

UPBGE

Components

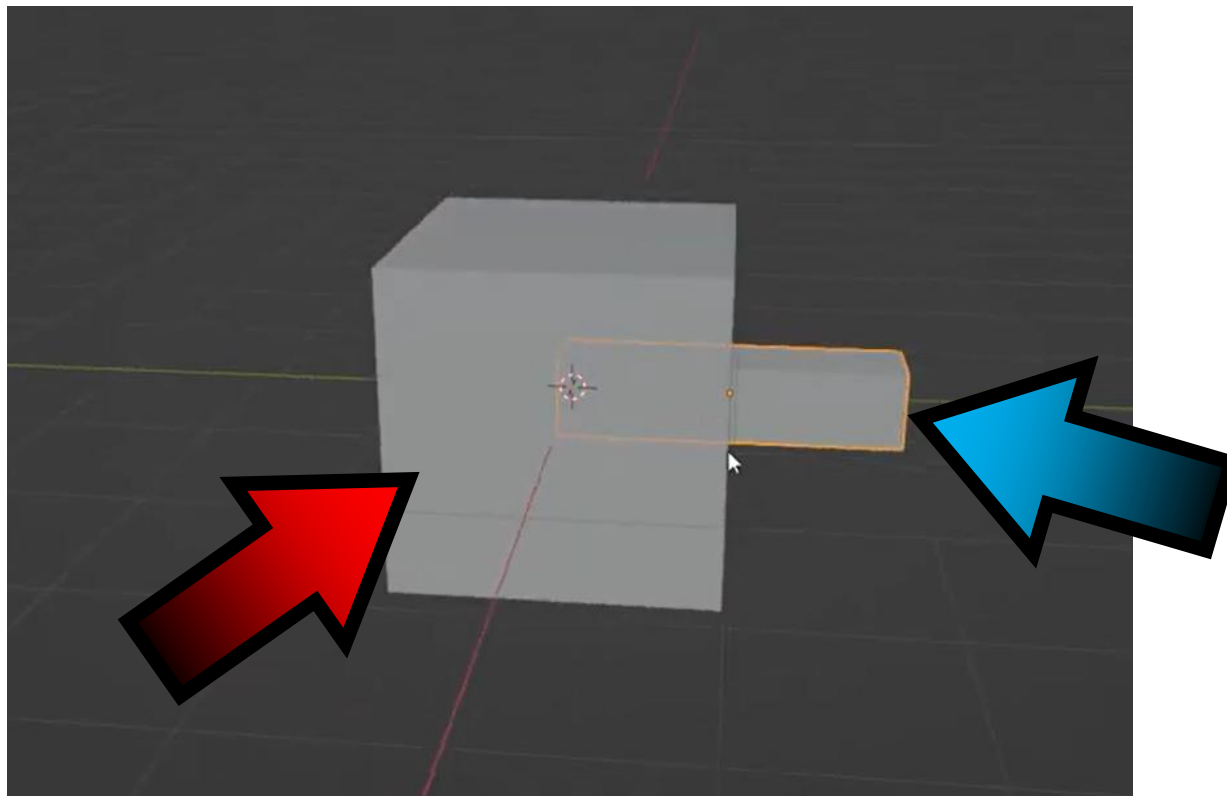


**Co-funded by
the European Union**

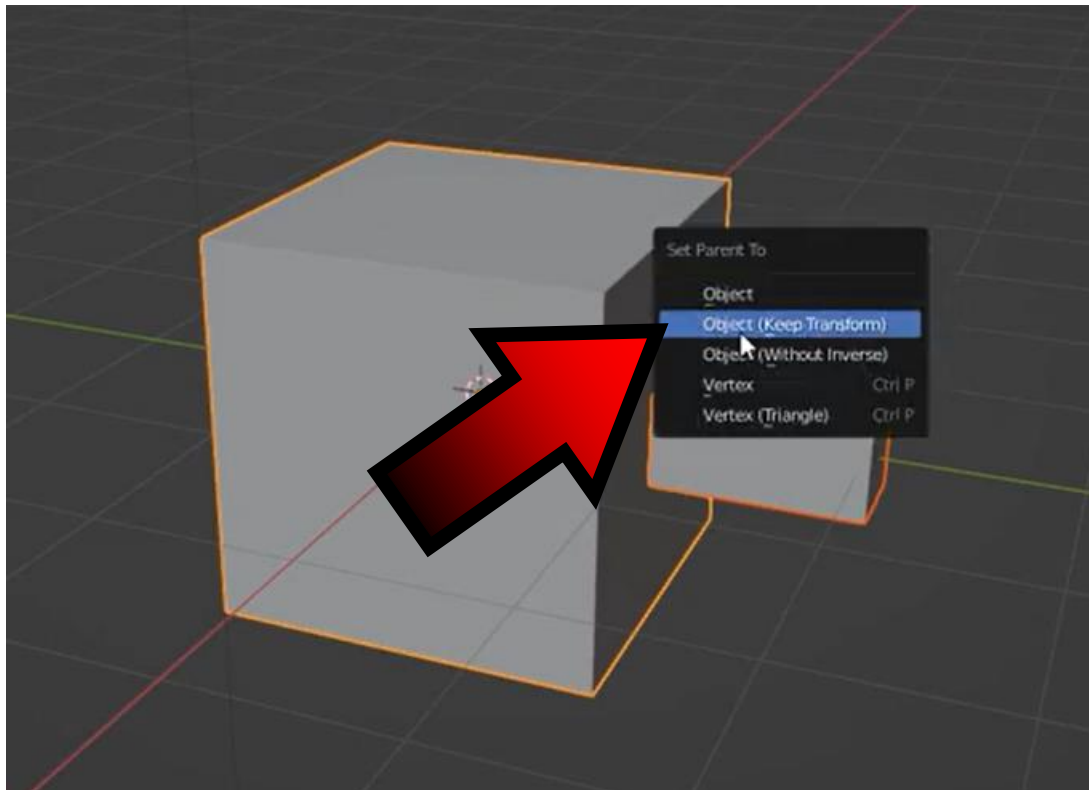


2024-1-PL01-KA220-VET-000243150

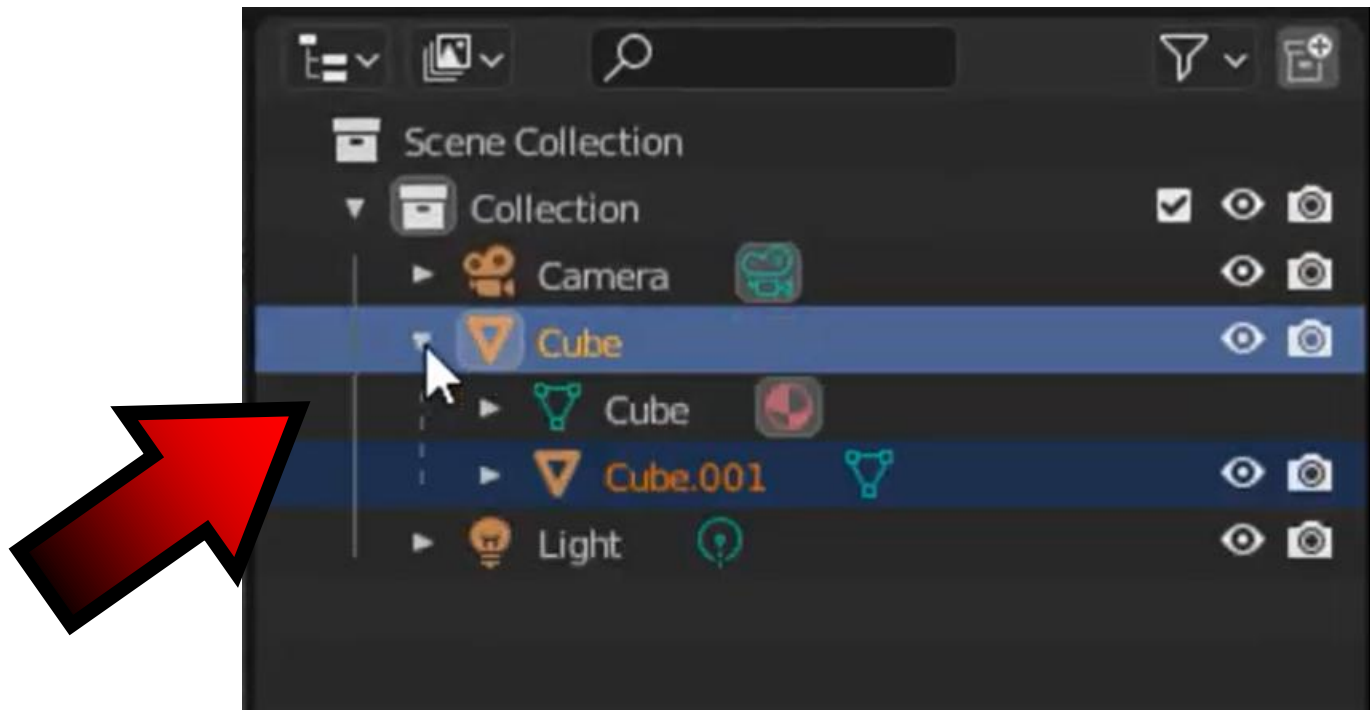
CREATE A MODEL AS SHOWN



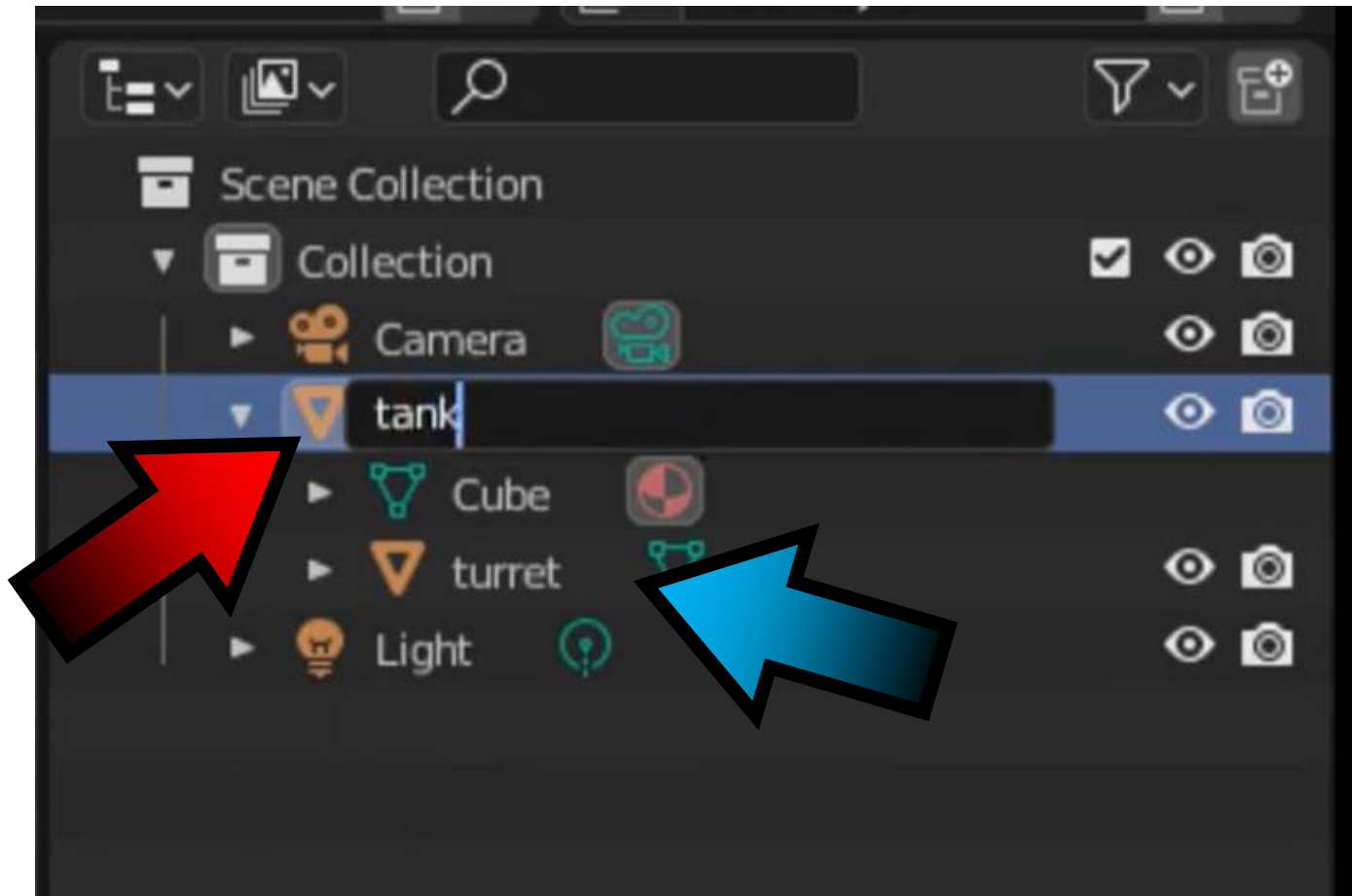
CONNECT WIDTH **CTRL+P**



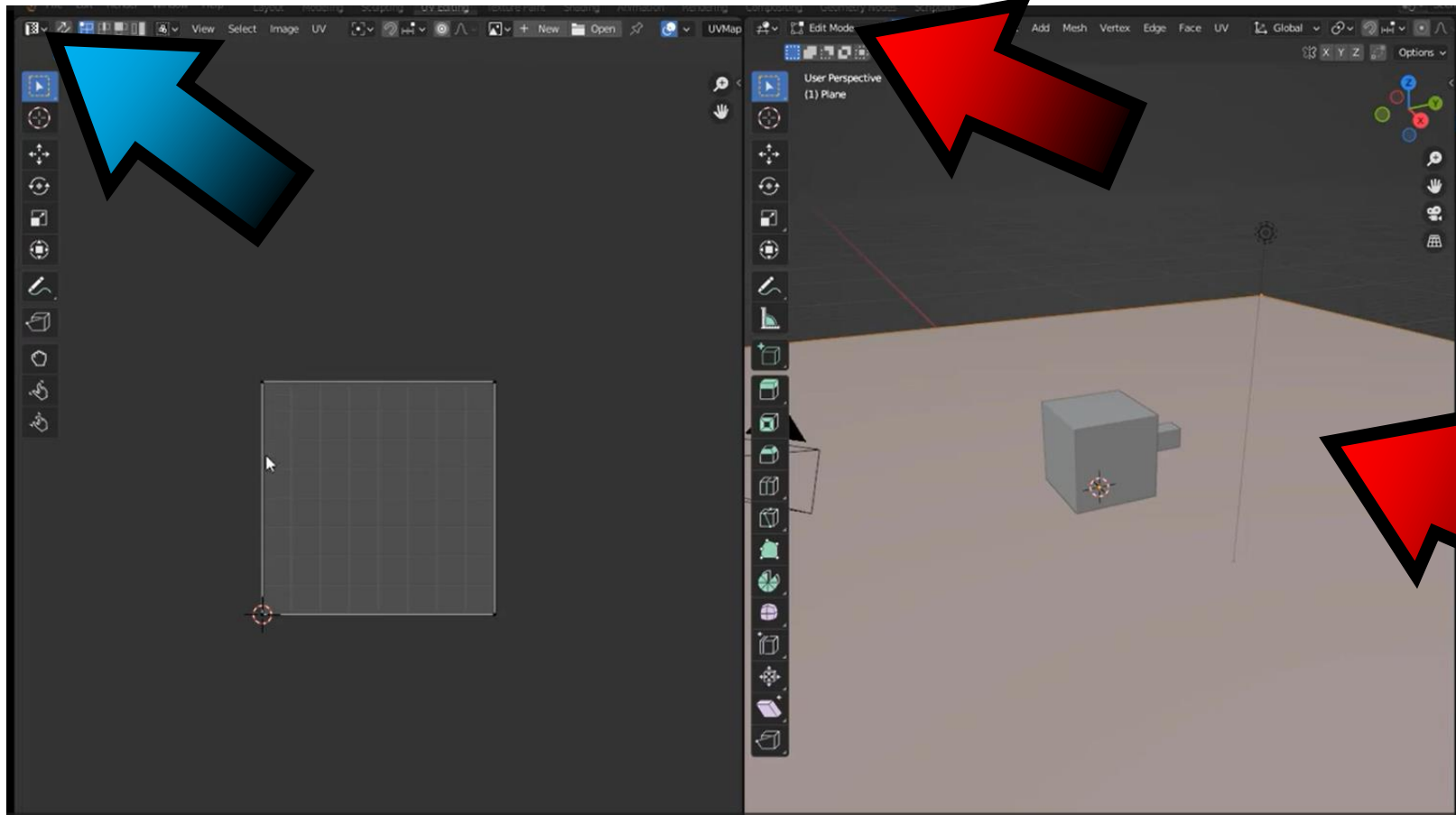
ENTER THE **OUTLINER**



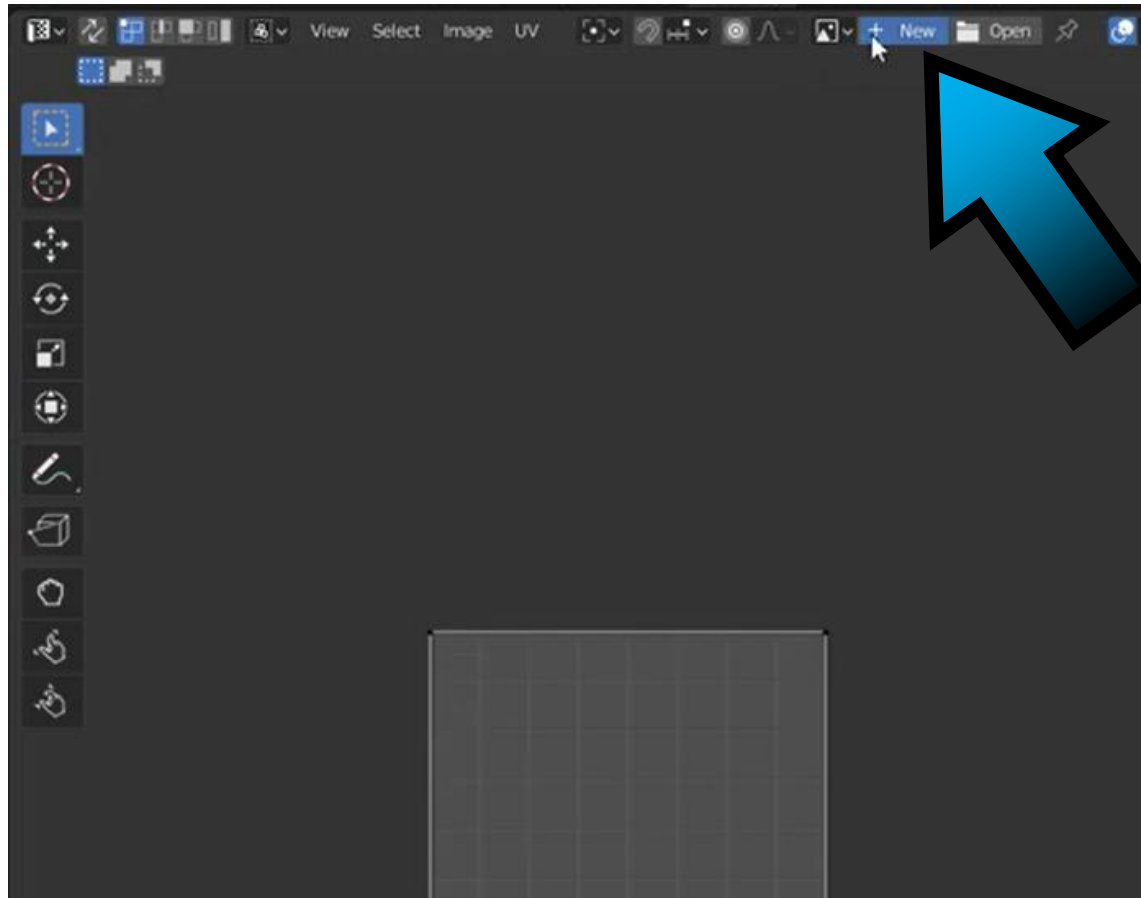
RENAME



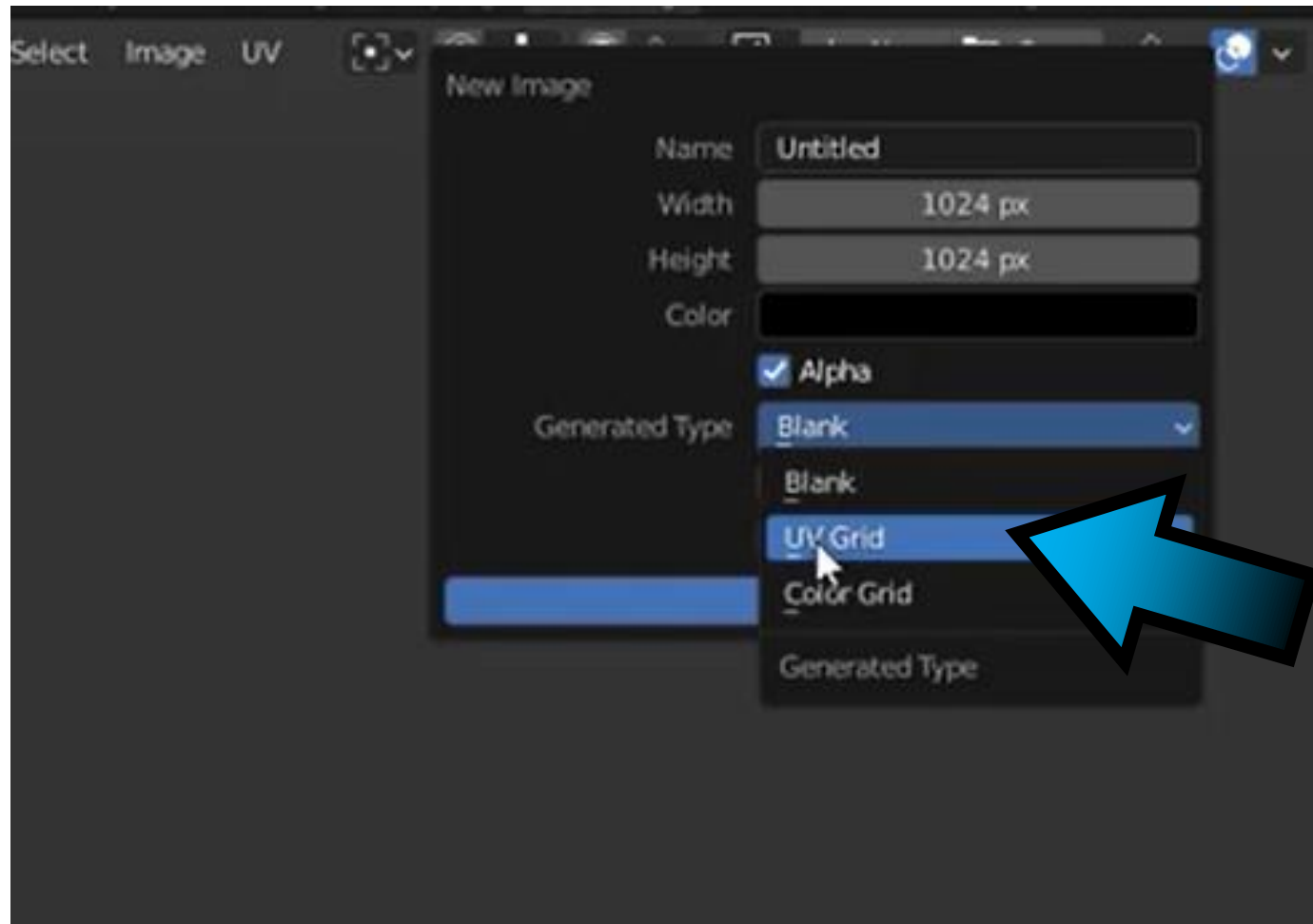
ENTER TO THE **PLANE** EDIT MODE AND SPLT THE WINDOWS BY SELECTING **UV EDITOR**



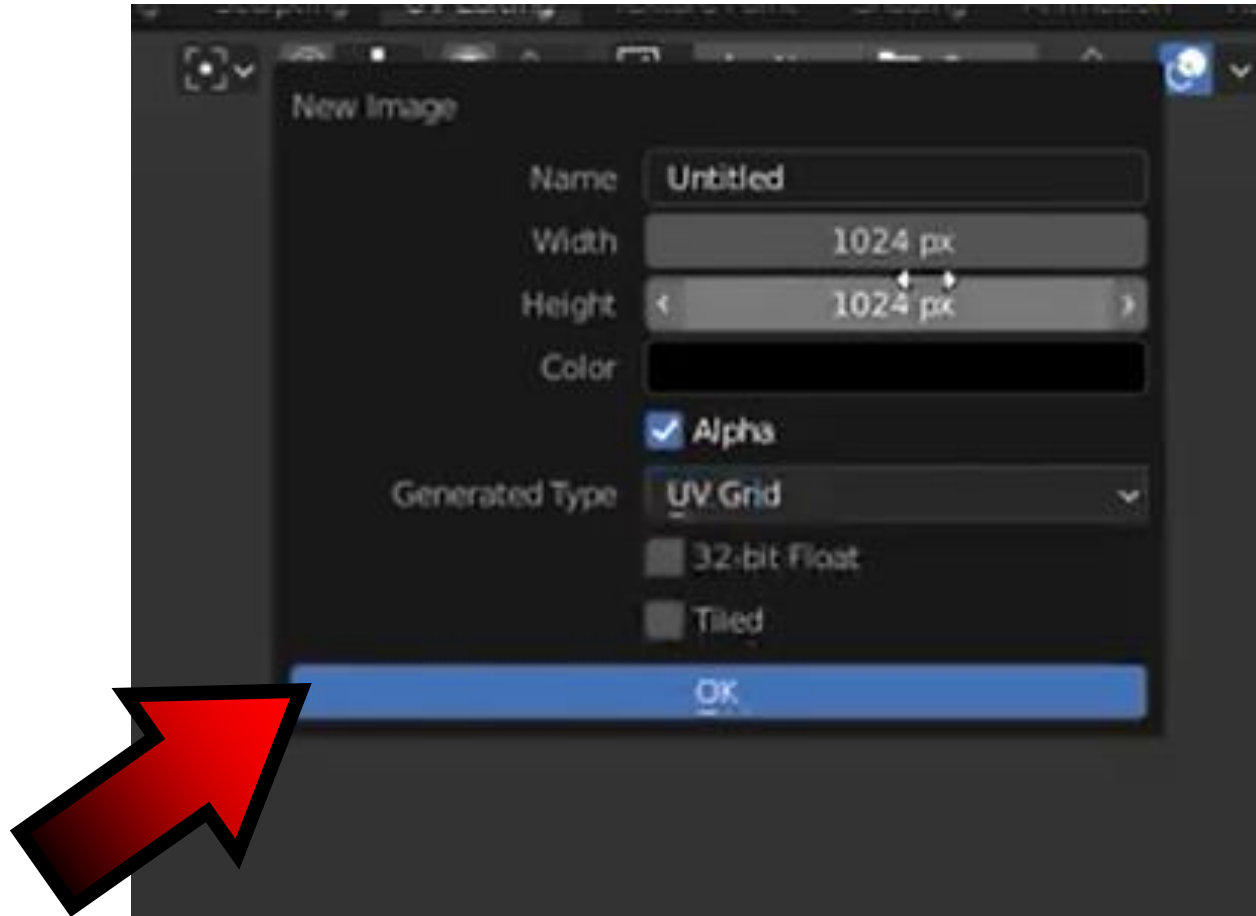
CLICK ON **NEW**



SELECT UV GRID

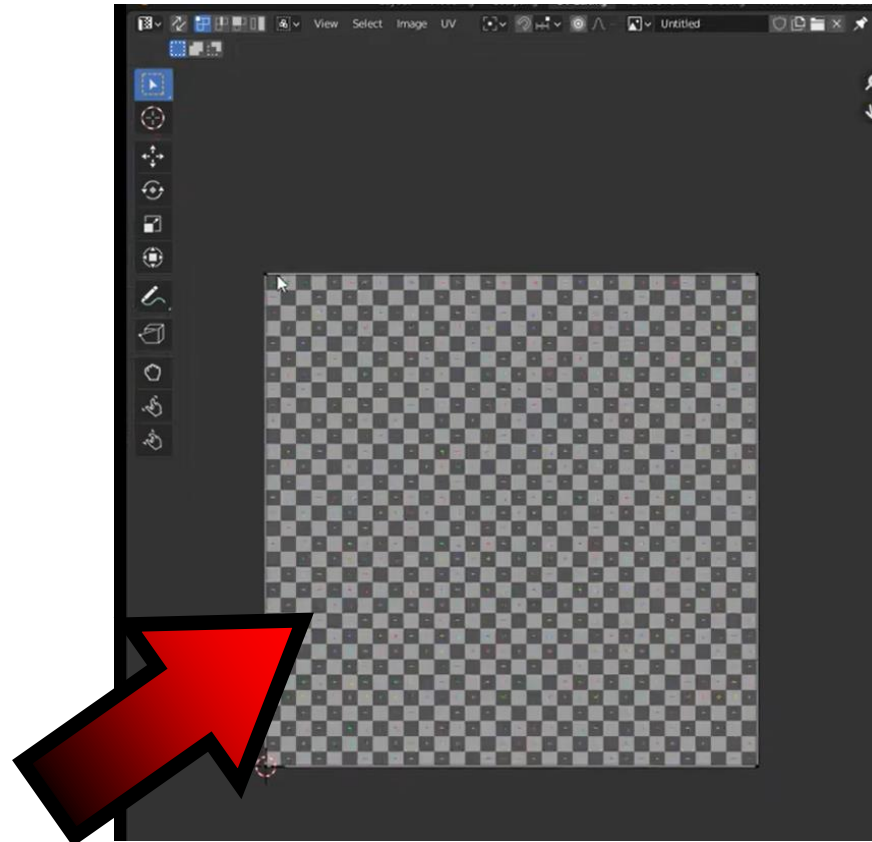


APPROVE **OK**

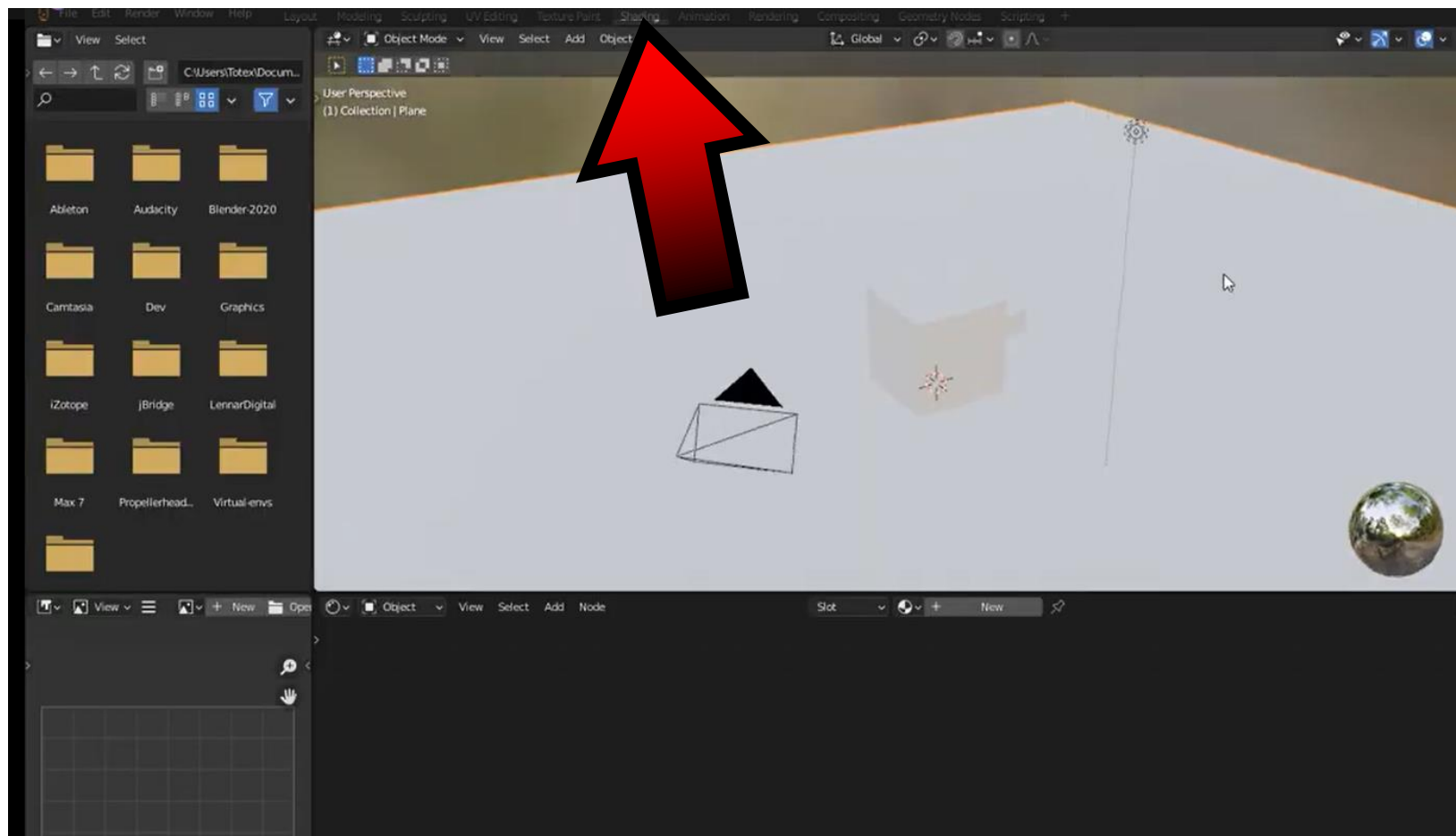


UPBGE

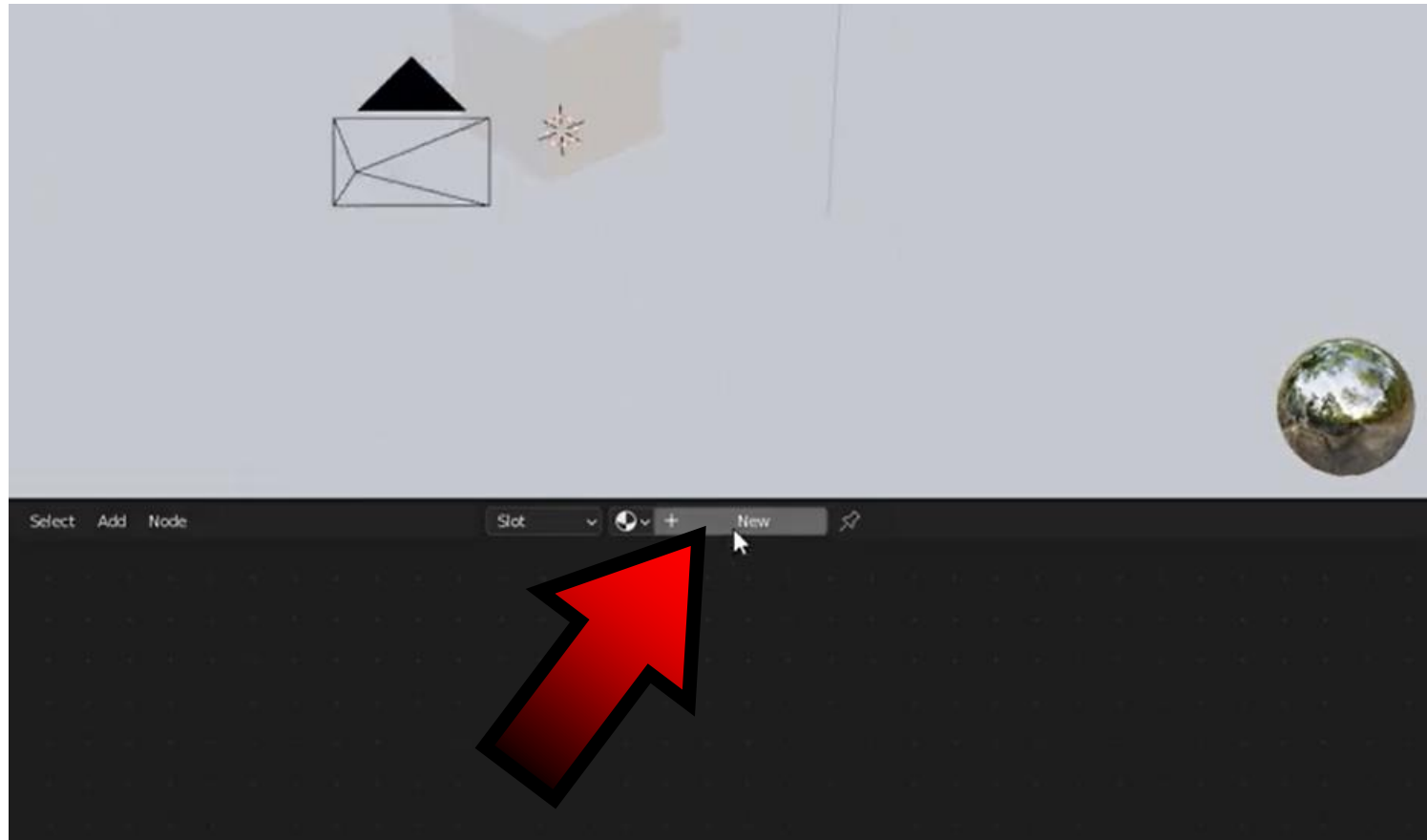
PLANE VIEW IN UV EDITOR



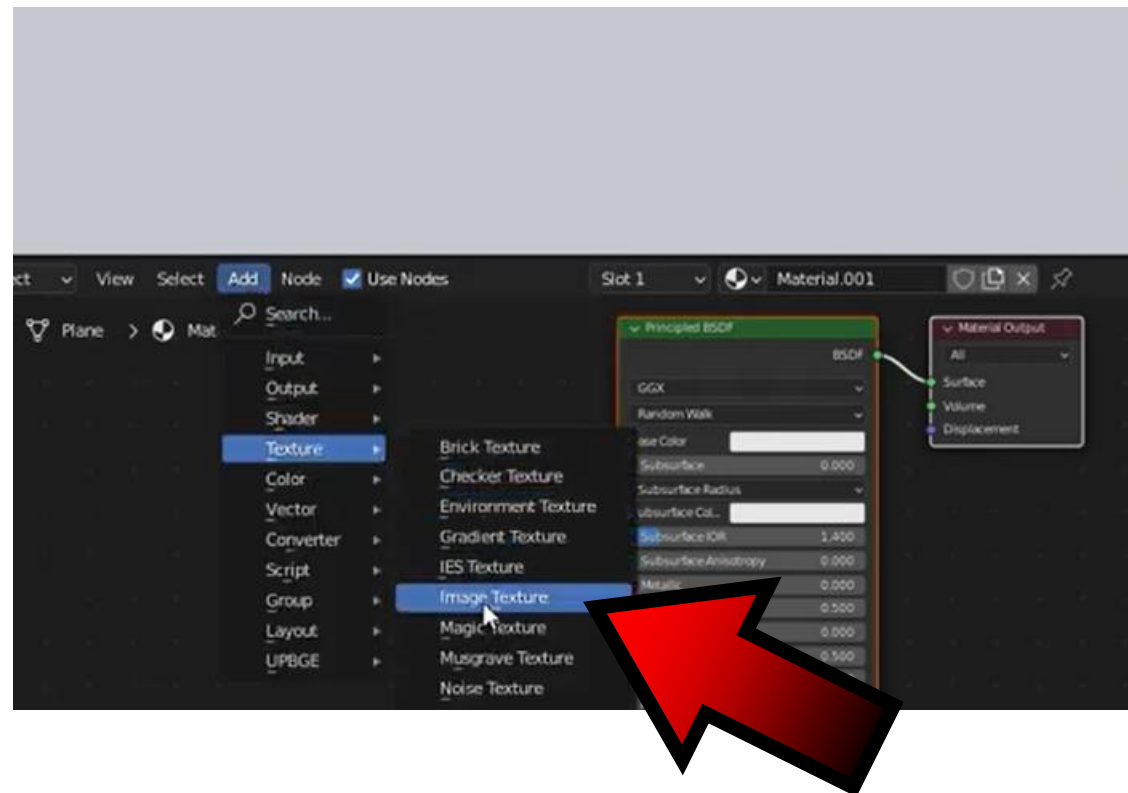
GO TO VIEW **SHADING**



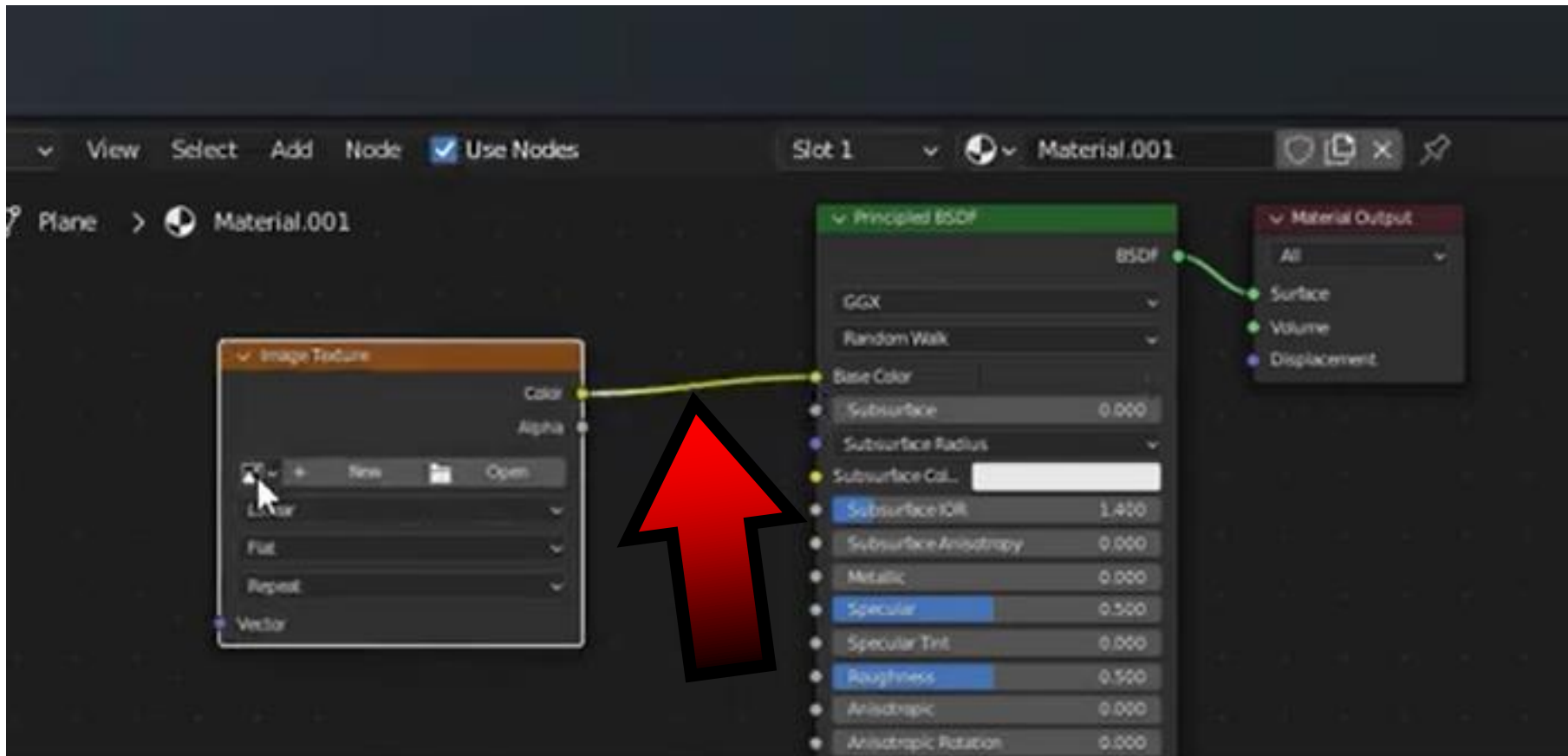
CLICK ON **NEW**



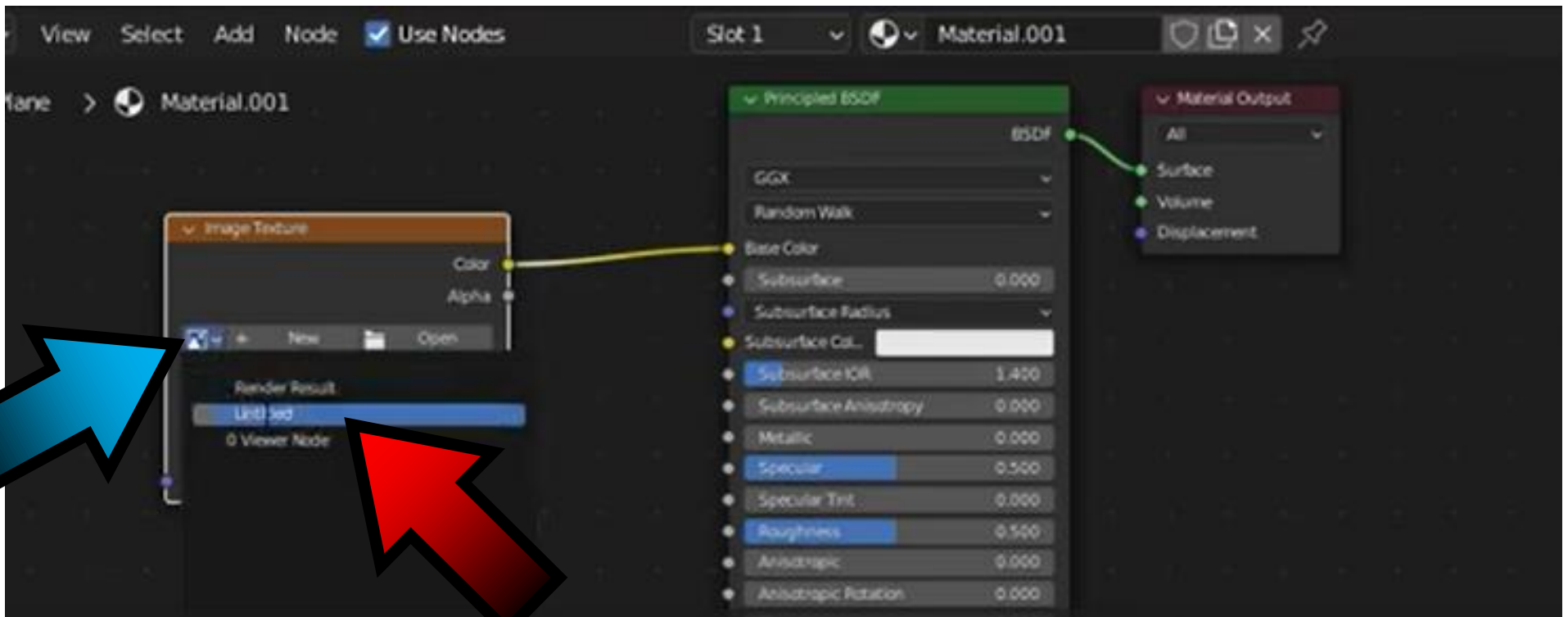
PRESS **SHIFT+A** AND ADD **IMAGE TEXTURE**



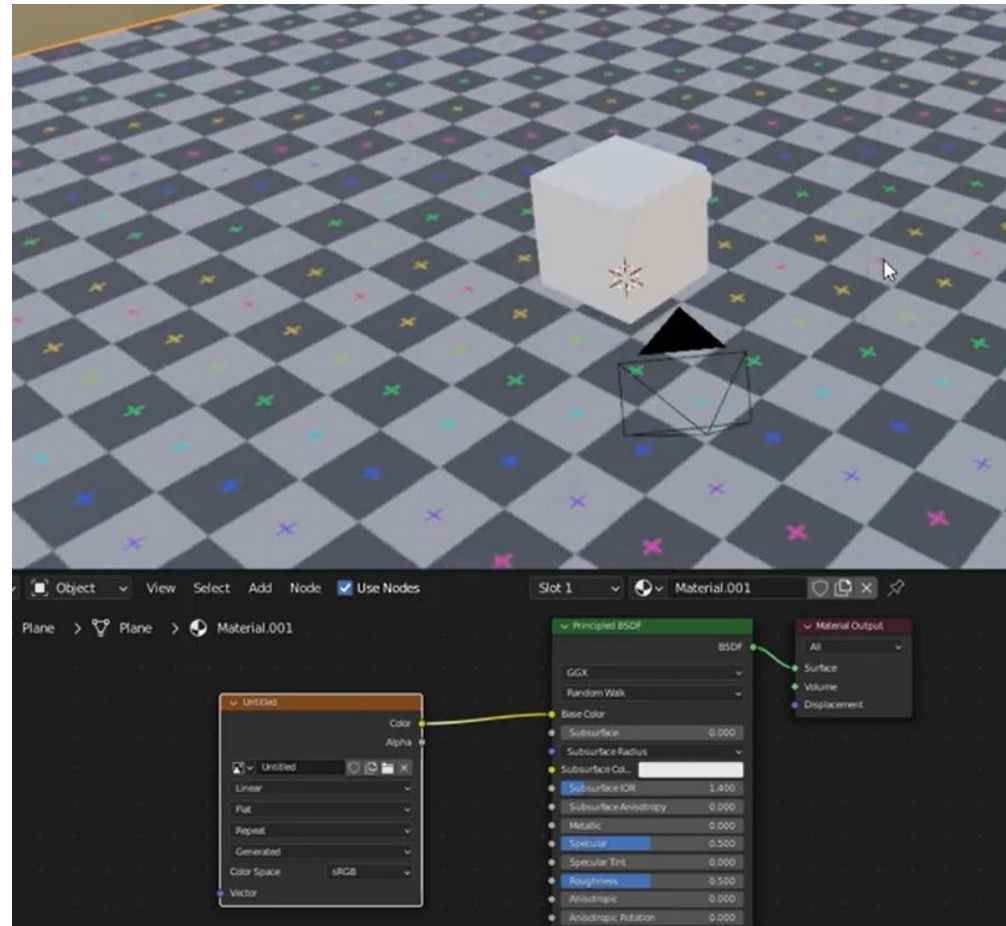
CONNECT NODS



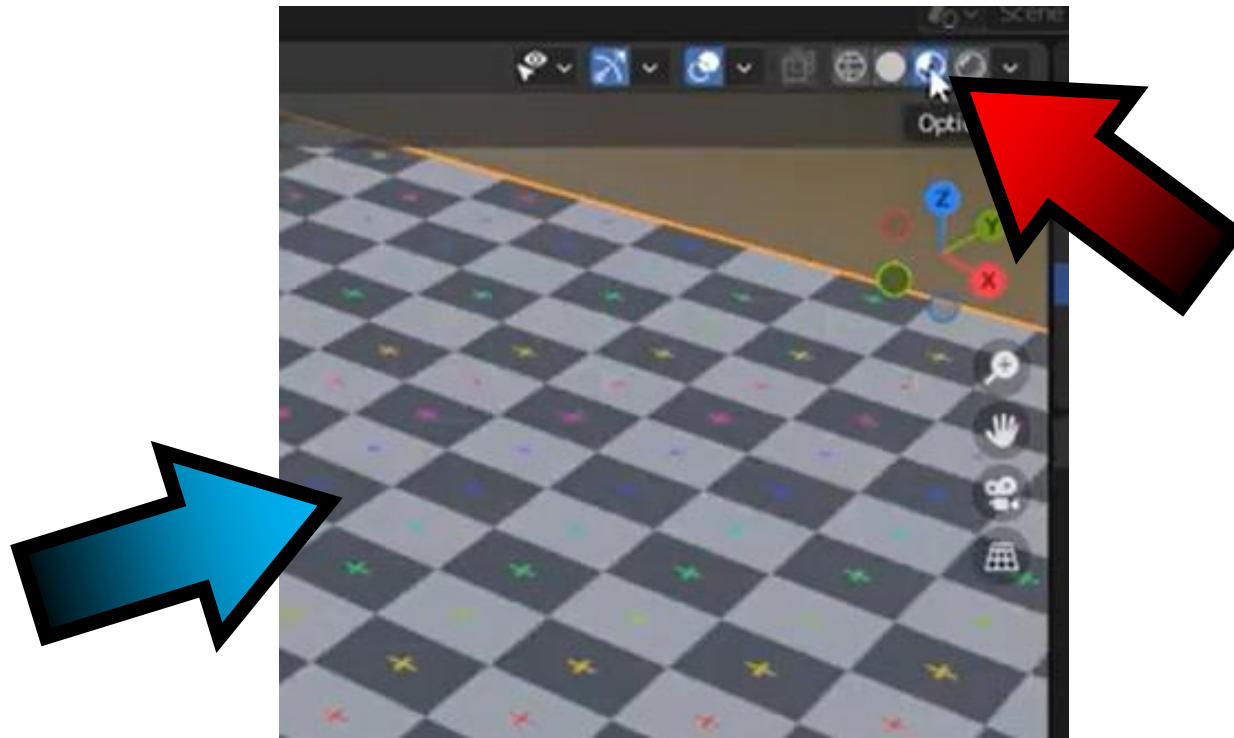
SELECT TEXTURE FROM **UV EDITOR**



VIEW IN SHADING



IF YOU DON'T SEE THE GRAPHIC
CLICK ON
MATERIAL PREVIEW

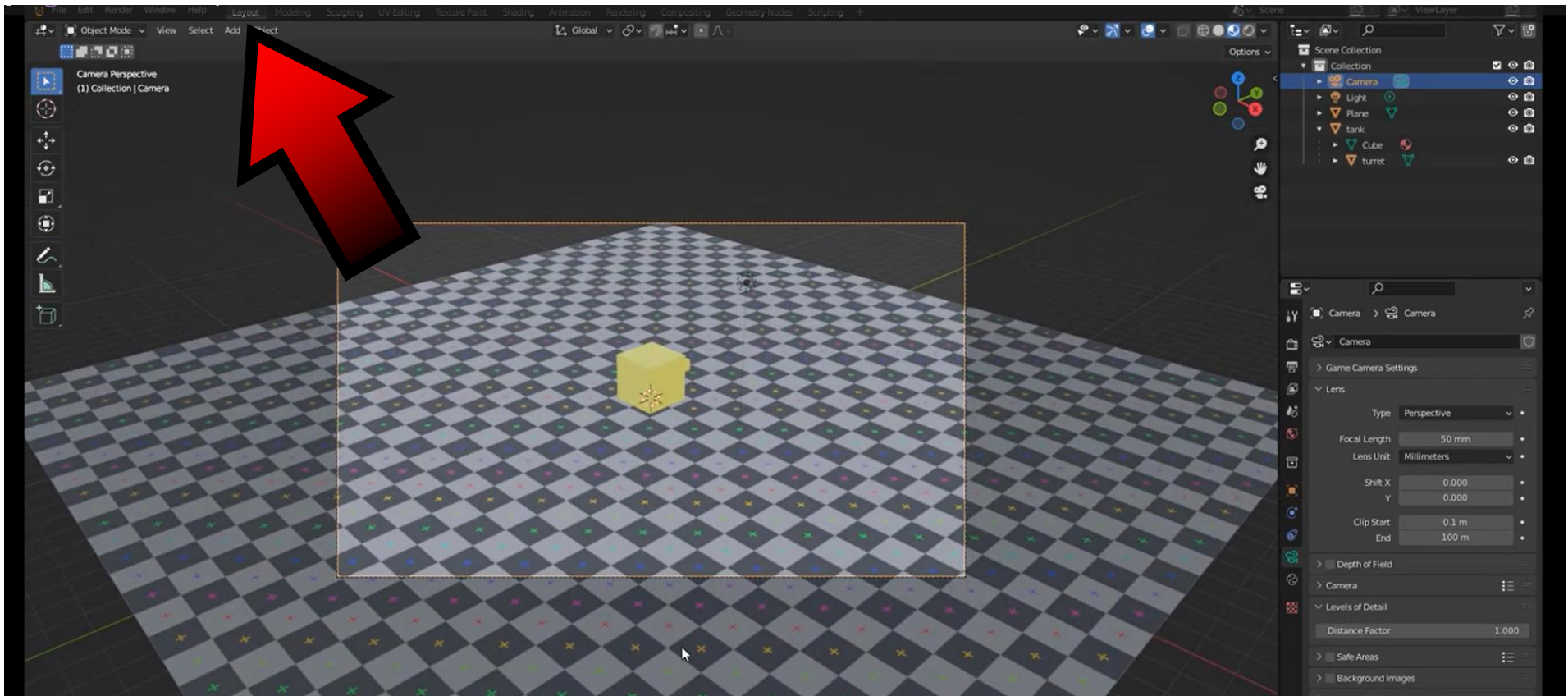




POWER OF AR AND VR

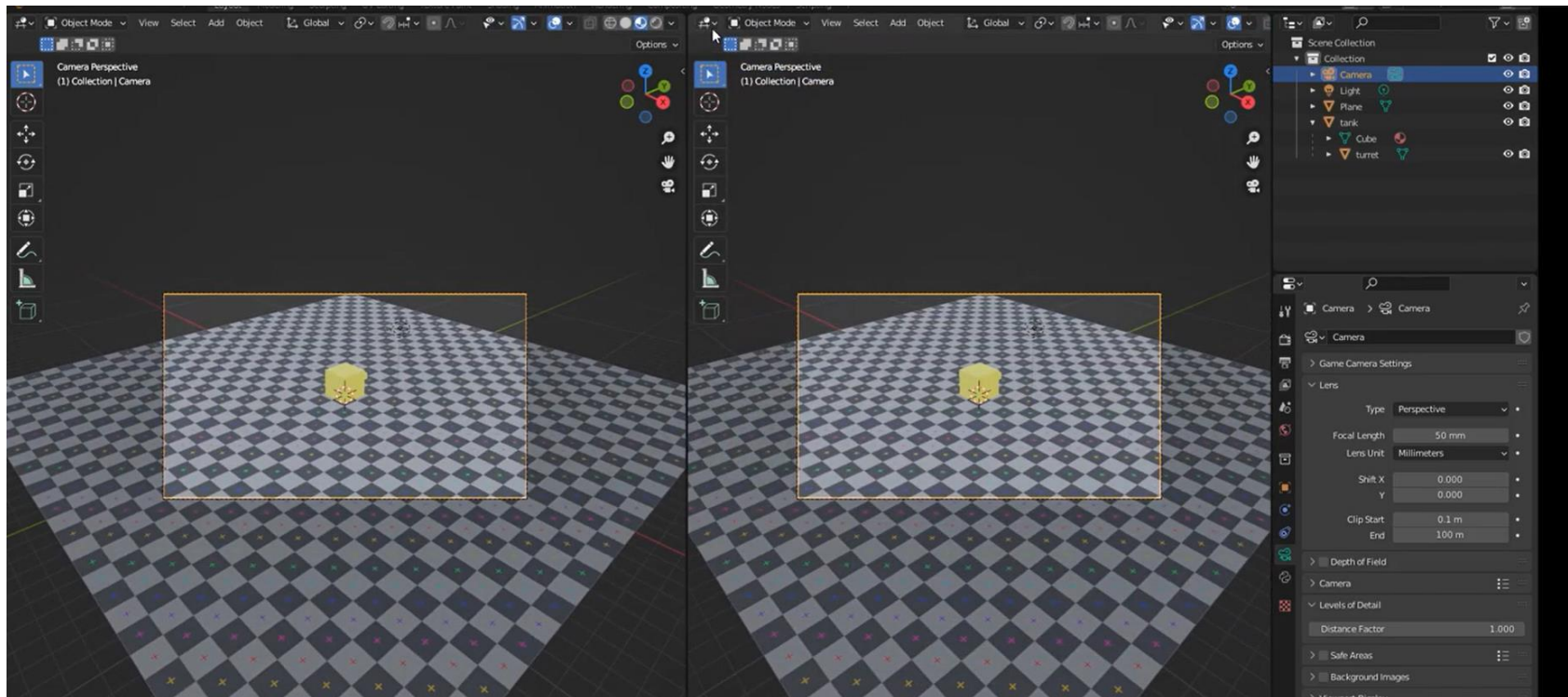


GO TO **LAYOUT**

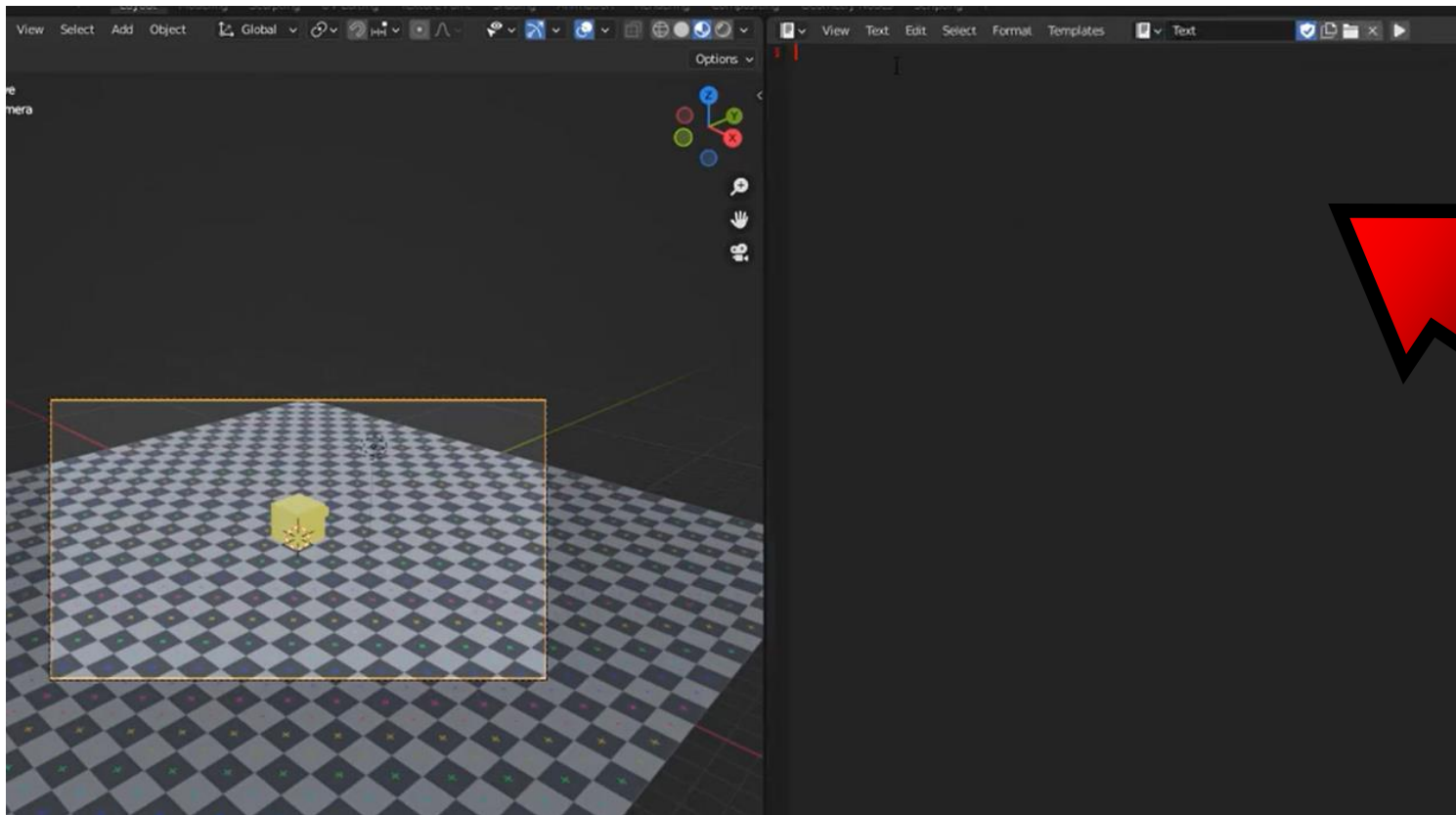


UPBGE

SPLIT WINDOW

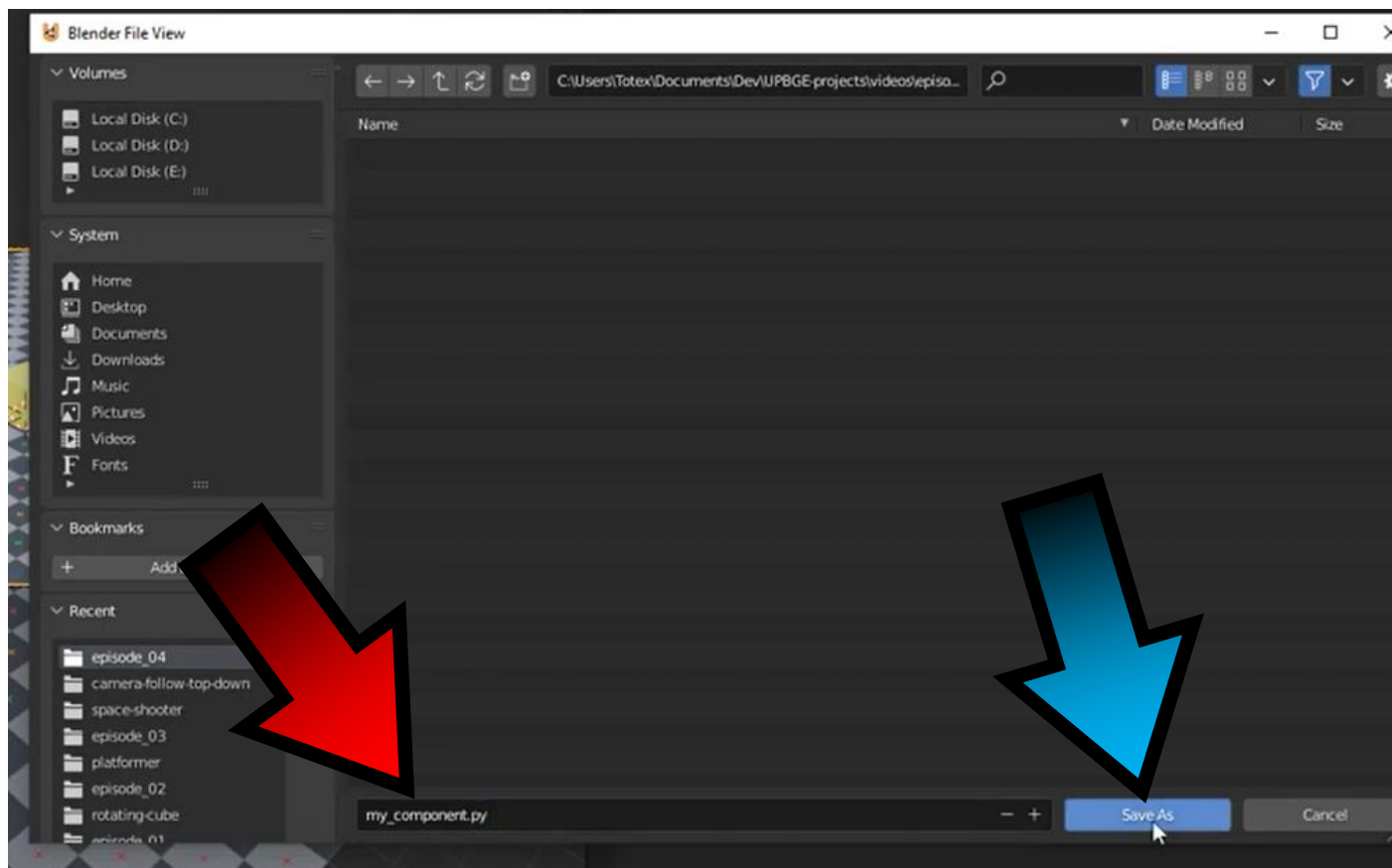


SELECT **TEXT EDIT**



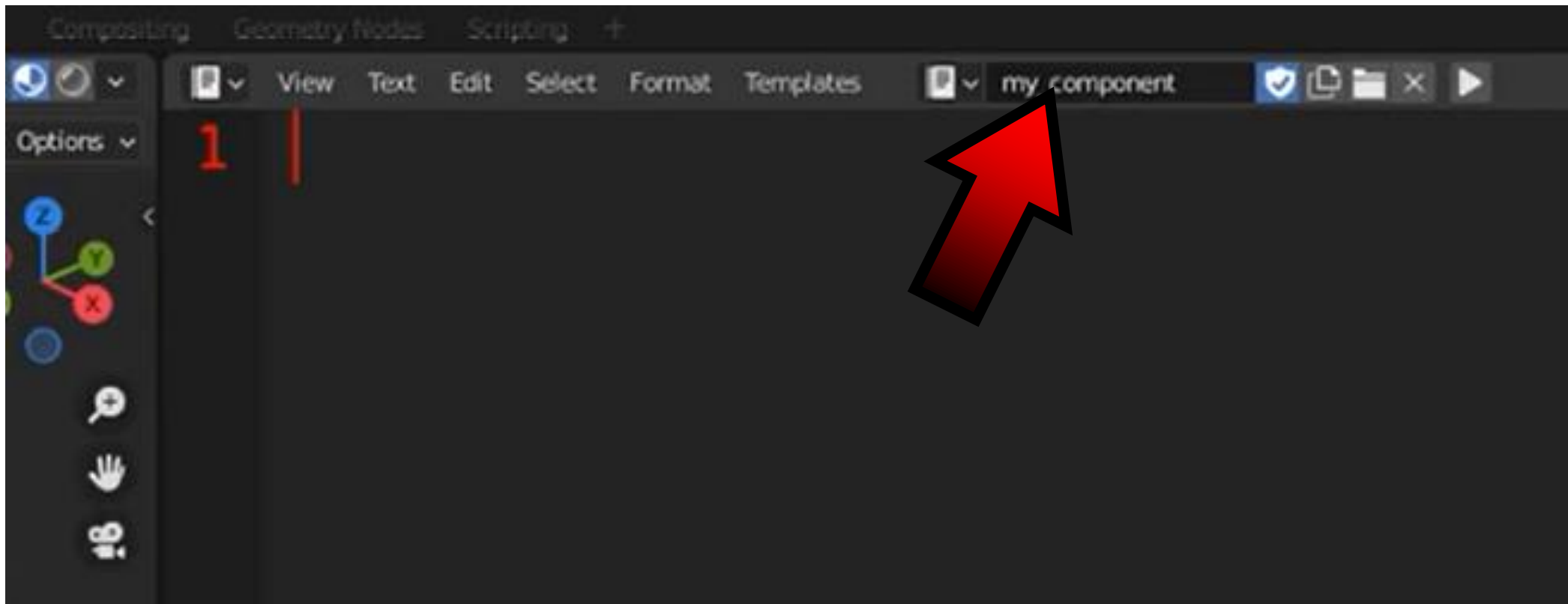
SAVE THE SCRIPT

THE NAME IS VERY IMPORTANT

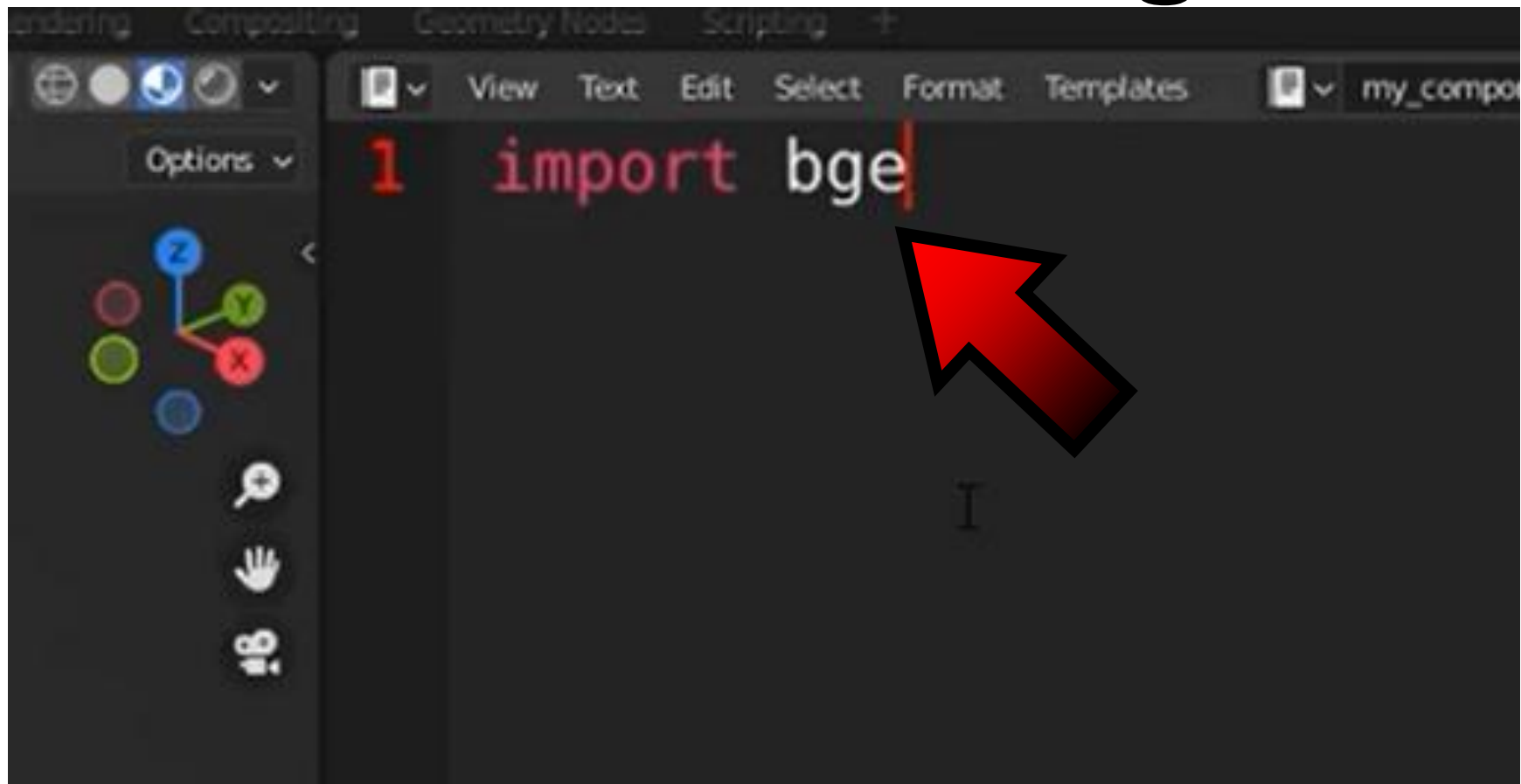


ENTER

THE NAME OF THE SCRIPT HERE

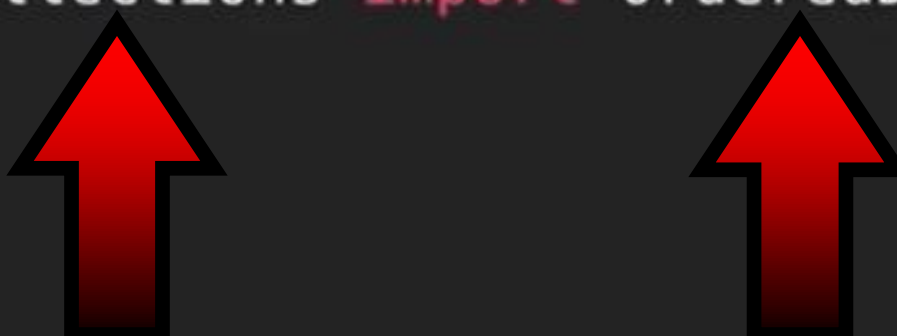


IMPORT FROM **BGE** **B**lender **G**ame **E**ngine



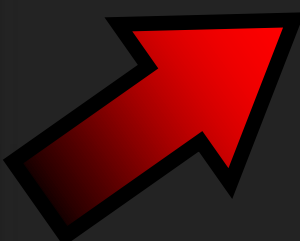
IMPORT FROM COLLECTIONS

```
ing Compositing Geometry Nodes Scripting +
View Text Edit Select Format Templates my_component
Options
1 import bge
2 |from collections import OrderedDict
```





REFERENCE TO CLASS MOVEMENT

```
Animation Rendering Compositing Geometry Nodes Scripting +
Options v View Text Edit Select Format Templates my_component
1 import bge
2 from collections import OrderedDict
3
4 class Movement(bge.types.KX_PythonComponent):|
```




WE DETERMINE **THE ARGUMENTS**

```
Rendering Compositing Geometry Nodes Scripting +
View Text Edit Select Format Templates my_component
1 import bge
2 from collections import OrderedDict
3
4 class Movement(bge.types.KX_PythonComponent):
5     args = OrderedDict([
6         ("Move Speed", 0.2),
7         ("Turn Speed", 0.04)
8     ])
9
```



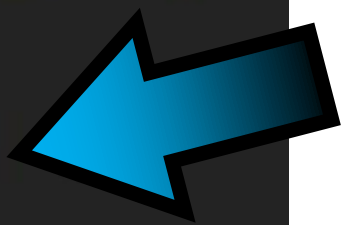

DETERMINING THE STARTING MODE

```
4 class Movement(bge.types.KX_PythonComponent):  
5     args = OrderedDict([  
6         ("Move Speed", 0.2),  
7         ("Turn Speed", 0.04)  
8     ])  
9  
10    def start(self, args):  
11        self.move_speed = args['Move Speed']  
12        self.turn_speed = args['Turn Speed']
```



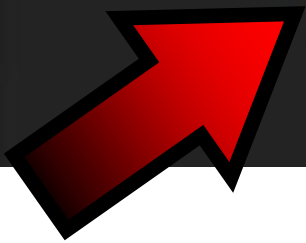
DETRMINING **THE UPDATEING MODE**

```
4 class Movement(bge.types.KX_PythonComponent):
5     args = OrderedDict([
6         ("Move Speed", 0.2),
7         ("Turn Speed", 0.04)
8     ])
9
10    def start(self, args):
11        self.move_speed = args['Move Speed']
12        self.turn_speed = args['Turn Speed']
13
14    def update(self):
15        keyboard = bge.logic.keyboard
```



WE DETERMINE HOW THE DATA IS ENTERED

```
9
10     def start(self, args):
11         self.move_speed = args['Move Speed']
12         self.turn_speed = args['Turn Speed']
13
14     def update(self):
15         keyboard = bge.logic.keyboard
16         inputs = keyboard.inputs
```





POWER OF AR AND VR



Mouse Keys

```
bge.events.LEFTMOUSE  
bge.events.MIDDLEMOUSE  
bge.events.RIGHTMOUSE  
bge.events.WHEELUPMOUSE  
bge.events.WHEELDOWNMOUSE  
bge.events.MOUSEX  
bge.events.MOUSEY
```

Alphabet keys

```
bge.events.AKEY  
bge.events.BKEY  
bge.events.CKEY  
bge.events.DKEY  
bge.events.EKEY  
bge.events.FKEY  
bge.events.GKEY  
bge.events.HKEY  
bge.events.IKEY  
bge.events.JKEY  
bge.events.KKEY  
bge.events.LKEY  
bge.events.MKEY  
bge.events.NKEY  
bge.events.OKEY  
bge.events.PKEY  
bge.events.QKEY  
bge.events.RKEY  
bge.events.SKEY  
bge.events.TKEY  
bge.events.UKEY  
bge.events.VKEY  
bge.events.WKEY  
bge.events.XKEY  
bge.events.YKEY  
bge.events.ZKEY
```

Number keys

```
bge.events.ZEROKEY  
bge.events.ONEKEY  
bge.events.TWOKEY  
bge.events.THREEKEY  
bge.events.FOURKEY  
bge.events.FIVEKEY  
bge.events.SIXKEY  
bge.events.SEVENKEY  
bge.events.EIGHTKEY  
bge.events.NINEKEY
```

Numberpad Keys

```
bge.events.PAD0  
bge.events.PAD1  
bge.events.PAD2  
bge.events.PAD3  
bge.events.PAD4  
bge.events.PAD5  
bge.events.PAD6  
bge.events.PAD7  
bge.events.PAD8  
bge.events.PAD9  
bge.events.PADPERIOD  
bge.events.PADSLASHKEY  
bge.events.PADASTERKEY  
bge.events.PADMINUS  
bge.events.PADENTER  
bge.events.PADPLUSKEY
```

Modifiers Keys

```
bge.events.CAPSLOCKKEY  
bge.events.LEFTCTRLKEY  
bge.events.LEFTALTKEY  
bge.events.RIGHTALTKEY  
bge.events.RIGHTCTRLKEY  
bge.events.RIGHTSHIFTKEY  
bge.events.LEFTSHIFTKEY
```

Arrow Keys

```
bge.events.LEFTARROWKEY  
bge.events.DOWNARROWKEY  
bge.events.RIGHTARROWKEY  
bge.events.UPARROWKEY
```

Function Keys


```
bge.events.F1KEY  
bge.events.F2KEY  
bge.events.F3KEY  
bge.events.F4KEY  
bge.events.F5KEY  
bge.events.F6KEY  
bge.events.F7KEY  
bge.events.F8KEY  
bge.events.F9KEY  
bge.events.F10KEY  
bge.events.F11KEY  
bge.events.F12KEY  
bge.events.F13KEY  
bge.events.F14KEY  
bge.events.F15KEY  
bge.events.F16KEY  
bge.events.F17KEY  
bge.events.F18KEY  
bge.events.F19KEY
```

Other Keys

```
bge.events.ACCENTGRAVEKEY  
bge.events.BACKSLASHKEY  
bge.events.BACKSPACEKEY  
bge.events.COMMAKEY  
bge.events.DELKEY  
bge.events.ENDKEY  
bge.events.EQUALKEY  
bge.events.ESCKEY  
bge.events.HOMEKEY  
bge.events.INSERTKEY  
bge.events.LEFTBRACKETKEY  
bge.events.LINEFEEDKEY  
bge.events.MINUSKEY  
bge.events.PAGEDOWNKEY  
bge.events.PAGEUPKEY  
bge.events.PAUSEKEY  
bge.events.PERIODKEY  
bge.events.QUOTEKEY  
bge.events.RIGHTBRACKETKEY  
bge.events.ENTERKEY  
bge.events.SEMICOLONKEY  
bge.events.SLASHKEY  
bge.events.SPACEKEY  
bge.events.TABKEY
```

COMPONENT ZEROING



```
13
14     def update(self):
15         keyboard = bge.logic.keyboard
16         inputs = keyboard.inputs
17
18         move = 0
19         rotate = 0
```



MOVE

THE W KEY



```
18     move = 0
19     rotate = 0
20
21     if inputs[bge.events.WKEY].values[-1]:
22         move += self.move_speed
```



MOVE


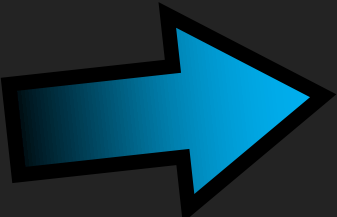
THE S KEY

```
21     if inputs[bge.events.WKEY].values[-1]:  
22         move += self.move_speed  
23     if inputs[bge.events.SKEY].values[-1]:  
24         move -= self.move_speed
```



ROTATE THE **A** AND **D** KEYS

```
21     if inputs[bge.events.WKEY].values[-1]:
22         move += self.move_speed
23     if inputs[bge.events.SKEY].values[-1]:
24         move -= self.move_s
25
26     if inputs[bge.events.AKEY].values[-1]:
27         rotate += self.turn_speed
28     if inputs[bge.events.DKEY].values[-1]:
29         rotate -= self.turn_speed
```

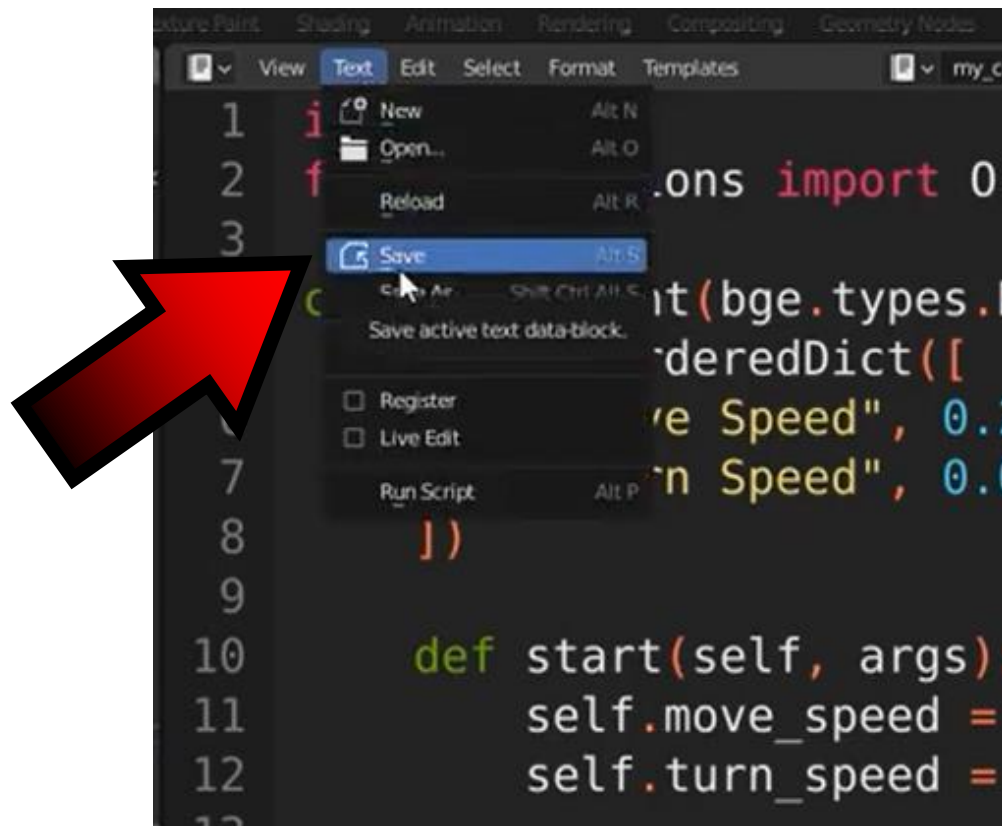


MOVEMENT ON LOCAL AXES

```
25
26     if inputs[bge.events.AKEY].values[-1]:
27         rotate += self.turn_speed
28     if inputs[bge.events.DKEY].values[-1]:
29         rotate -= self.turn_speed
30
31     self.object.applyMovement((0, move, 0), True)
```



SAVE THE SCRIPT



A screenshot of a code editor interface. The menu bar includes 'View', 'Text', 'Edit', 'Select', 'Format', and 'Templates'. A context menu is open over the code, listing options: 'New' (Alt N), 'Open...' (Alt O), 'Reload' (Alt R), 'Save' (Alt S), 'Copy All' (Shift Ctrl Alt C), 'Save active text data block.', 'Register', 'Live Edit', and 'Run Script' (Alt P). The 'Save' option is highlighted in blue. A large red arrow with a black outline points to the 'Save' option. The code in the background is Python, showing a class definition with a 'start' method.

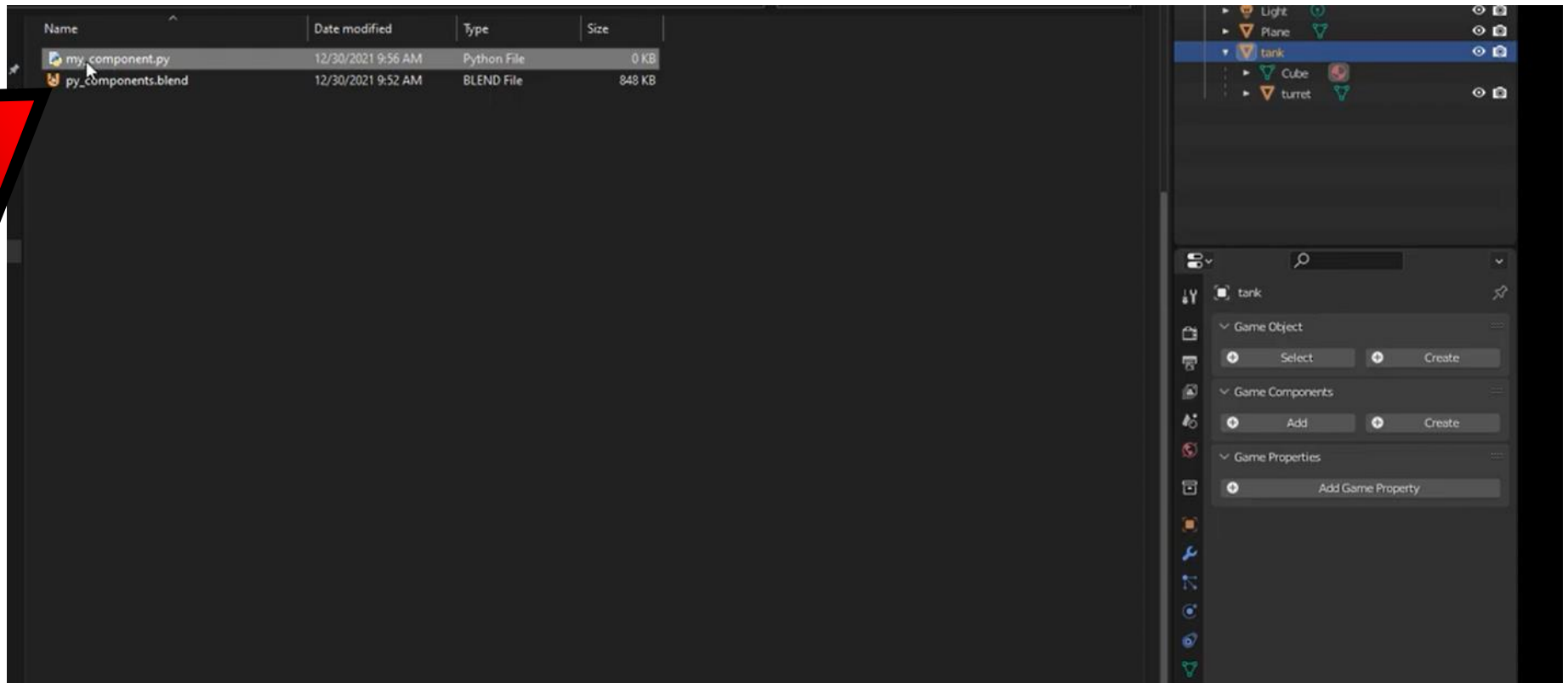
```
1 i New Alt N
2 f Open... Alt O
3 Reload Alt R
4 Save Alt S
5 Copy All Shift Ctrl Alt C
6 Save active text data block.
7 Register
8 Live Edit
9 Run Script Alt P
10 ))
11
12 def start(self, args):
13     self.move_speed =
14     self.turn_speed =
```



POWER OF AR AND VR

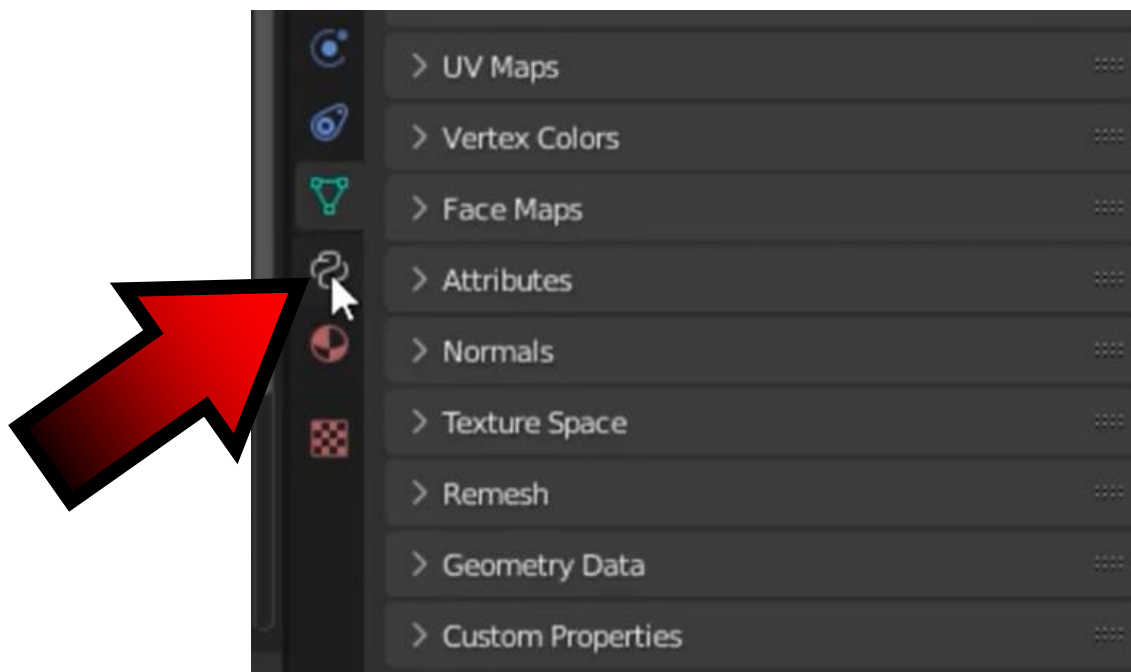


WE WILL NOT CHANGE THE NAME

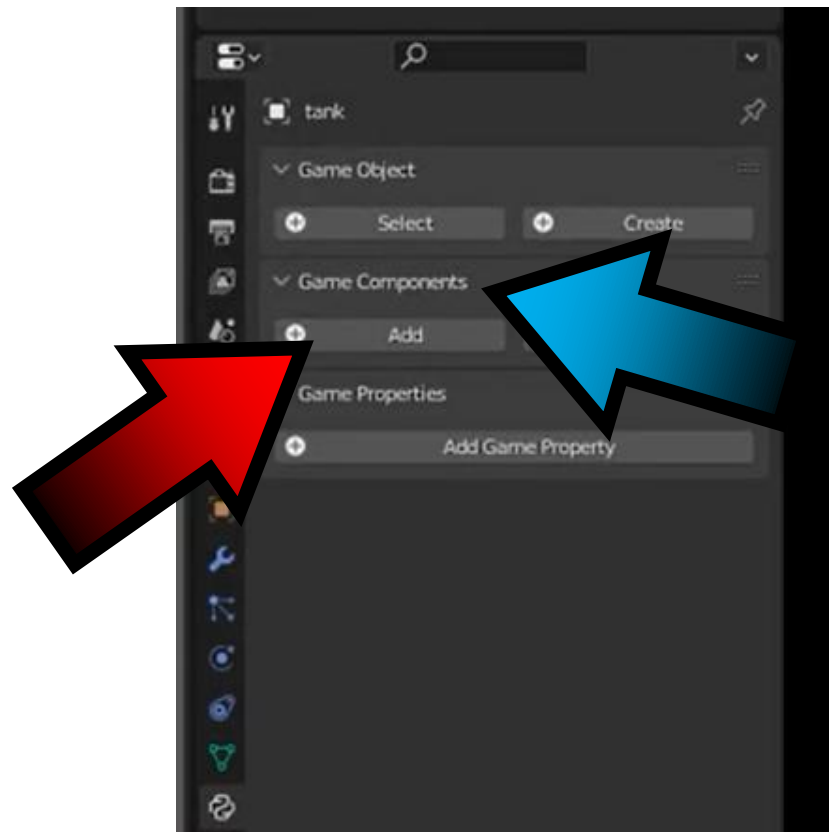


UPBGE

WE ENTER THE GAME OBJECT PROPERTIES



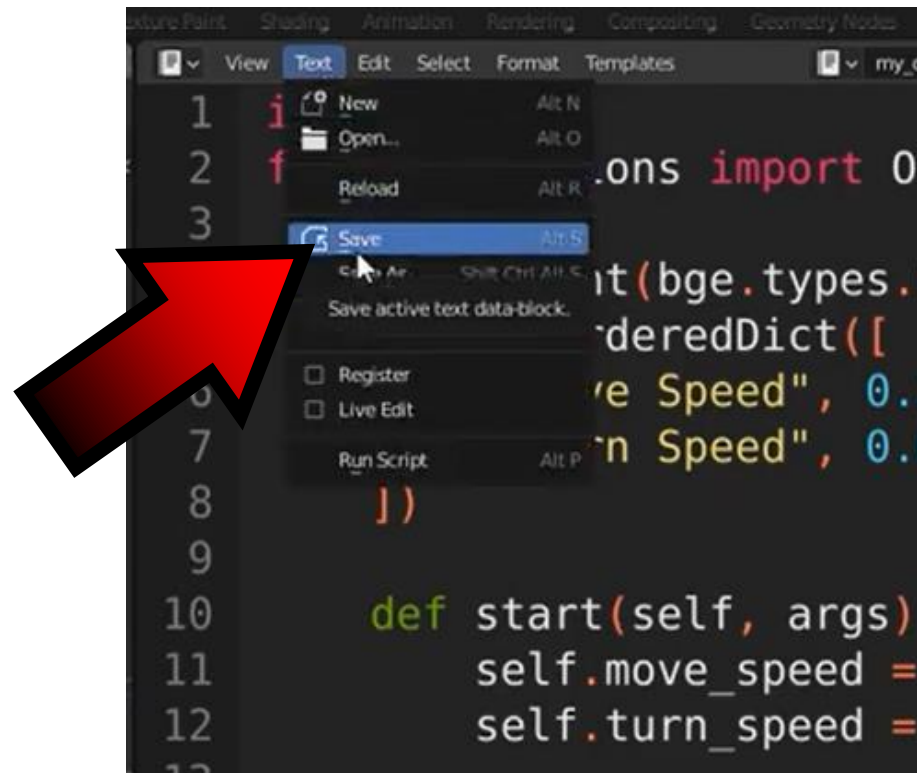
CLICK ON **ADD** IN **GAME COMPONENTS**



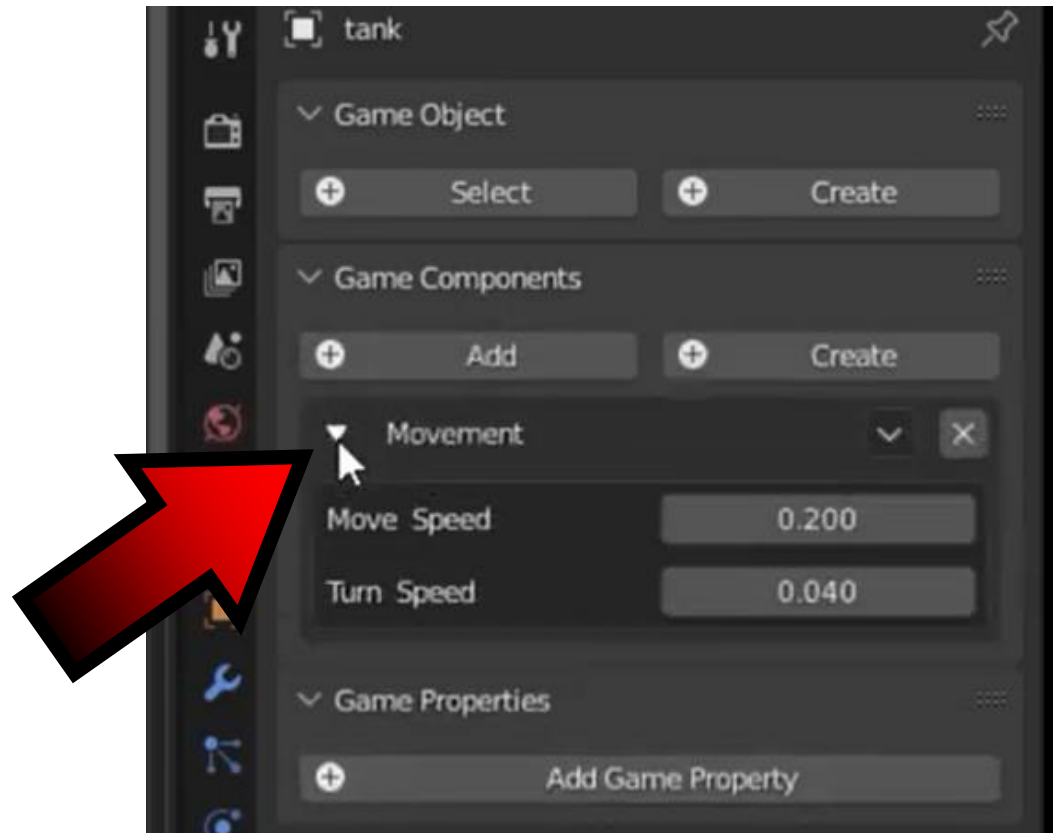
**ENTER THE FILE NAME
AND AFTER THE DOT
THE NAME OF THE CLASS**



SAVE THE SCRIPT AGAIN

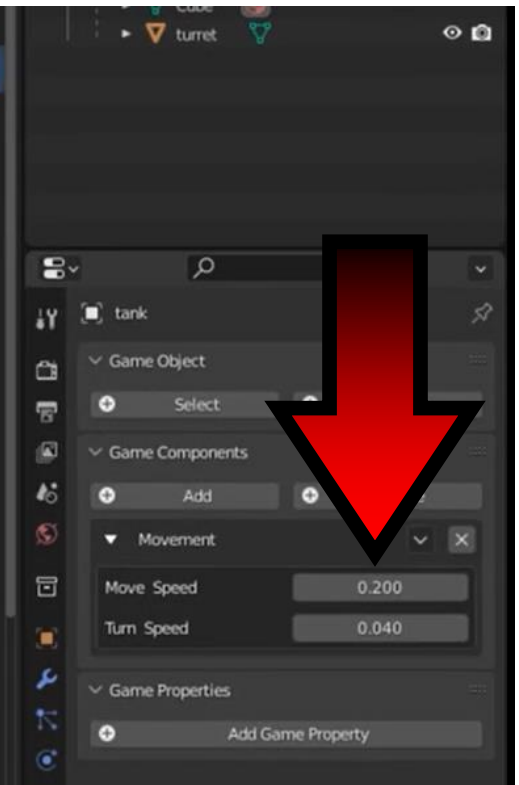


WE OPEN **MOVEMENT**



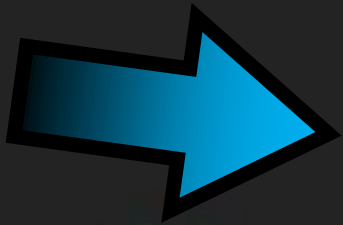
THESE **TWO ARGUMENTS** THEY WERE DOWNLOADED THERE

```
5  args = OrderedDict([
6      ("Move Speed", 0.2),
7      ("Turn Speed", 0.04)
8  ])
9
10 def start(self, args):
11     self.move_speed = args['Move Speed']
12     self.turn_speed = args['Turn Speed']
13
14 def update(self):
15     keyboard = bge.logic.keyboard
16     inputs = keyboard.inputs
17
18     move = 0
19     rotate = 0
20
21     if inputs[bge.events.WKEY].values[-1]:
```

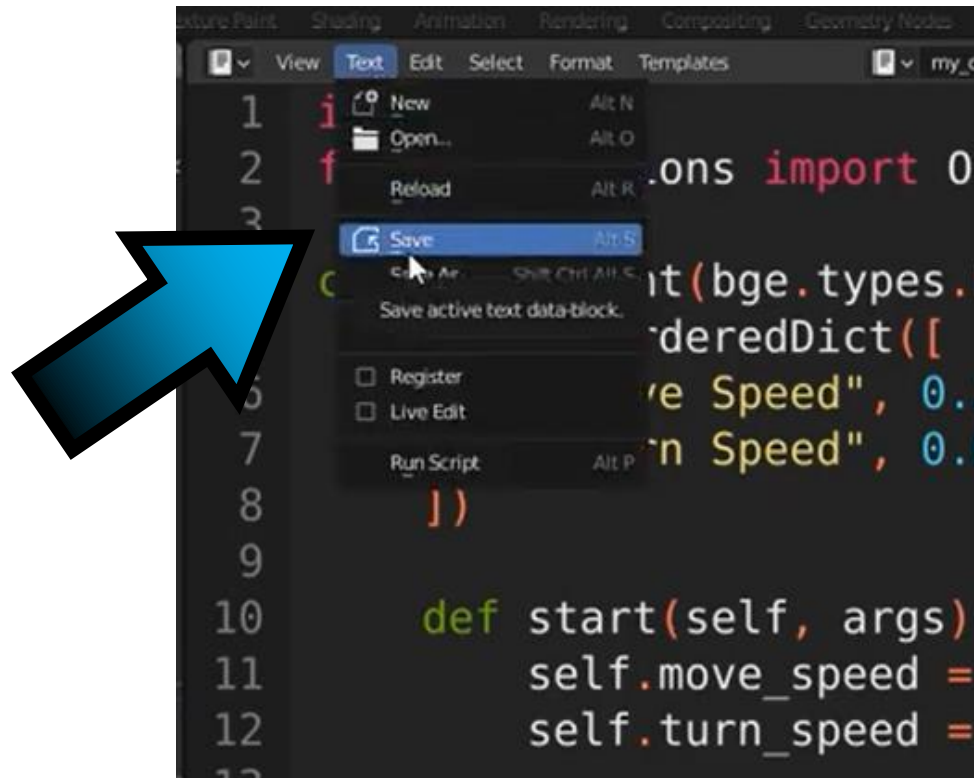


IF WE ADD **A THIRD**

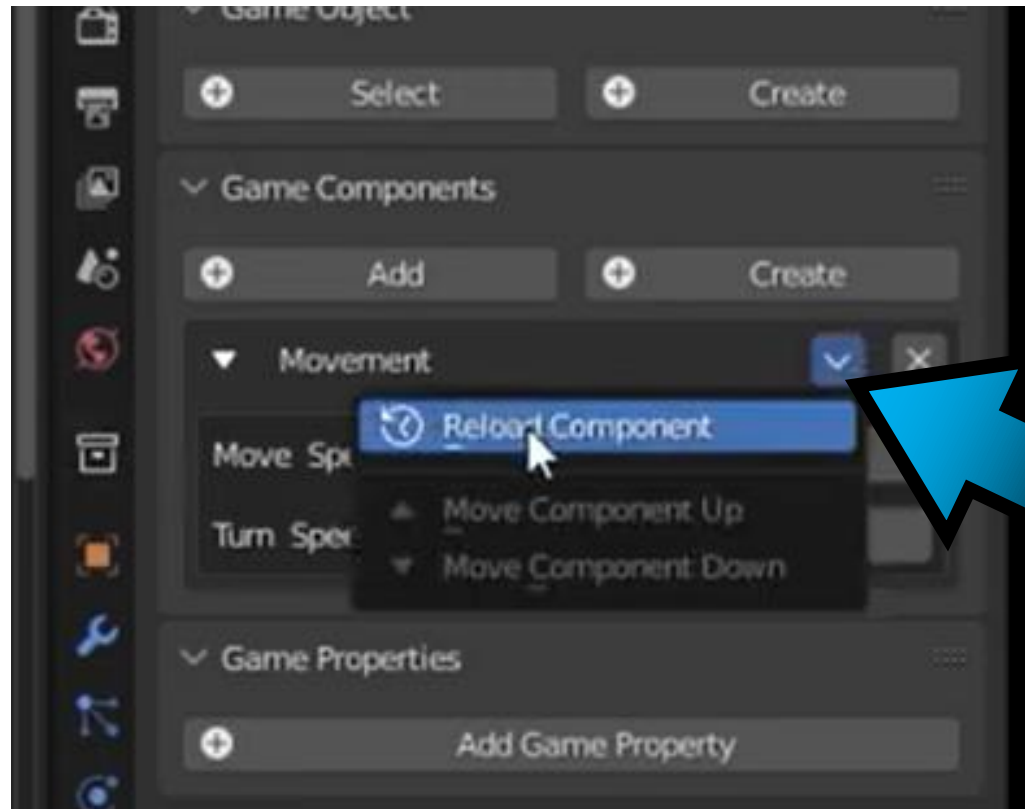
```
args = OrderedDict([  
    ("Move Speed", 0.2),  
    ("Turn Speed", 0.04),  
    ("My Var", 5)  
])
```



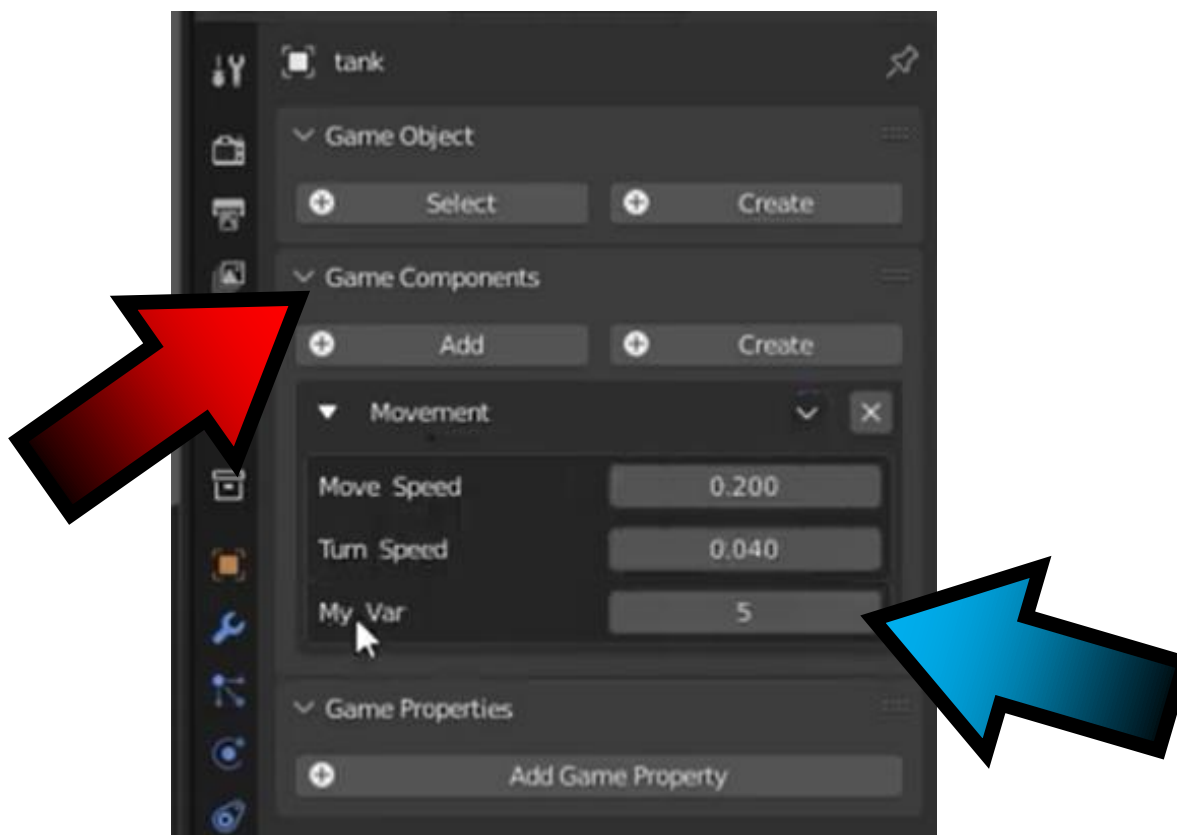
AND WE WILL SAVE THE SCRIPT



AFTER REFRESHING



A THIRD VARIABLE WILL APPEAR IN **GAME COMPONENTS**

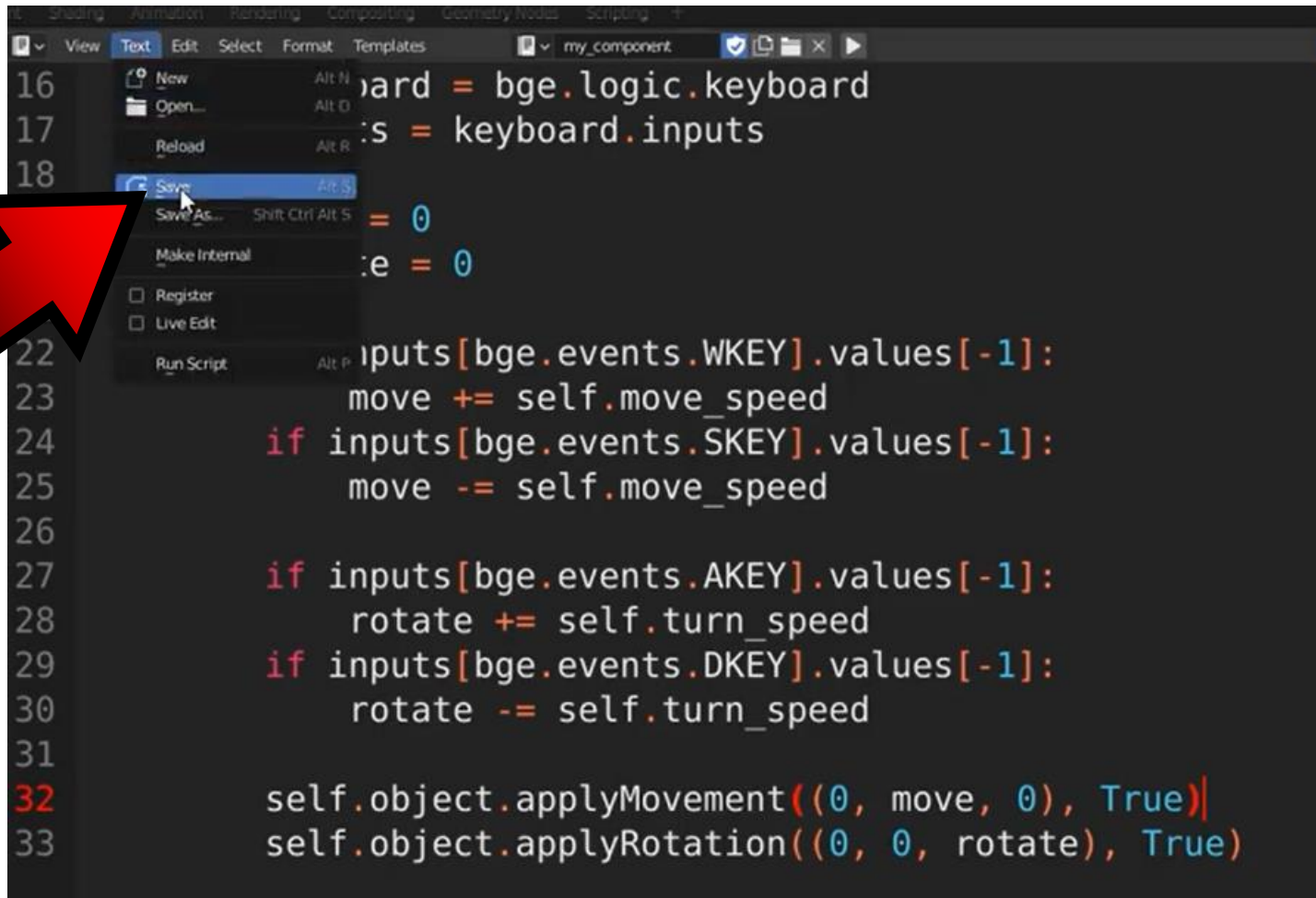


WE ADD A LOCAL ROTATION VECTOR **ARROUND THE Z AXIS**

```
31  
32 self.object.applyMovement((0, move, 0), True)  
33 self.object.applyRotation((0, 0, rotate), True)
```



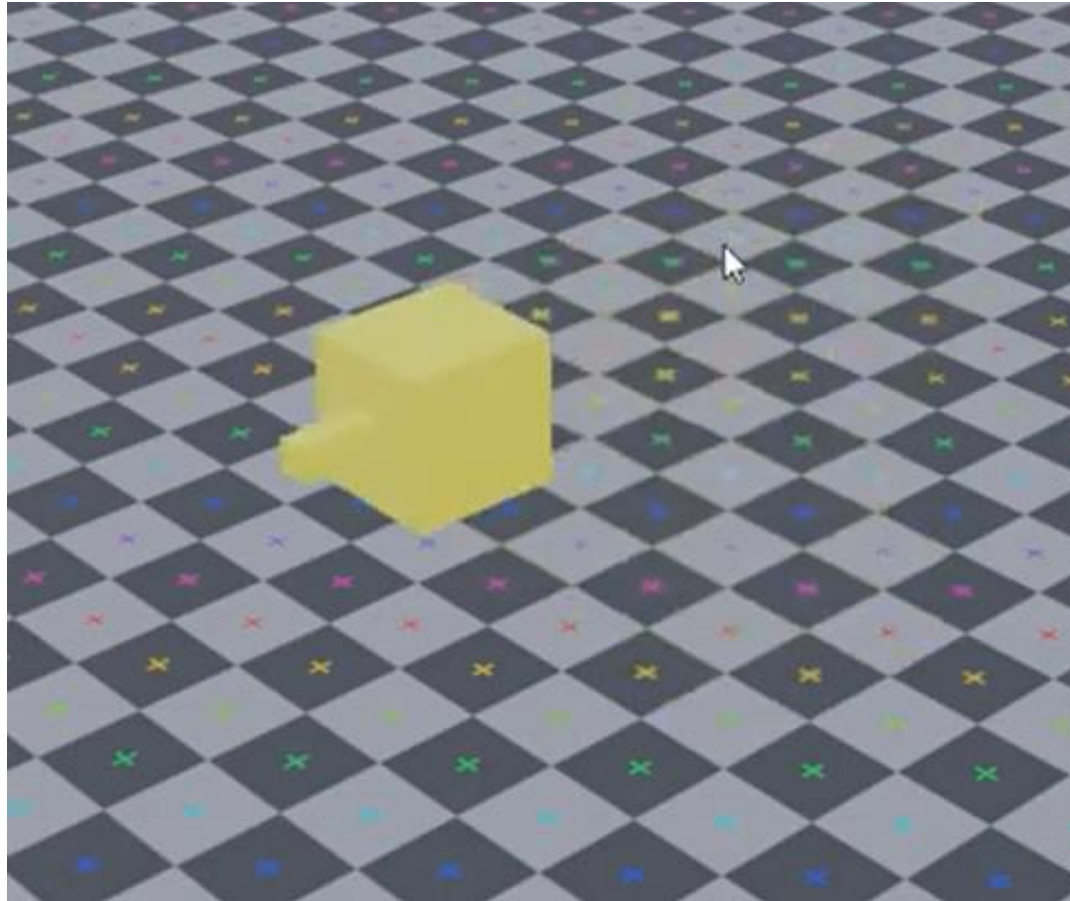
SAVE THE SCRIPT



```
16  keyboard = bge.logic.keyboard
17  :s = keyboard.inputs
18  :e = 0
22  inputs[bge.events.WKEY].values[-1]:
23      move += self.move_speed
24  if inputs[bge.events.SKEY].values[-1]:
25      move -= self.move_speed
26
27  if inputs[bge.events.AKEY].values[-1]:
28      rotate += self.turn_speed
29  if inputs[bge.events.DKEY].values[-1]:
30      rotate -= self.turn_speed
31
32  self.object.applyMovement((0, move, 0), True)
33  self.object.applyRotation((0, 0, rotate), True)
```

POWER OF AR AND VR

START THE GAME WITH THE P KEY



UPBGE

POWER OF AR AND VR

**THANK YOU FOR
YOUR ATTENTION**



**Co-funded by
the European Union**



JACEK KAWAŁEK