2020

BGE + PYTHON

Edinaldo Cicero

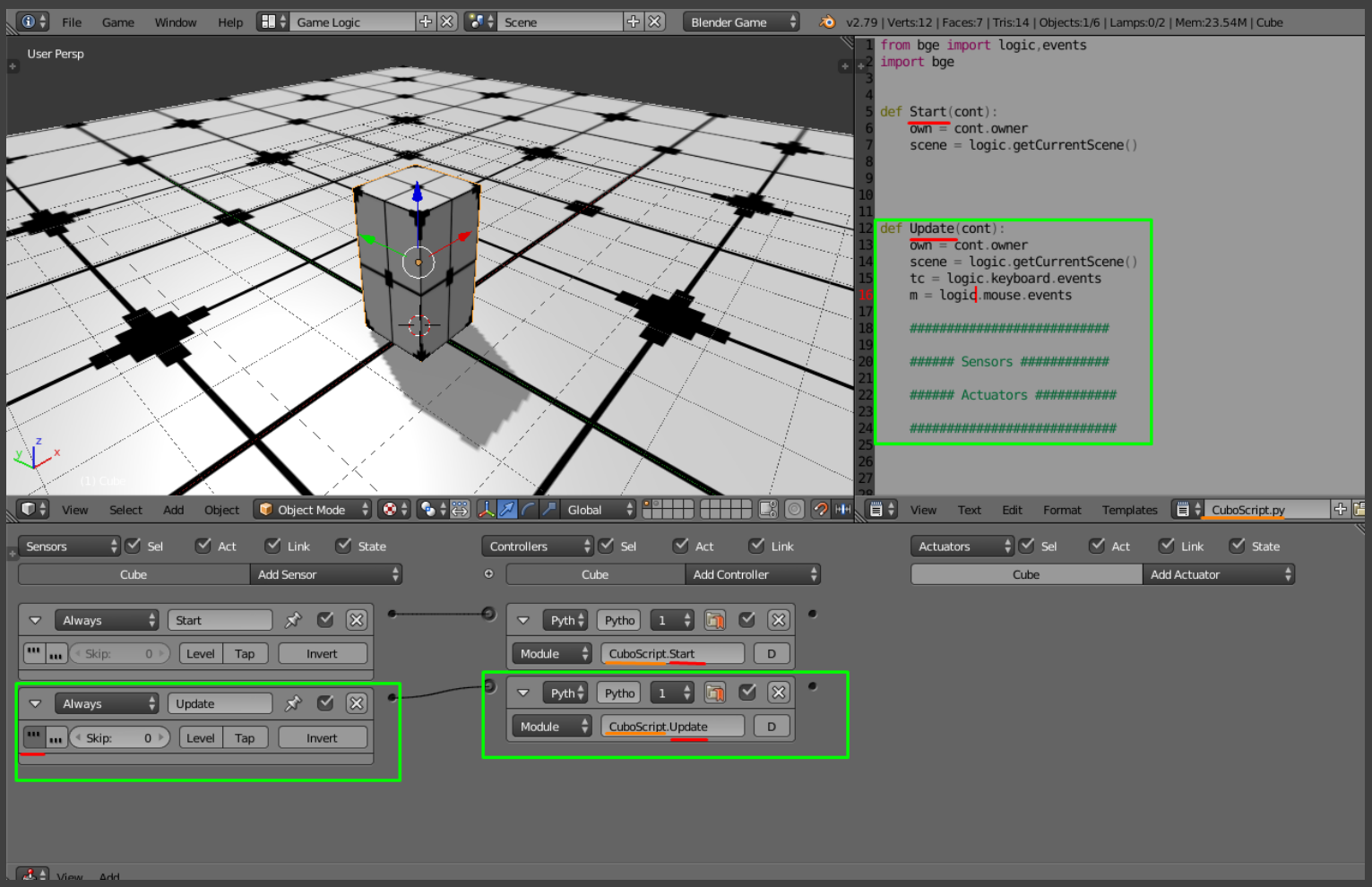
ÁtomoGames.

28/07/2020

BGE + PYTHON

Começando com Funções para Modulos :

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



Para chamar o script pelo Modulo vc 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 logic bricks de controlador do objeto ou seja o que está em verde com isso vc tem acesso ao sensores e actuators do objetos que estiverem ligados nesse controlador .

Own ou (owner = cont.owner) aqui corresponte ao próprio objeto proprietário do script com isso em mão vc 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 vc 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.

API BLENDER 2.79 :

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 vc poderá perceber que o que estiver detro 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 obj e o que é Vetor2 E 3 :

Para movimentar ou Rotacionar um obj no universo do blender existe duas classes Vetoriais :

1 = owner.applyMovent([vetor3],Boo)

owner.applyMovent([X,Y,Z],Boo)

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 geograficas 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 desejas que o obj ande ou rotacione , tenha em mente que se vc colocar (- ou +) na frente do valor desejado isso fará mudar de direção pois isso está baseado nos paramentros 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 interiros são números exatos e corresponde a mais espaços para o objeto se deslocar pois o obj não andar realmente claro! , mais sim se deslocar de acordo com o valor digitado , lembrou física muito bem !.

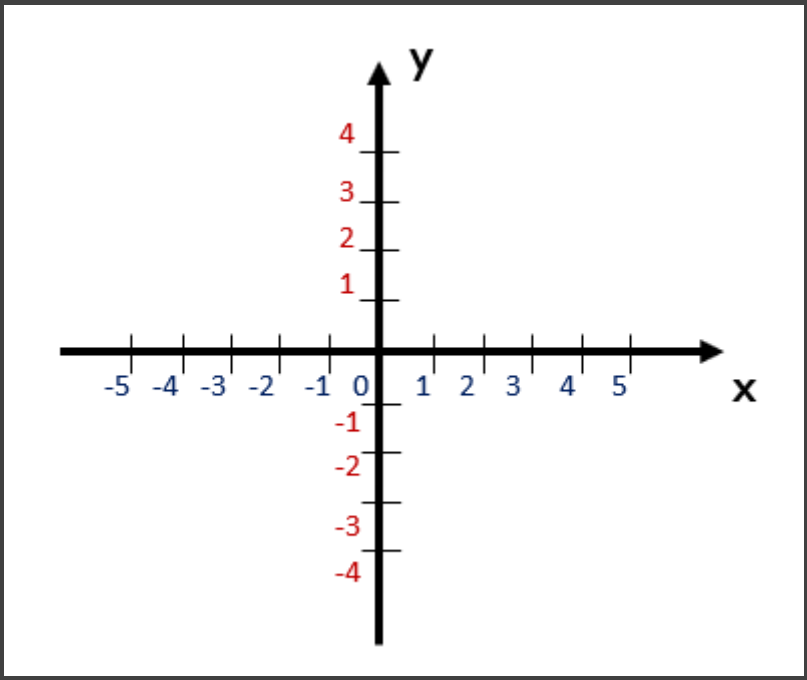
X = Largura

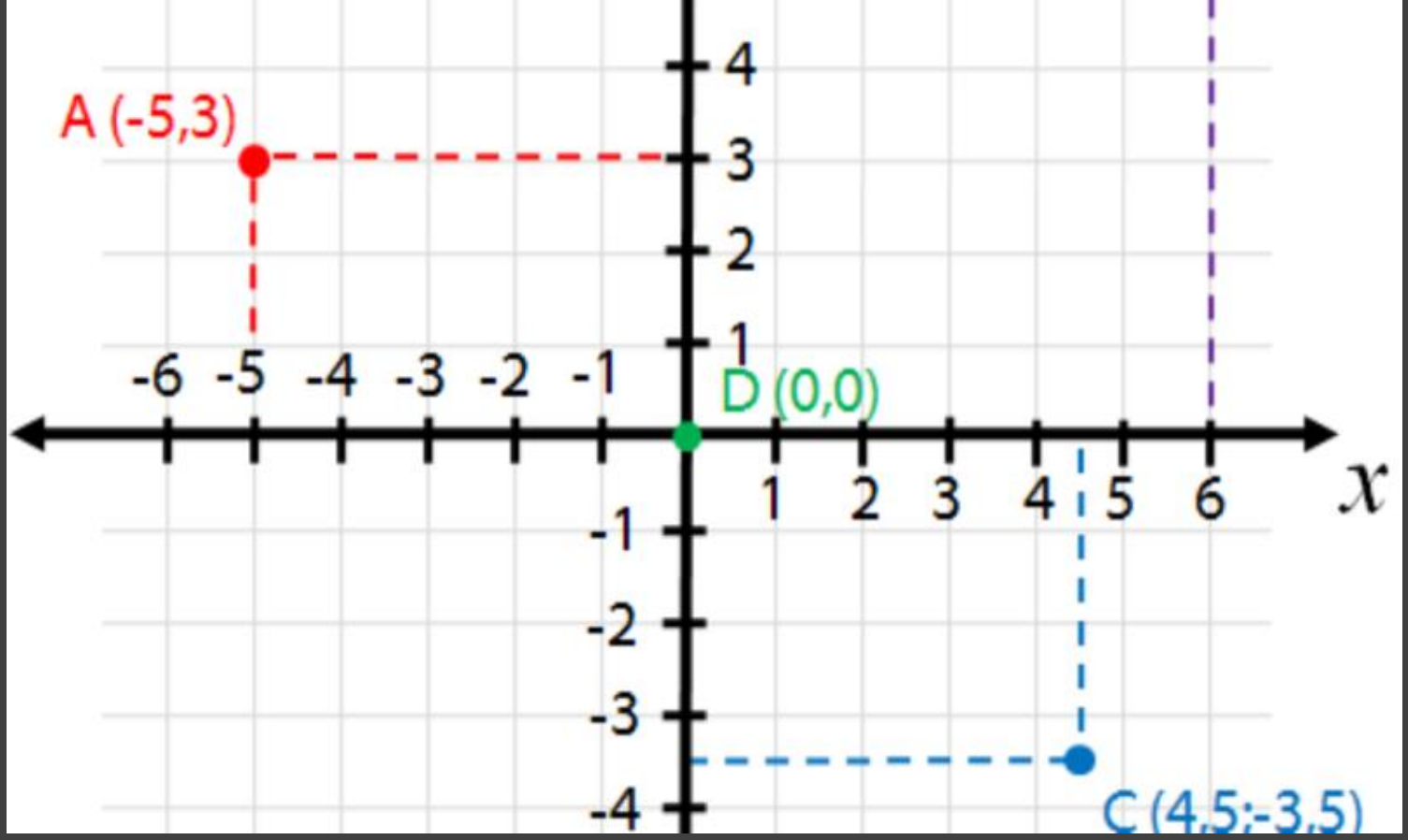
Y = Altura

Z = Profundidade

Vector2 pra Games 2D

Vector3 pra Games 3D





Como 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 variavel com o nome que quiser mais que se torne fácil de lembrar claro e faça desse jeito :

Teclado = logic.keyboard.events

Com isso pronto vamos para 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 :

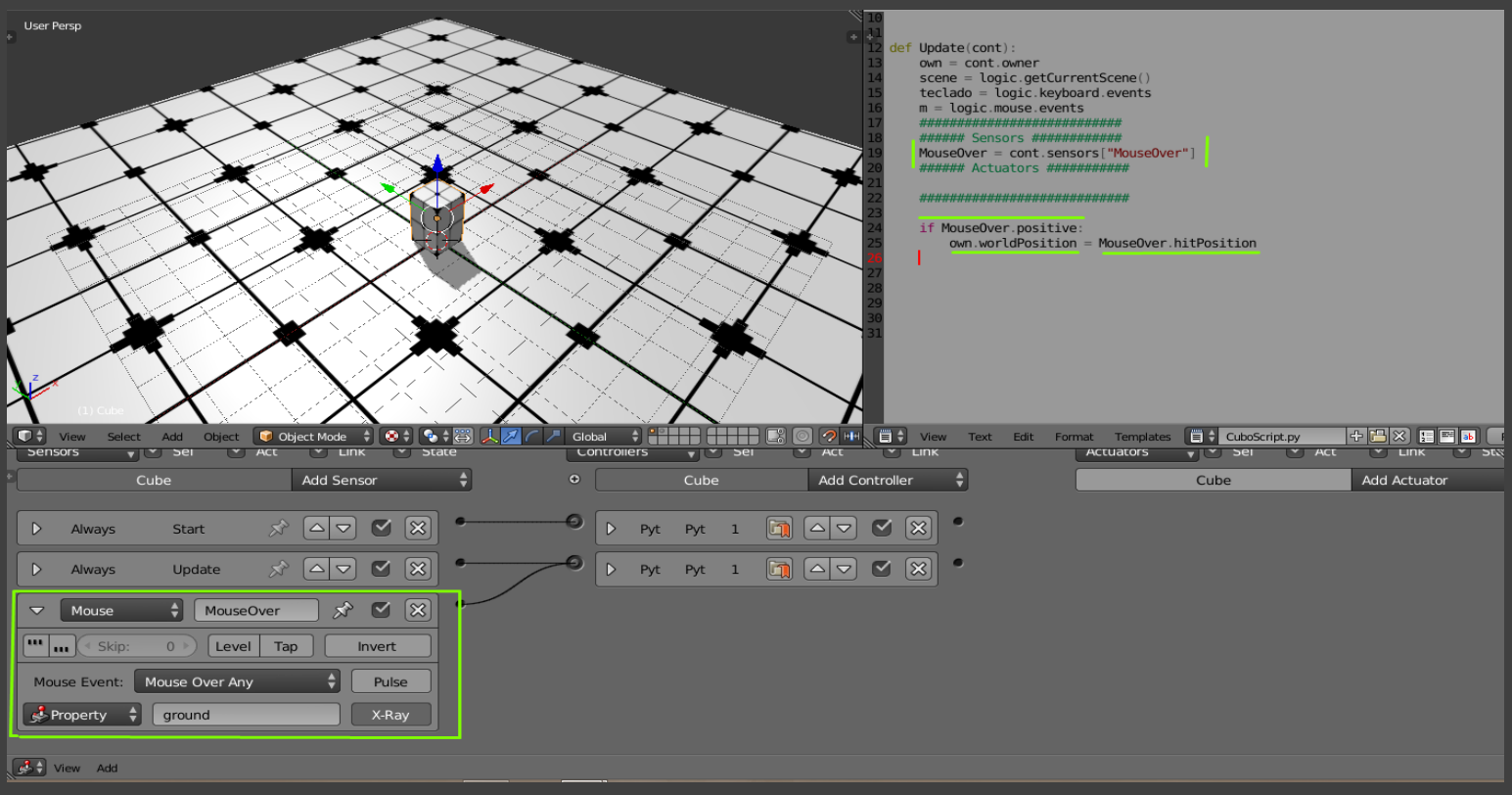
```
def Update(cont):
    own = cont.owner
    scene = logic.getCurrentScene()
    teclado = logic.keyboard.events
    m = logic.mouse.events
    ############################
    ###### Sensors ############
    ###### Actuators ###########
    ############################

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

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

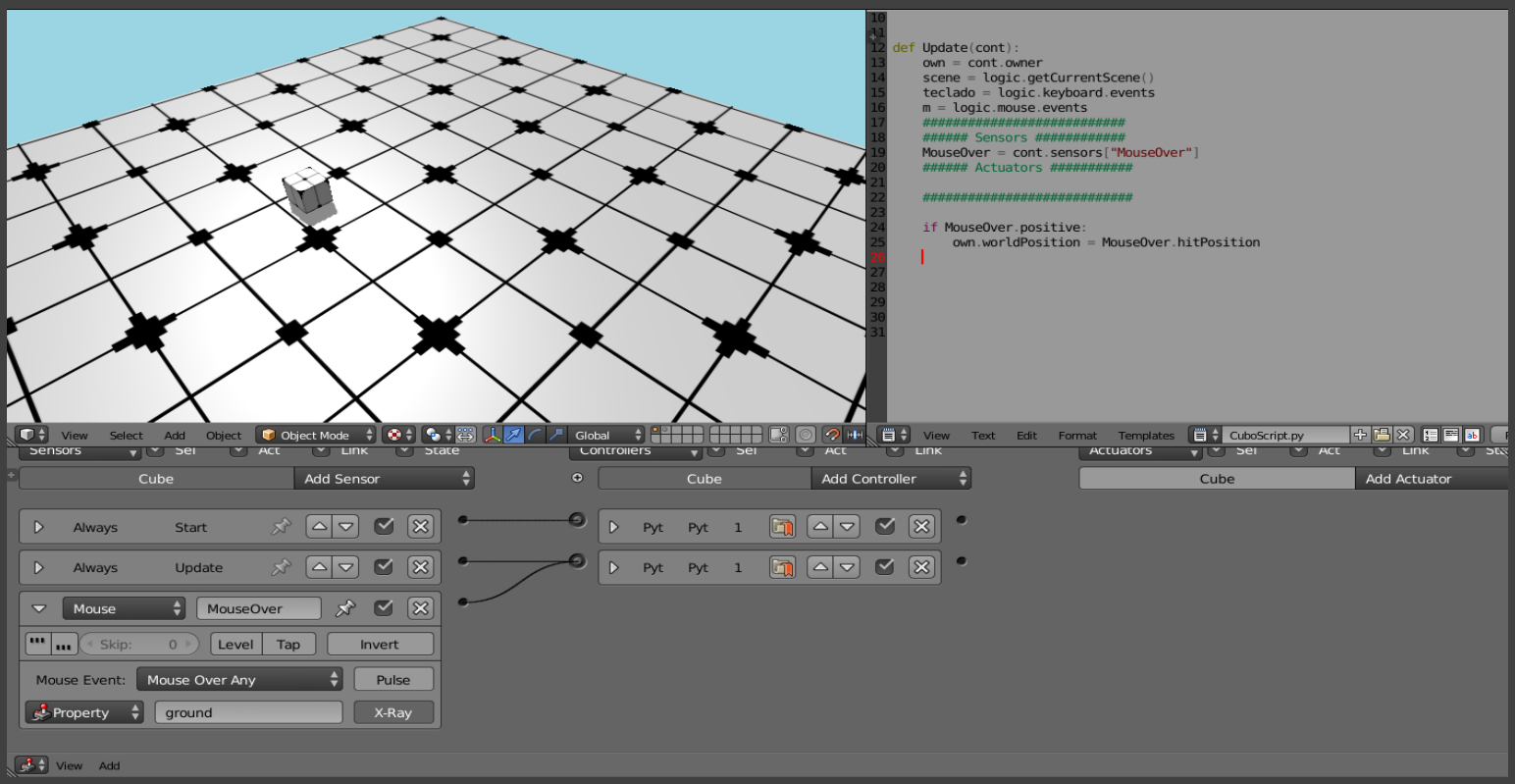
Com isso temos uma condição que quando a tecla W for apertada apois os dois pontos vc aperta inter e entrará dentro do escopo da condição que será abaixo com isso vc poderár fazer ações com o obj colocando até o próprio own.applyMovement([x,y,z],True) e no lugar dos ([X,Y,Z]) coloque o valor que desejas que o obj ande ou rotacione !..

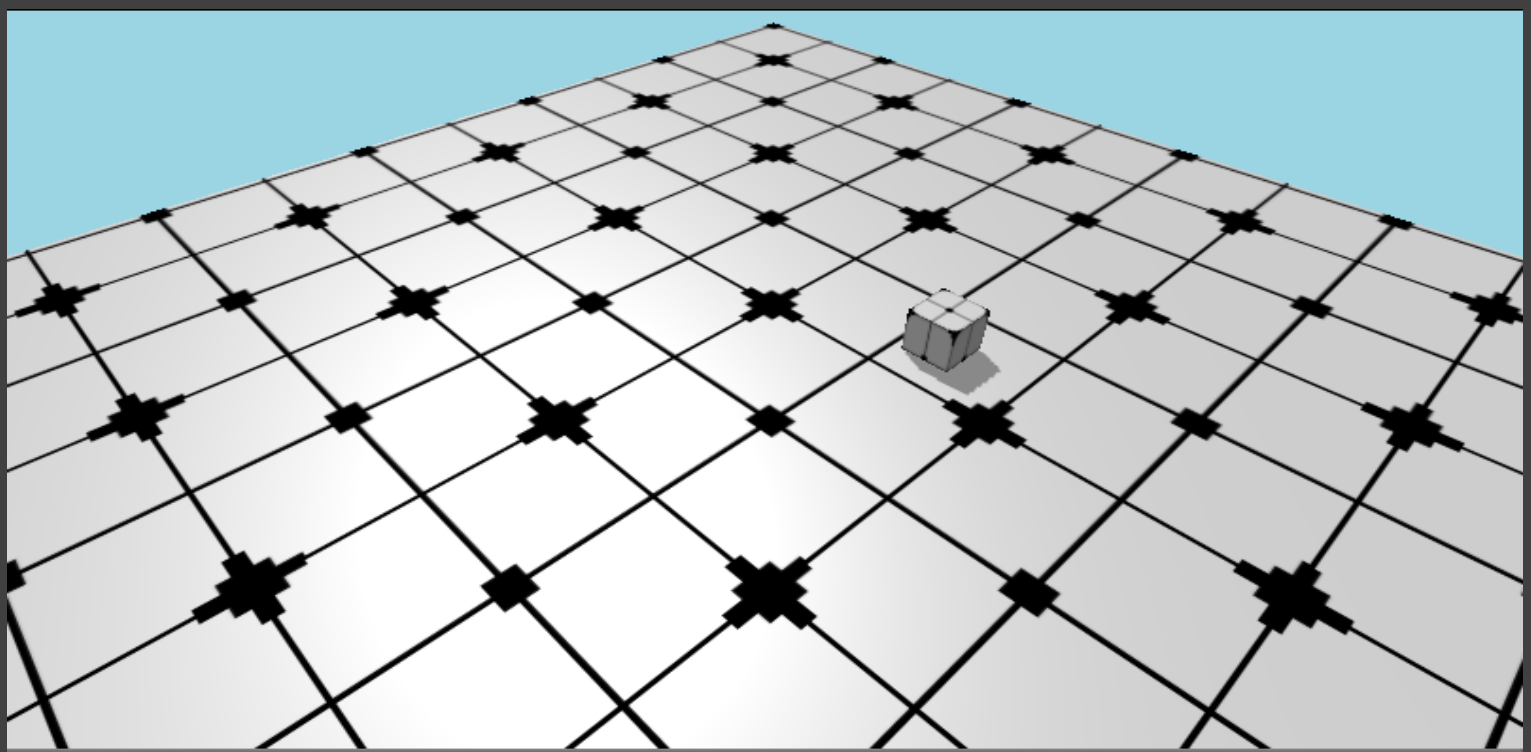
Como fazer um objeto seguir a posição do mouse:



Para um objeto seguir a posição do mouse é preciso ter um sensor do tipo (Mouse Over Any) ,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 a câmera sem ordem.

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

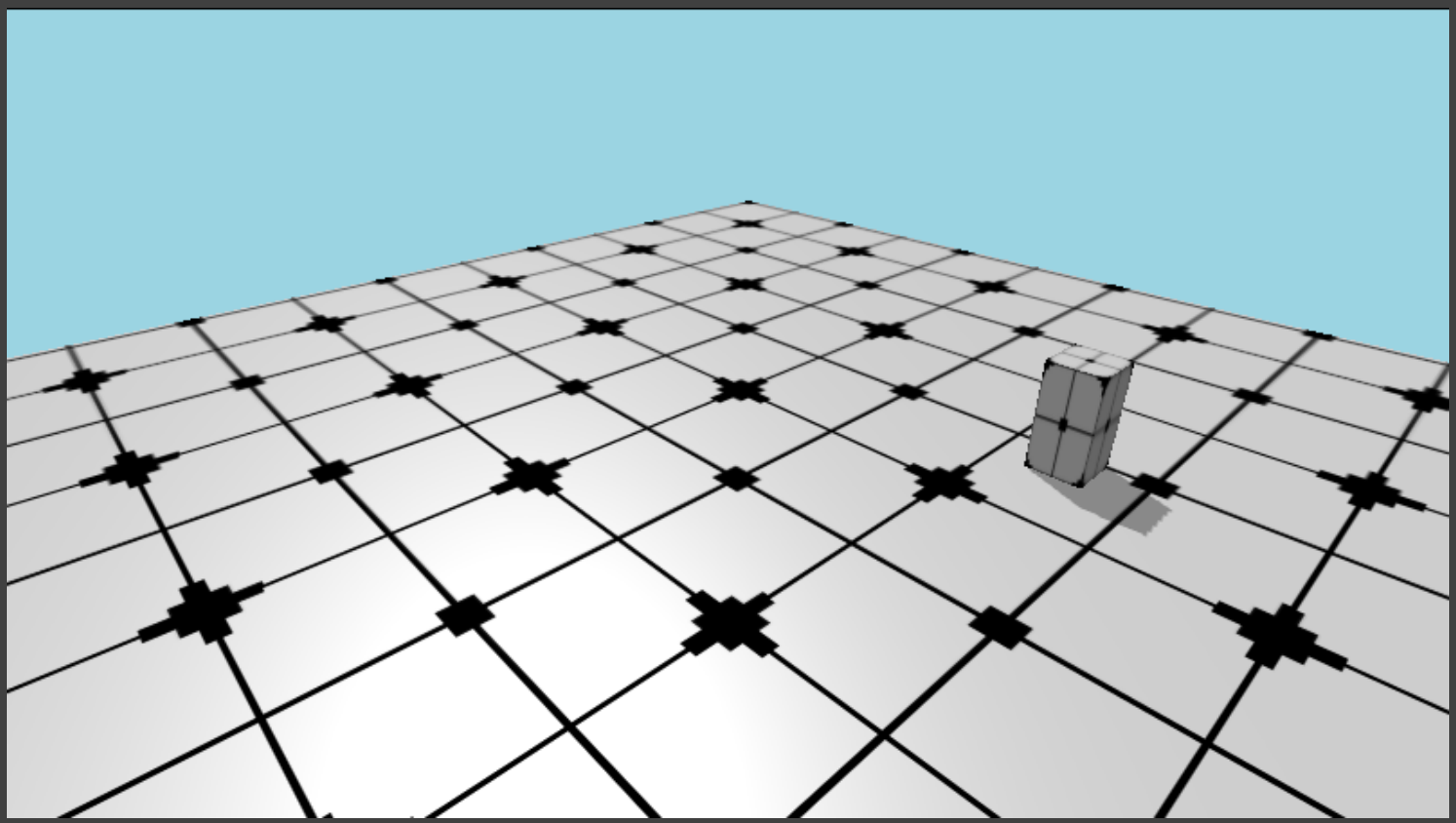


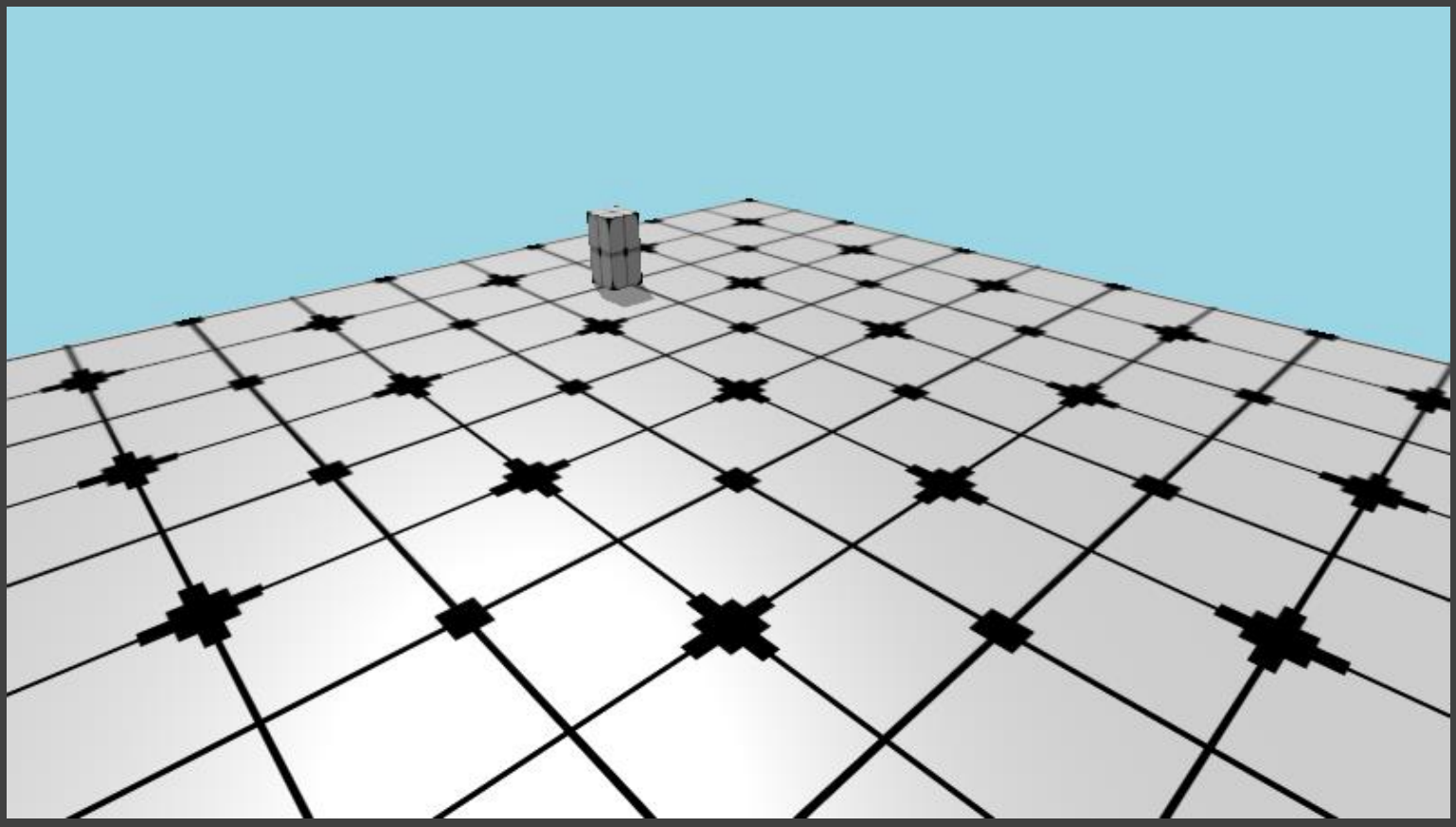


Mais note que o objeto está pela metade pois o resto está entrando no plano para resolver isso é muito simples vc só precisar usar dois vetores em vez dos três ,ou seja , (X,Y) e como fazer isso é simples :

```
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
27
28
29
30
31
32
```

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





Como usar funções dos sensors no python:

#Ao