2017/2018 Physics Question (pictures to prove I worked it out and didn’t just google it)

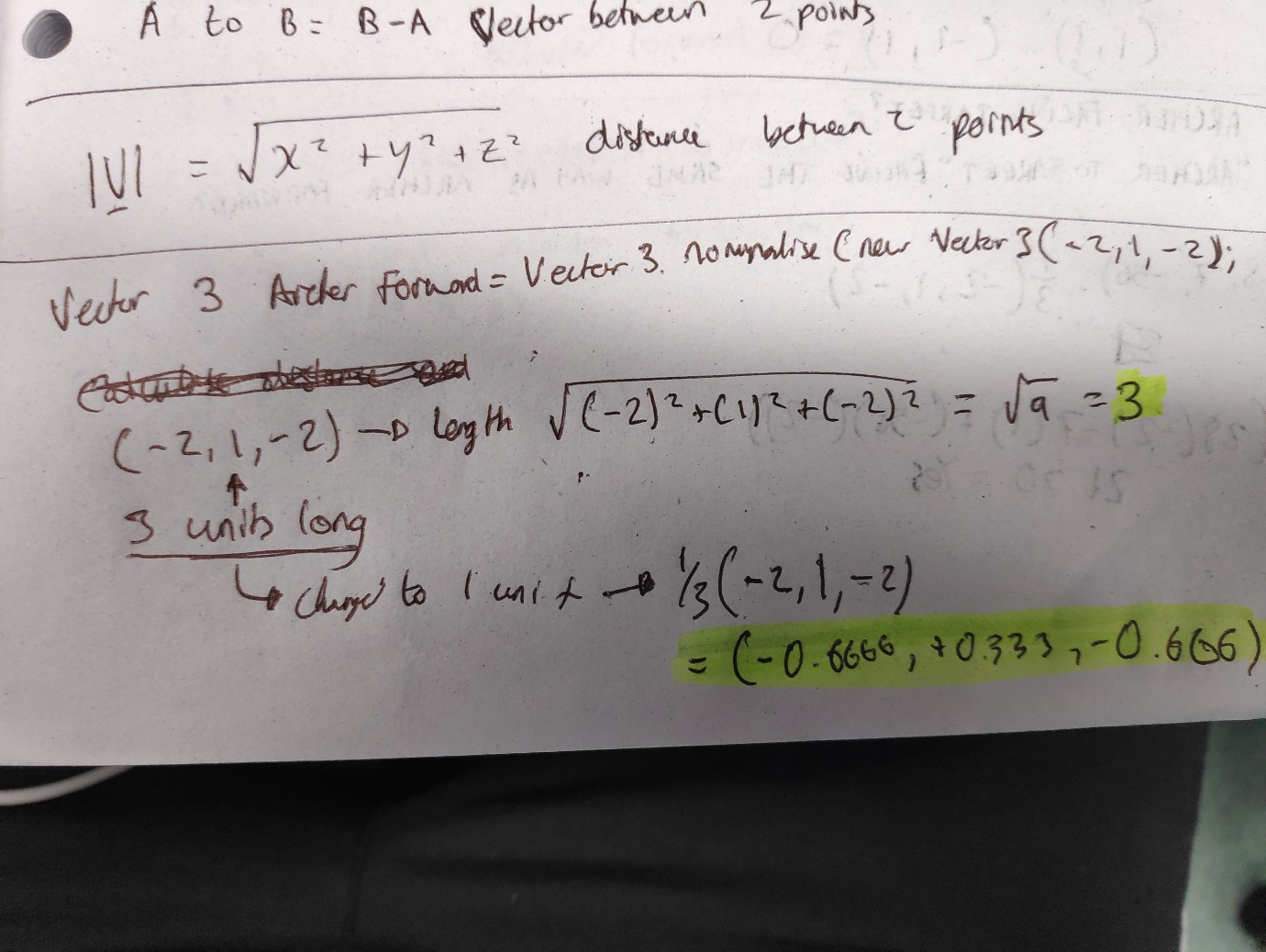
Liam Dowling T00199360

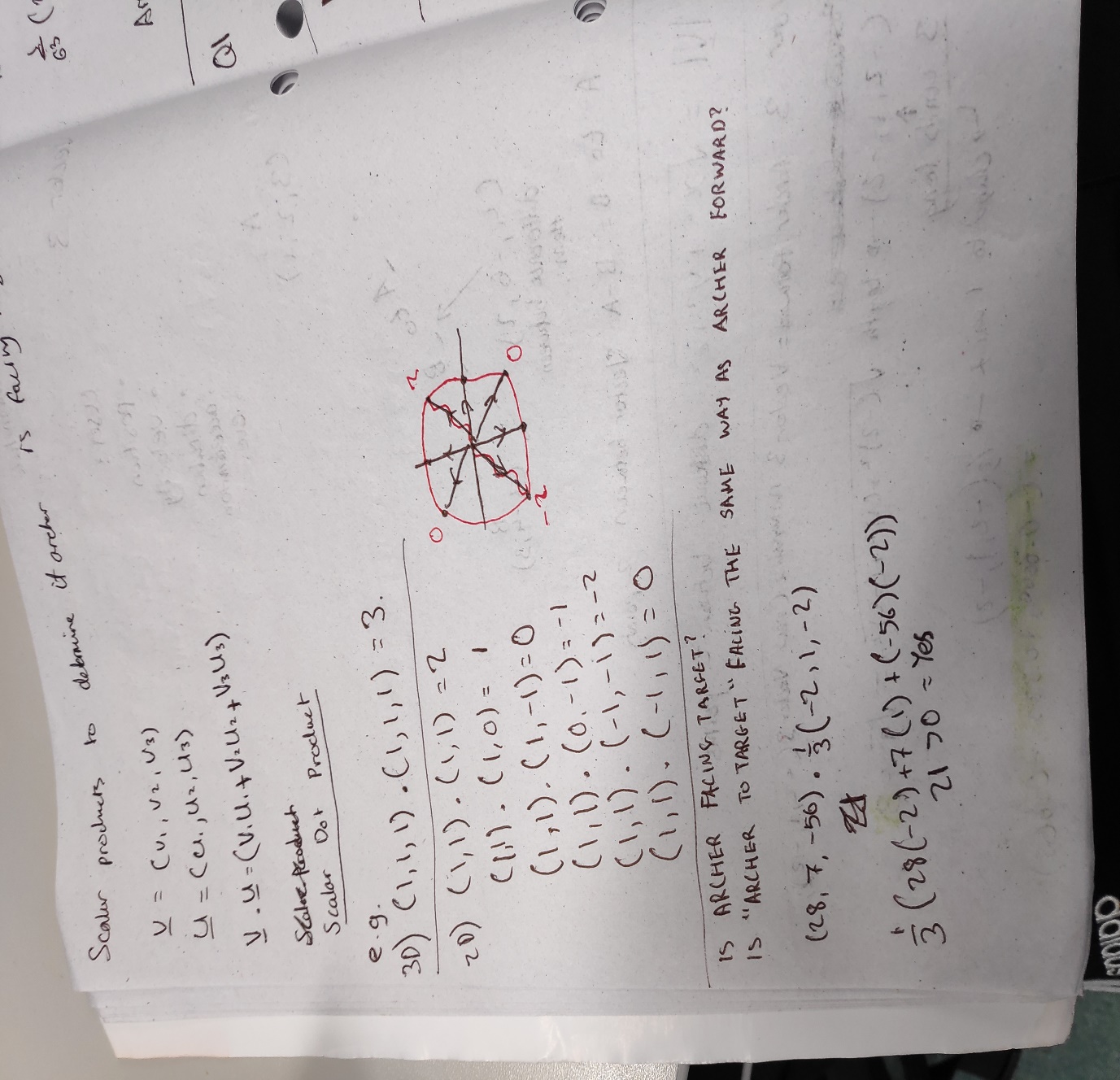
2017/18 Physics Winter exam

1. (i) Vector from archer to target= (28,7,-56)

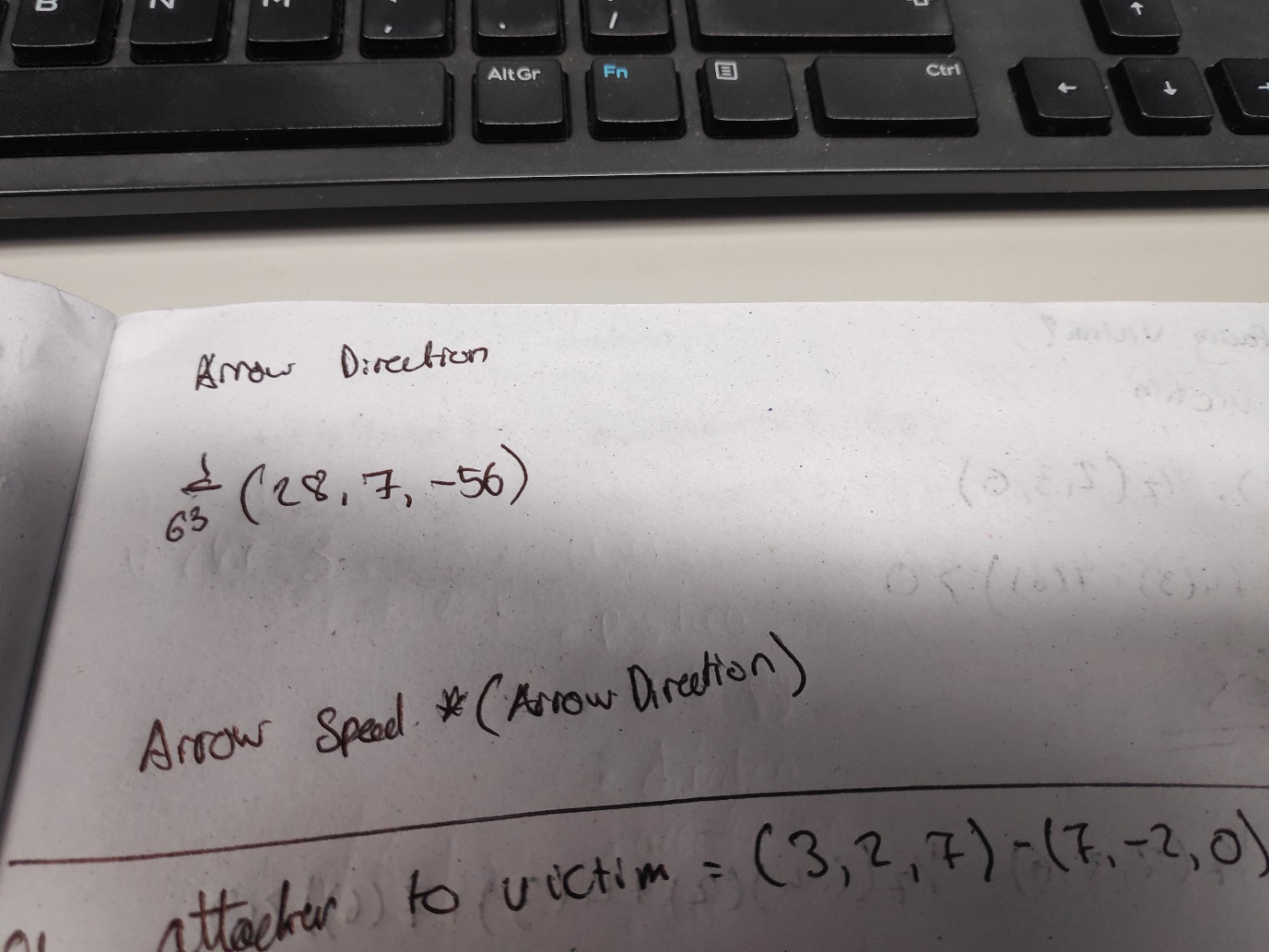
(ii) Distance from archer to target= 63

(iii) Archer Forward = (0-0.6666, 0.333, -0.666)



(iv) Archer is facing the target as 21>0 and 63<80 so yes (workings below at bottom of page)

(vi) Direction of arrow



(vii) Considering that it’s a fixed arrow speed the initial velocity and the final velocity should be the exact same so the arrow speed does not have to depreciate. You could use a ray cast to cast the collision directly to the last effective distance in this case being 80 metres, if no collision is detected, remove the arrow after 80 metres. If there is a collision then doSomething() e.g takeDamage()

1. (i) Use delta time to prevent movement from becoming Frame Rate Dependent. A well known example of games being Frame Rate Dependent for gameplay would be many of the various “Fallout” games. Frame rate independent programs should run at the same speed no matter how fast your device is.

(ii) Delta Time to implement Frame Rate Independence:

targetFrameTime = 33.3f

while game is running

realDeltaTime = time since last frame

gameDeltaTime = realDeltaTime \* gameTimeFactor

// Process inputs

...

update game world with gameDeltaTime

// Render outputs

...

while (time spent this frame) < targetFrameTime

// Do something to take up a small amount of time

...

loop

loop

(iii)

W = Fd cosθ work formula.

If different forces effect an object and propel it different ways it e.g. two grenades go off and blow up a wall, one force propels it left and the other propels it right.

Forces can be added to objects with:

Object.addForce(Vector3.PLACEHOLDER\_DIRECTION \* PREDEFINED\_FORCE, ForceMode.Acceleration);

<https://unity3d.com/learn/tutorials/topics/physics/adding-physics-forces>: referenced as I missed the class that this was covered in