

hitNormal

o espaço do mundo normal da face no ponto de interseção.

Tipo: lista (vetor normalizado de 3 carros alegóricos)

hitUV

as coordenadas UV no ponto de interseção.

Tipo: lista (vetor de 2 carros alegóricos)

Se o objeto não tiver mapeamento UV, ele retornará [0, 0].

As coordenadas UV não são normalizadas, elas podem ser <0 ou> 1, dependendo do mapeamento UV.

usePulseFocus

Quando ativado, mover o mouse sobre um objeto diferente gera um pulso. (usado apenas quando a opção do sensor 'Passe o mouse sobre qualquer' está definida).

Tipo: booleano

useXRay

Se ativado, permite que o sensor veja objetos do jogo que não possuem a propriedade ou o material selecionado.

Tipo: booleano

propName

A propriedade ou material que o sensor está procurando.

Tipo: corda

useMaterial

Determina se o sensor está procurando uma propriedade ou material. KX_True = Localizar material; KX_False = Localizar propriedade.

Tipo: booleano

Como pode ver acima todas essas funções você pode usar, mais fica a pergunta como? É simples um sensor praticamente é uma classe, tendo isso em mente use essas funcionalidades como uma função que literalmente é o processo é simples:

Ou chamar um sensor como uma variável (`MouseOver = cont.sensors["MouseOver"]`) você só basta agora usar o nome da variável que você criou seguido do ponto exemplo:

```
MouseOver = cont.sensors["MouseOver"]
```

```
MouseOver."função desejada"
```

Exemplos :

MouseOver.hitObject

MouseOver.hitPosition

MouseOver.hitNormal

E outros, porém debes saber o que a função retorna para teres o resultado que desejas para isso consulte a API do sensor que desejas usar e logo abaixo do nome da função terá algumas informação que mostras o que a função retorna para você usar. Exemplo:

hitPosition

A posição do espaço no mundo do raio intersecton.

Tipo: lista (vetor de 3 carros alegóricos)

Como usar as funções dos actuators no python:

Para usar as funções do (actuators) não é diferente, pois também funcionam como classe, ou seja, também pode usar suas respectivas funções.

Você também pode acessar atuadores vinculados ao controlador

```
# Para conectar um atuador ao controlador:
# + ----- + + ----- +
# + Atuador Python + - + "actuatorname" |
# + ----- + + ----- +
atuador = co . atuadores [ "actuatorname" ]

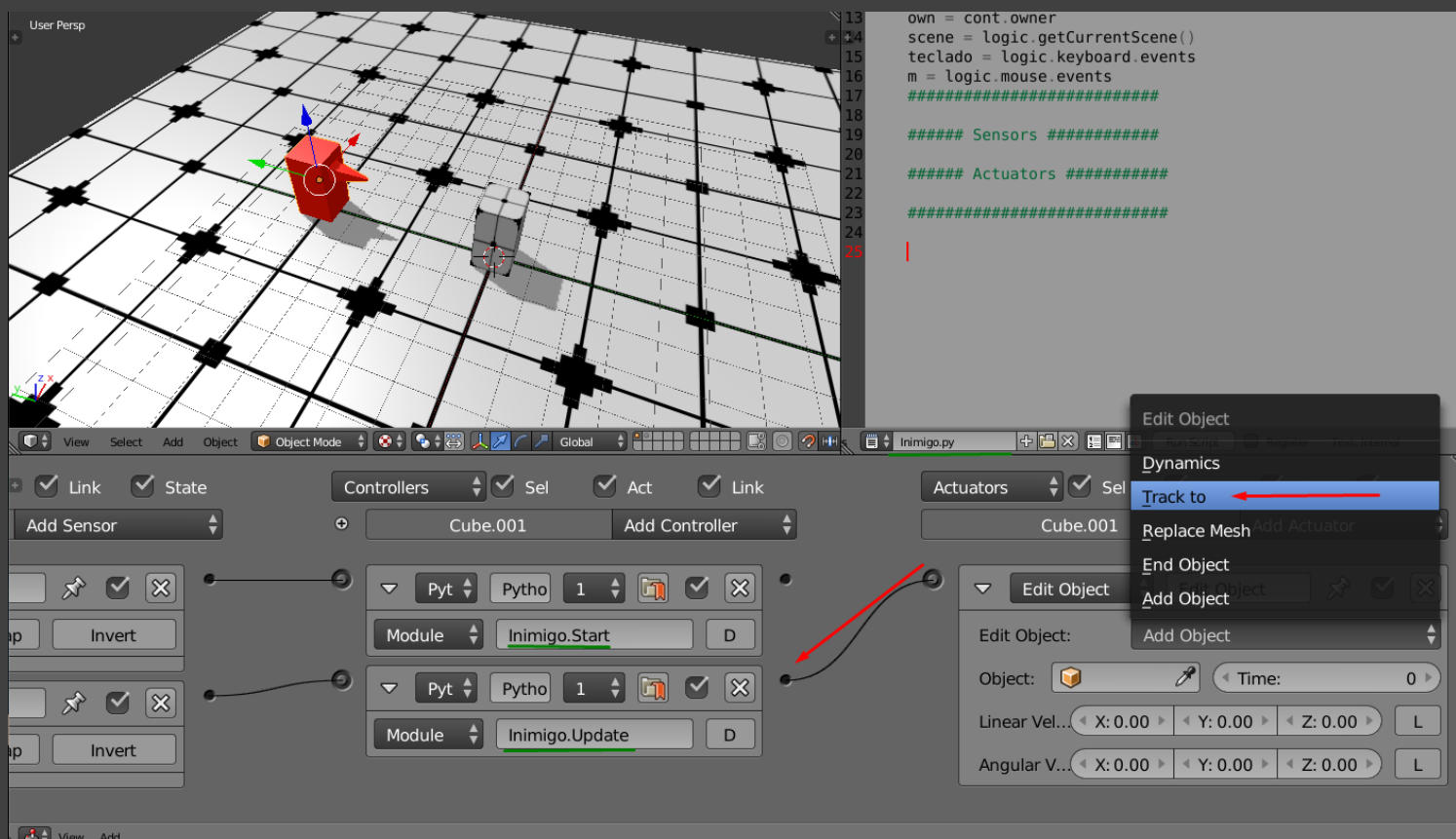
# Ative um
controlador de atuador . ativar ( atuador )
```

Consulte a referência do atuador para obter os métodos disponíveis

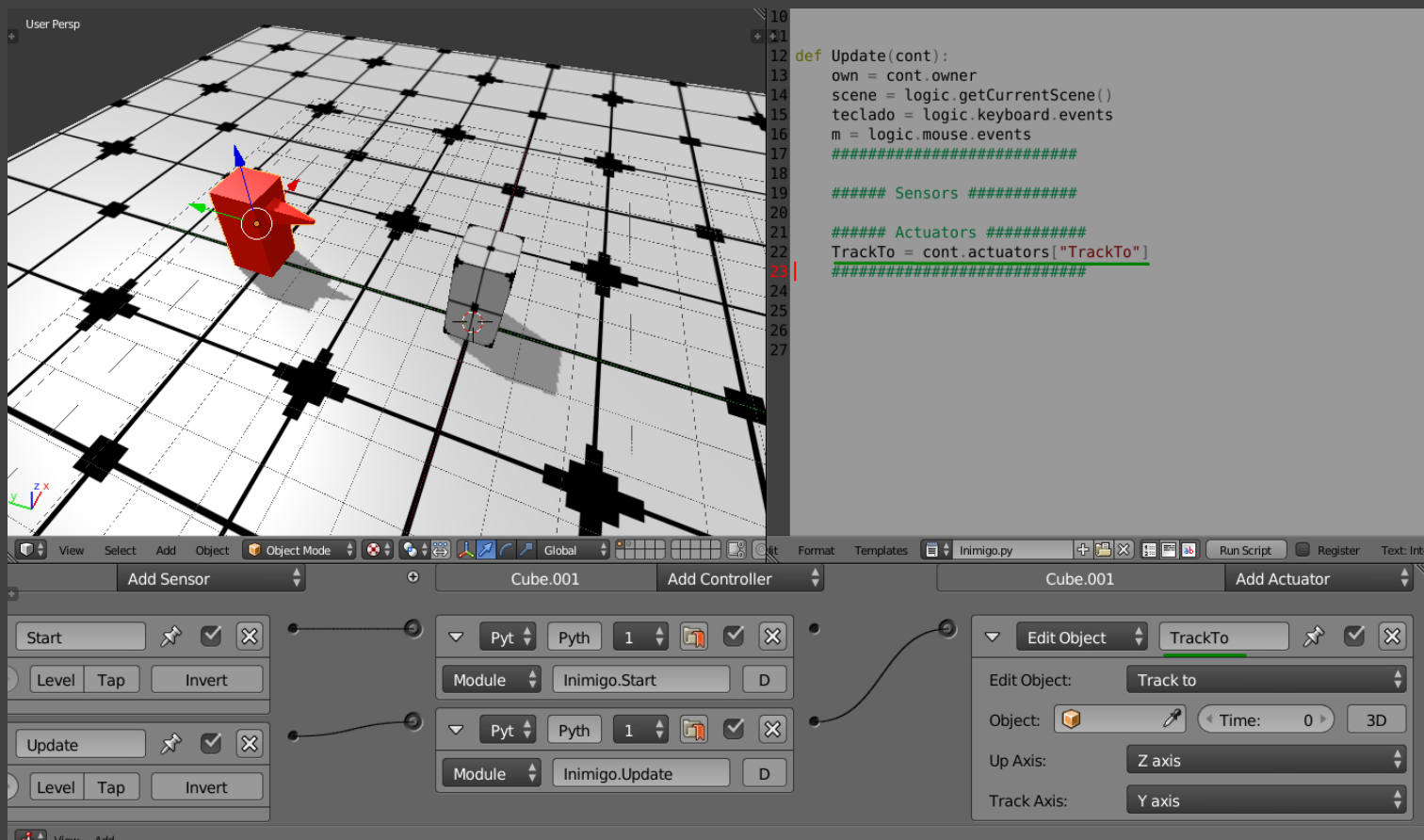
- BL_ActionActuator
- KX_CameraActuator
- KX_ConstraintActuator
- KX_GameActuator
- KX_MouseActuator
- KX_NetworkMessageActuator
- KX_ObjectActuator
- KX_ParentActuator
- KX_SCA_AddObjectActuator
- KX_SCA_DynamicActuator
- KX_SCA_EndObjectActuator
- KX_SCA_ReplaceMeshActuator
- KX_SceneActuator
- KX_SoundActuator
- KX_StateActuator
- KX_TrackToActuator
- KX_VisibilityActuator
- SCA_2DFilterActuator
- SCA_PropertyActuator
- SCA_RandomActuator

E para exemplo vamos usar o (KX_TrackToActuator) e eu quero fazer com que o inimigo por acaso sempre olhe para o meu personagem diretamente como poderemos, fazer vejamos então:

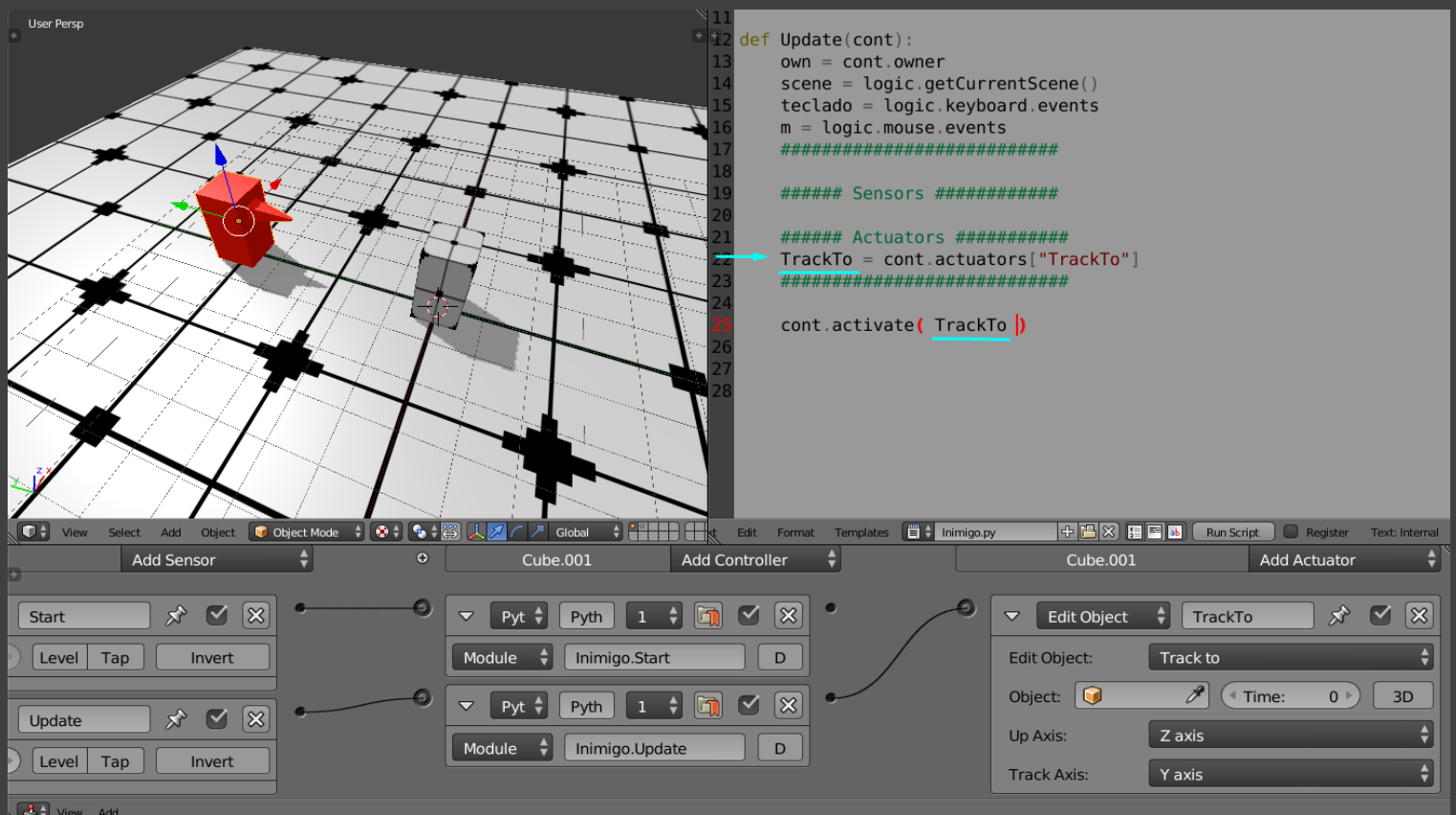
Vamos criar um actuator (Edit Object) e colocar no tipo (Track To) como a imagem abaixo :



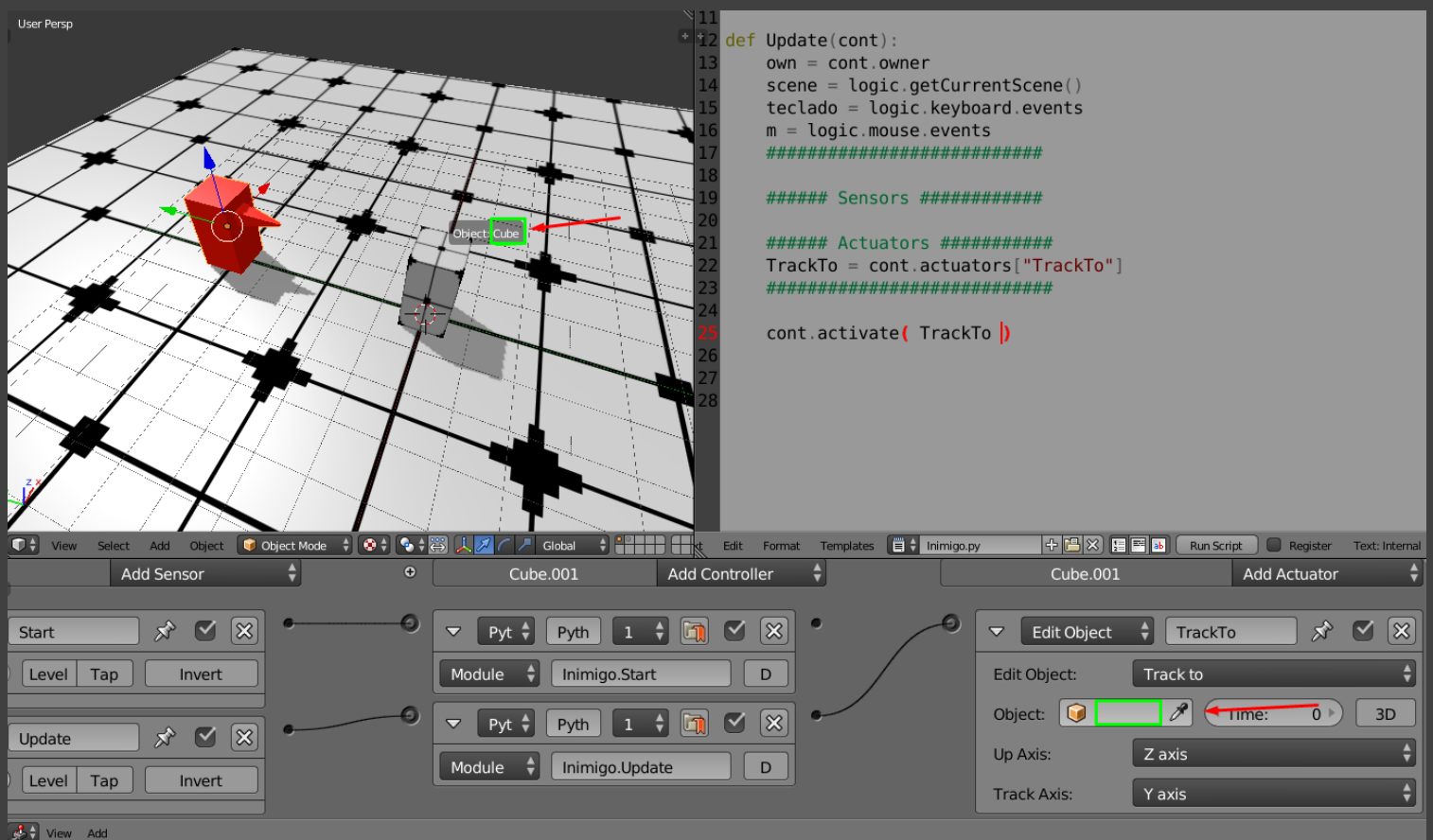
Como mostra na imagem muda para o tipo (Track To) e ligue o actuador no controlador do modulo Update que é o seu script do inimigo ,depois disso feito vamos chamar o actuador no script , desse jeito :

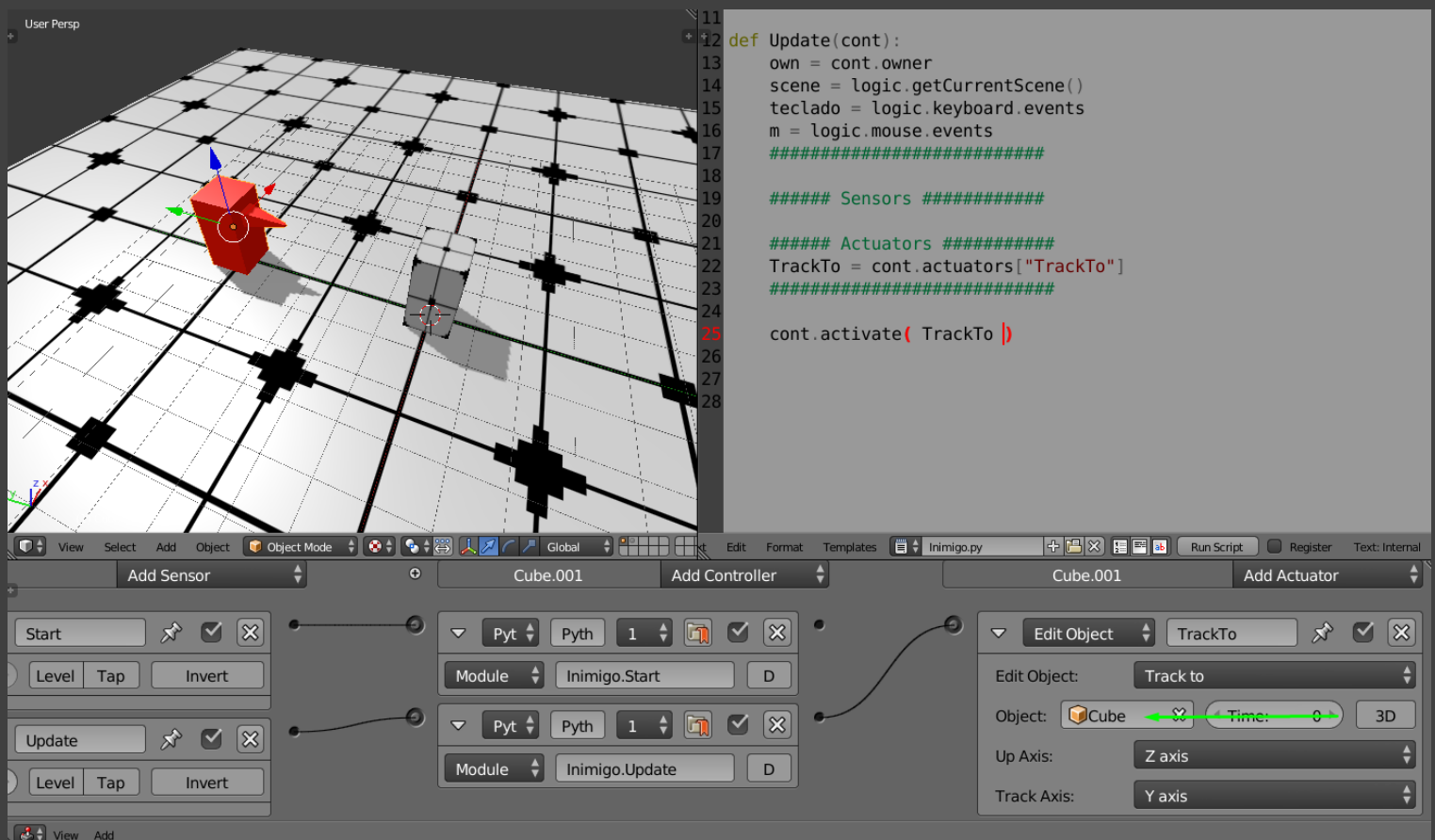


Com isso pronto agora vamos ativar o actuador por python mesmo, lembrando que ele deve ser chamado no Update pois se chamar ele no Start ele só vai ser ativado apenas uma vez, ou seja, no inicio do jogo e pronto.



(cont.activate(Nome da variável do actuators)) esse é o método para ativar um actuador, com isso se você iniciar o jogo, não vai acontecer nada pois você não defini-o qual objeto que se deve olhar, obvio como ele vai saber o que você quer se você não disser !, Vejamos como faremos isso:





Quando você clicar com o botão esquerdo do mouse no ícone de conta gotas você pode transferir diretamente o nome do objeto desejado, “Boa beleza isso é simples mais como posso fazer isso diretamente no python?”. É muito simples também veja, no actuator tens se o nome (Object) do lado do campo onde se colocar o nome do objeto no python você vai usar esse mesmo nome para chamar o objeto desejado por python também é sim ples porem lembre-se de escrever em letras minúsculas mesmo como está na API que é o melhor lugar para ver e pegar as funções desejadas.

KX_TrackToActuator (SCA_IActuator)


classe base - SCA_IActuator

classe bge.types.KX_TrackToActuator(SCA_IActuator)

Edite o atuador de objetos no modo Track To.

Aviso: Atuadores de Rastrear até serão ignorados se, no início do jogo, o objeto para rastrear for inválido. Isso irá gerar um aviso no console:

```
GameObject 'Name' nenhum objeto no EditObjectActuator 'ActuatorName'
```

object  o objeto que esse atuador rastreia.

Tipo: KX_GameObject ou nenhum

time o tempo nos quadros com os quais atrasar o movimento de rastreamento.

Tipo: inteiro

use3D o movimento de rastreamento para usar 3D.

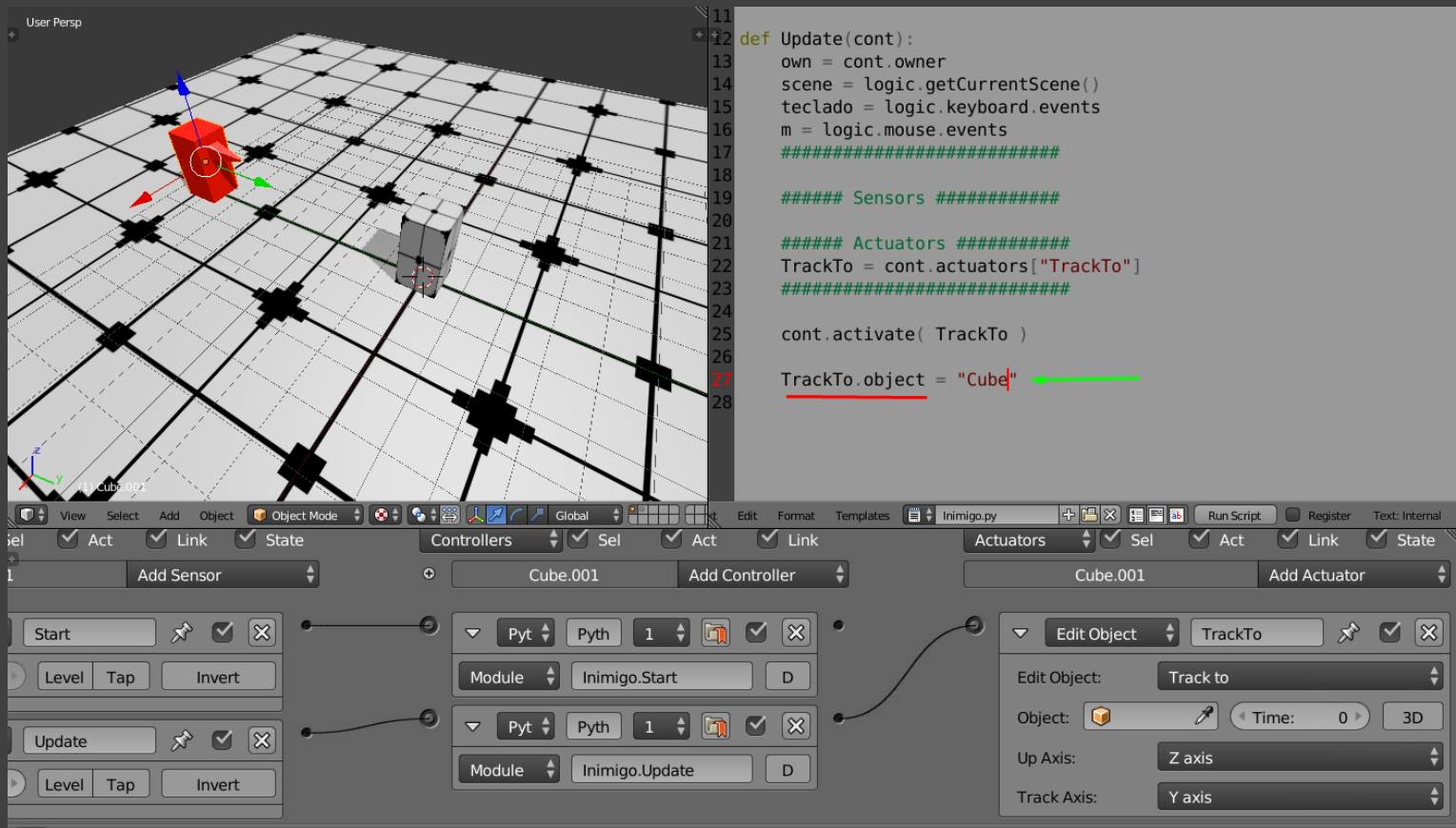
Tipo: booleano

upAxis O eixo que aponta para cima.

Tipo: número inteiro de 0 a 2

- KX_TRACK_UPAXIS_POS_X
- KX_TRACK_UPAXIS_POS_Y
- KX_TRACK_UPAXIS_POS_Z

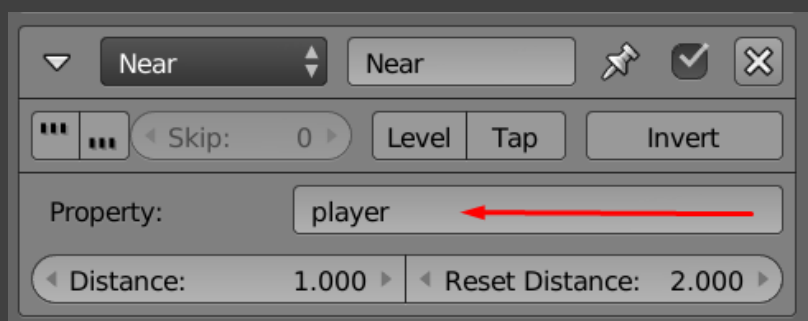
Com isso basta fazer desse jeito (Nome da variável do `actuator.objetc` = " String nome do objeto " ou o próprio objeto pegado em cena coisa que veremos mais adiante) então em python fica assim :



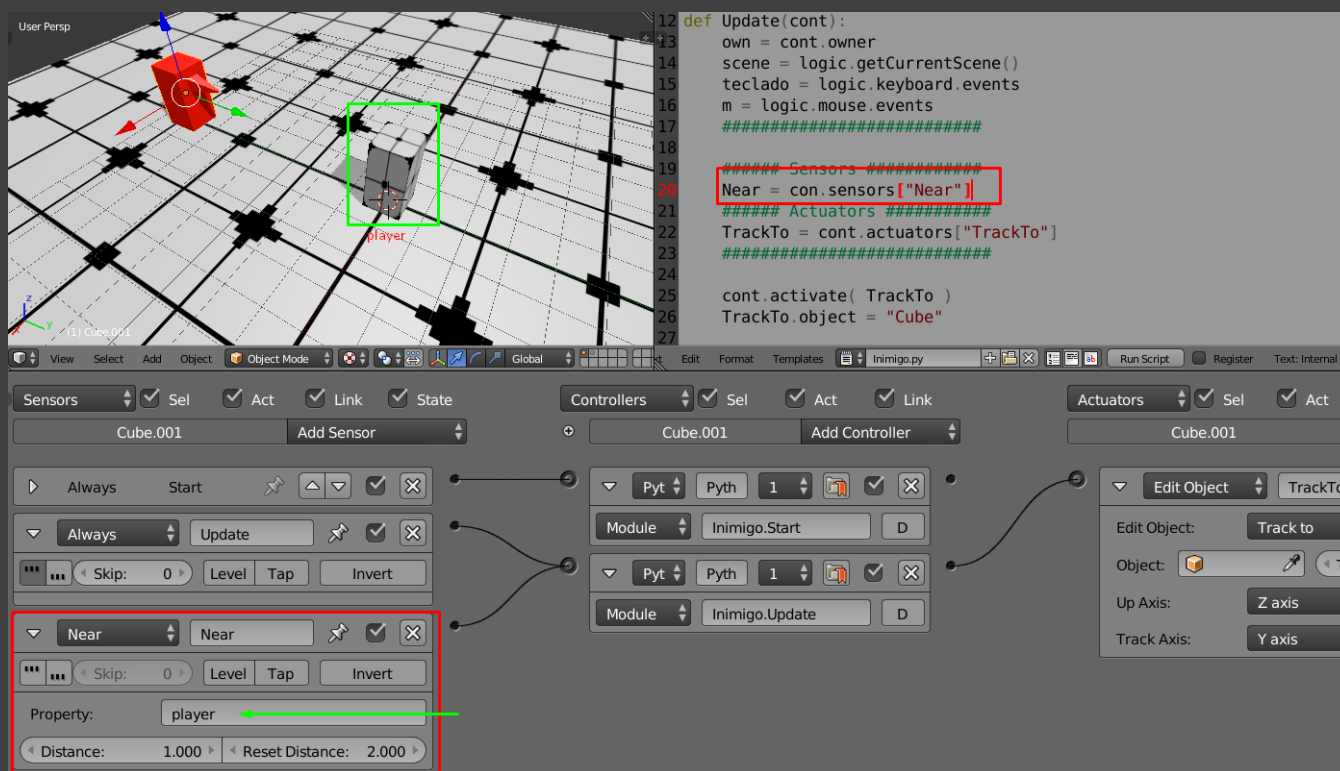
E agora pode iniciar o jogo e veras que o inimigo aqui no caso o cubo vermelho estará sempre apontado com a frente para o personagem, se por acaso o seu ficar do lado a contrario, simplesmente o gire e aplique a rotação modificada e pronto está feito um(`TrackTo`) simples.

Agora que quero que quando o meu personagem cheque á uma distancia relativamente perto o meu inimigo ai sim olhe para ele, e quando se afastar ele pare de seguir. Bom isso é fácil usaremos um pouco mais do que aprendemos até aqui, para isso precisaremos de um sensor pode ser (Near) e criar as condições no python, vamos lá.

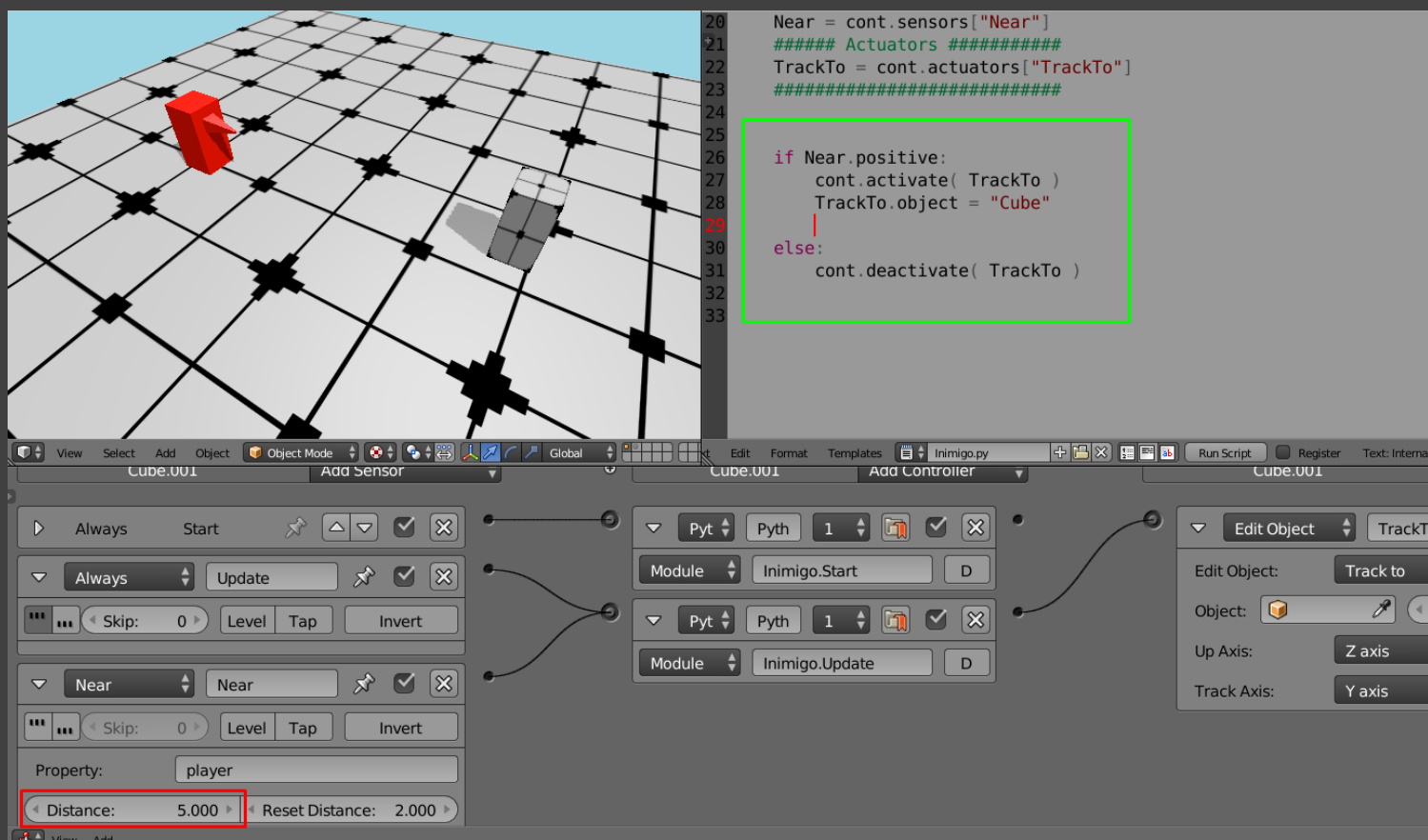
Crie um sensor do tipo (Near) seu nome aqui vai ficar assim mesmo, mais se lembre, que você pode mudar os nome dos sensores e actuators para o que quiser, só deixo uma dica coloque nomes não muito complicados e sem sentido isso deixará seu condigo um pouco bagunçado e talvez incompressível até para você mesmo alguma hora hahaha!.



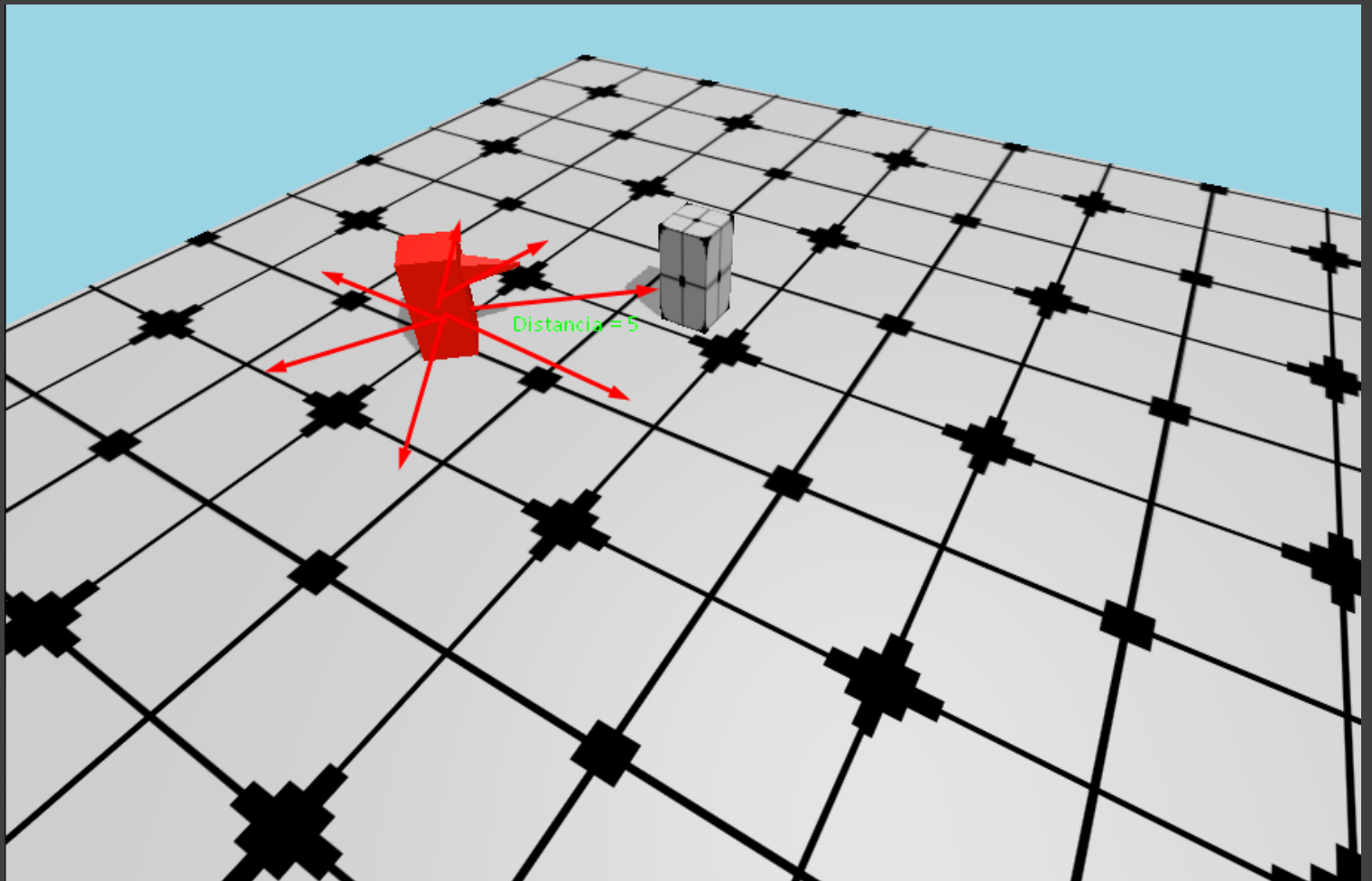
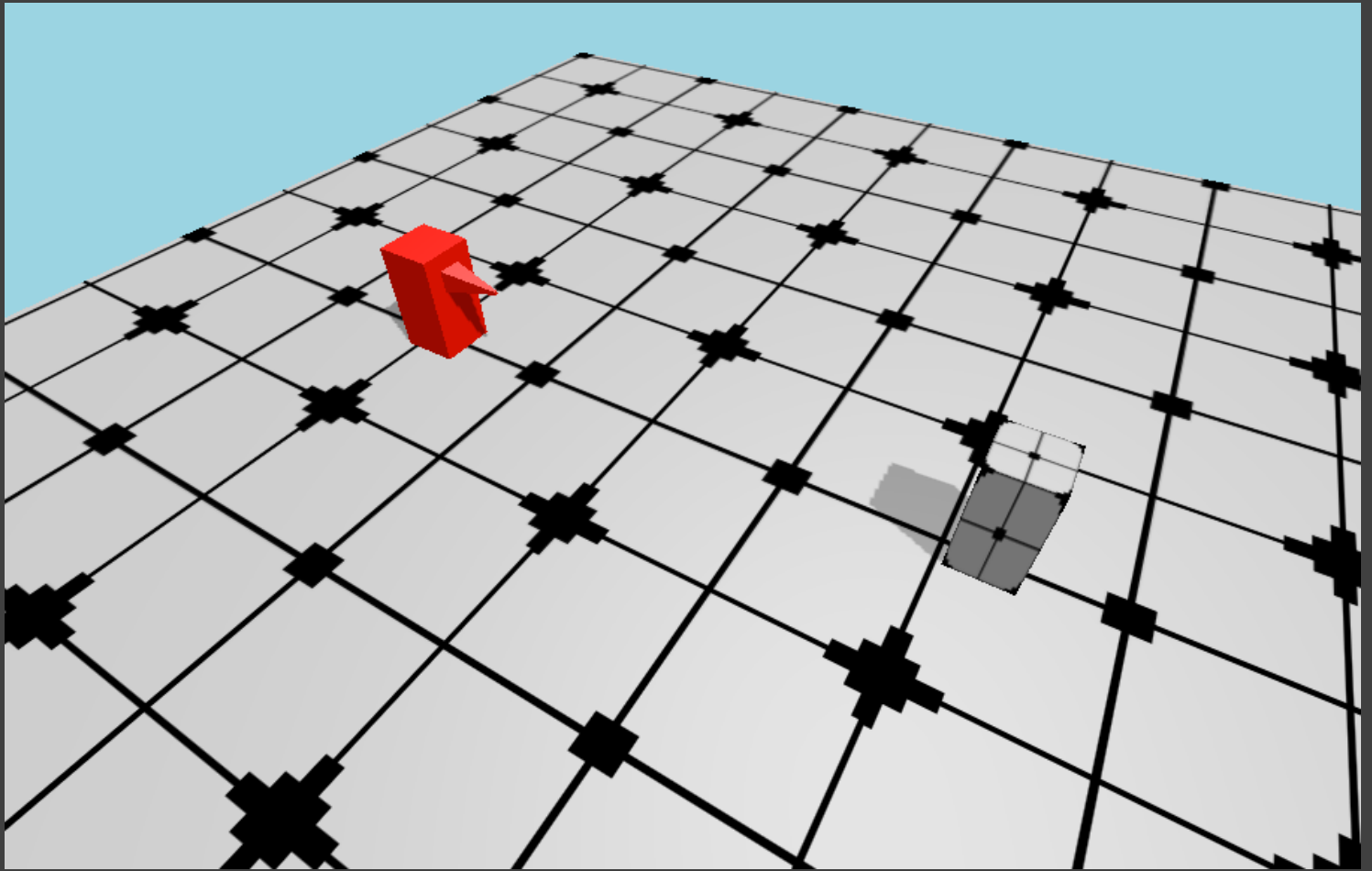
(`player`) será uma propriedade que você deve criar no outro objeto que para o sensor funcionar, lembrando qualquer nome certo.



E ao chamar o sensor vamos para as estruturas de condições como vimos nas paginas passadas, veja a imagem, sua estrutura de condição dever ficar assim para funcionar como está explicado aqui certo:



Como pode ver coloquei (5.000) na distancia no sensor que pode ser qualquer uma, veja as imagens abaixo quando o personagem está em uma distancia fora de 5, ou seja, maior que 5 ele não vai detectar o personagem porem quando ele estiver em uma distancia menor que 5 ai sim ele detecta o personagem e faz o que lhe foi programado .




Vamos entender como funciona o que programamos agora a estrutura de condição que fizemos certo.

```
def Update(cont):
    own = cont.owner
    scene = logic.getCurrentScene()
    teclado = logic.keyboard.events
    m = logic.mouse.events
    #####

    ##### Sensors #####
    Near = cont.sensors["Near"]
    ##### Actuators #####
    TrackTo = cont.actuators["TrackTo"]
    #####

    if Near.positive:
        cont.activate( TrackTo )
        TrackTo.object = "Cube"
    else:
        cont.deactivate( TrackTo )
```



Com (if Near.positive:) eu estou dizendo que, quando o sensor Near for positivo, ou seja, se ele realmente for iniciado/funcionar, reproduza tudo que estiver depois e abaixo do (:), o que significa, tudo que estiver dentro do escopo da condição.

E logo abaixo coloquei a ativação do meu TrackTo e em seguida o uso de sua função e mais a baixo o (else:) que server para que quando o sensor não for positivo ele simplesmente reproduza tudo que estiver dentro do seu escopo ou seja desative o actuator TrackTo (cont.deactivate(TrackTo)), simples não agora pratique e use outros actuadores e sensors isso servirar muito de aprendizado, lembrando que se quiser usar as funções dos seus sensores e actuadores de sempre uma passada pela a API do seu blender nesse caso desse Ebook estamos usando o blender 2.79, ou seja, sua API isso vale para a Upbge porém algumas coisas são sim diferente mais não os conceitos certo, bom estudos!.

