Will Legg
1-19-2024

This is a template for your use to answer the reflection prompt for the week. This is shown below on this page. On the second page, paste images of your whiteboards from Tuesday's class and on the third page paste the images of your whiteboards from Thursday's class. It is entirely okay to use additional pages for your images should you need them. Once the document is complete, save it as a single pdf and upload this to the Week 2 Whiteboard item in gradescope: https://www.gradescope.com/

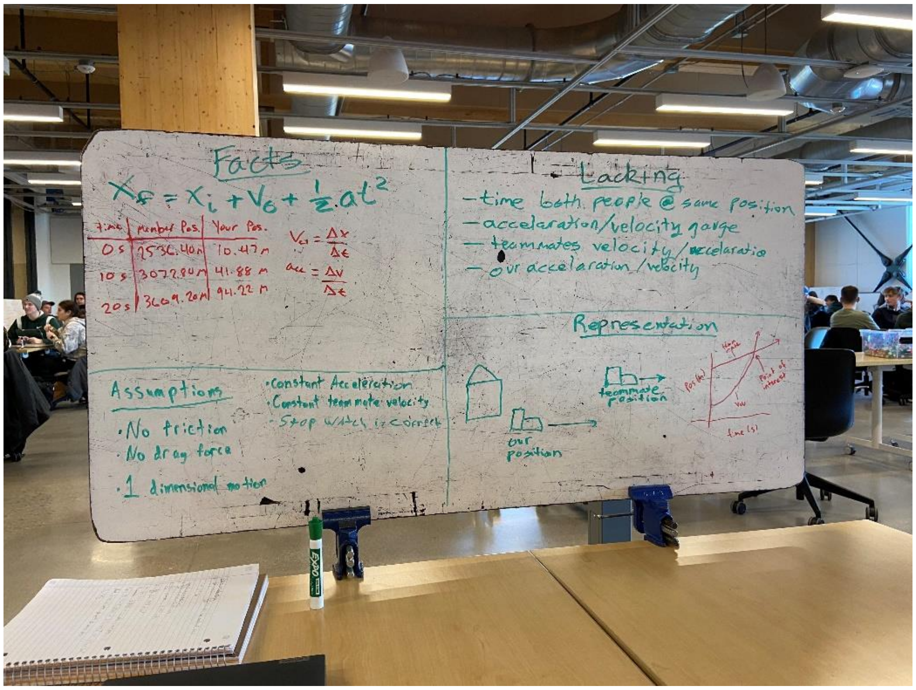
REFLECTION PROMPT: Reflecting on this week in class, were you able to improve on what you said last week? Why or why not? What went well this week? What area(s) could you improve on for next week related to in class and how might you work to improve those next week? What strategies might you try to improve next week? Be specific and include supporting examples from this week's classes. See the "In Class Assessment Rubric" on D2L under the General Course Info folder for more details.
Enter your reflection here

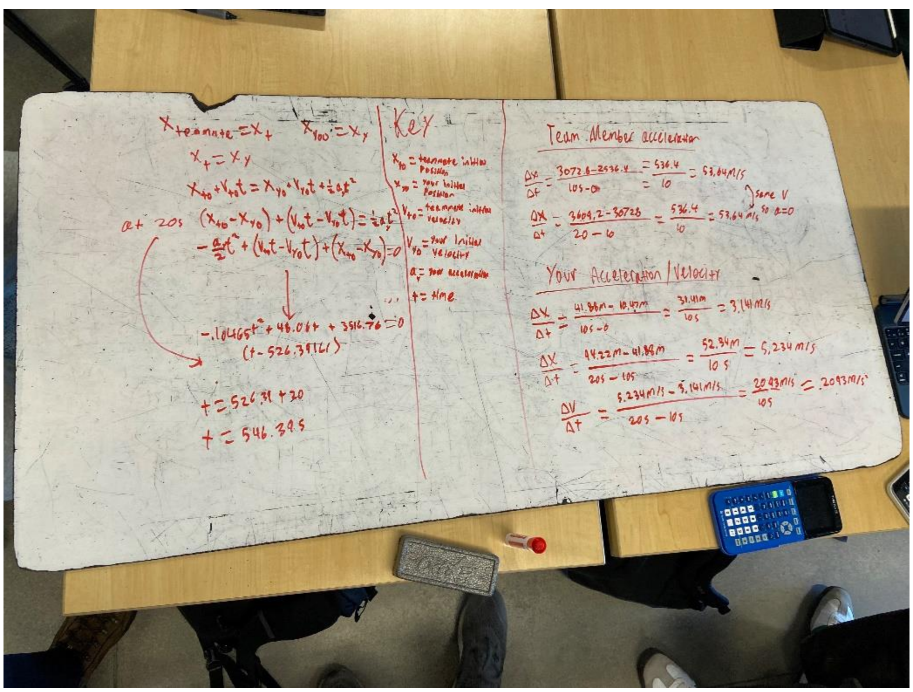
Reflecting on this week in class, were you able to improve on what you said last week? Why or why not?
Our group organization certainly improved from last week. We were able to set up our four quadrants to identify the information that was given to us in the problem, identify what our goal should be, and try to identify which methods/equations would be useful to find the information we needed.

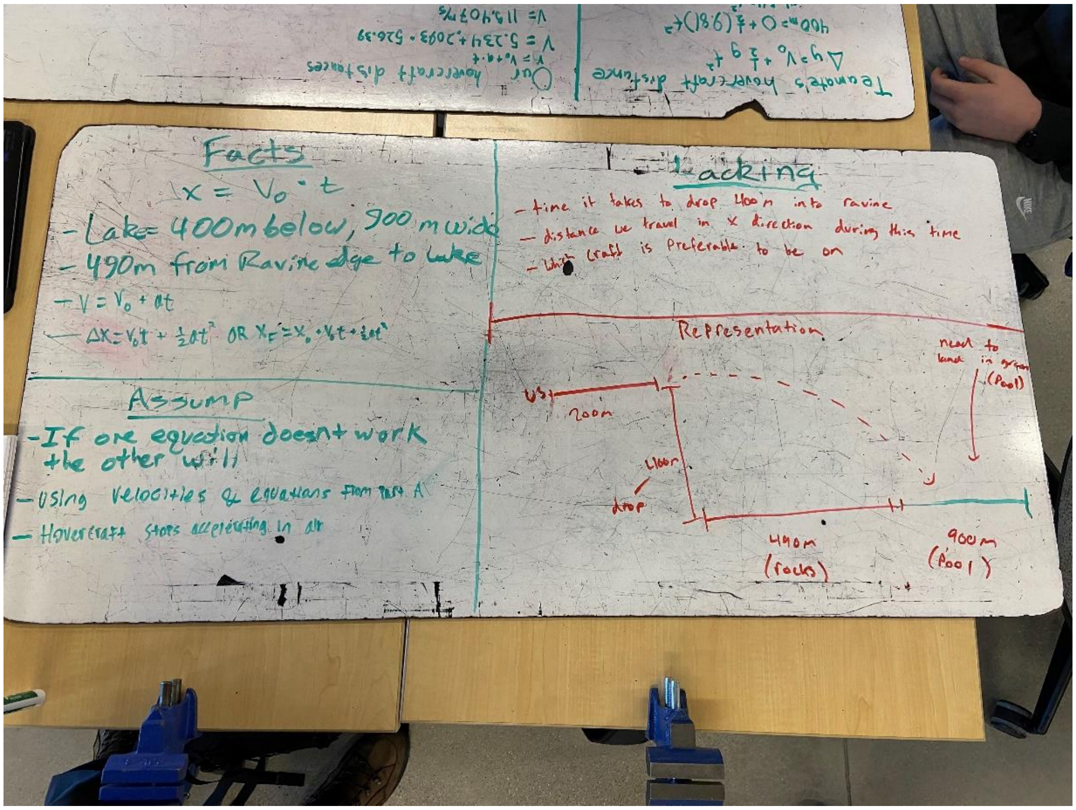
What went well this week?
On part A and B out group quickly identified what information we had to derive from the problem. In part A we determined that we needed to find the time that both hovercrafts were at the same position and identified the most useful equations to determine this. After identifying these elements, it was just a matter of determining velocity/acceleration of both crafts using the given information and using them to find the correct time. Part B was much the same in utilizing equations. I think we did well to divide our work into people who were doing the actual arithmetic with those who were setting up the organization of the problem and guiding the computers to make the correct calculations. In Part C we also divided our workforce to allow the members of the group without programming experience to learn how the program represented the problem we worked out the class before. This allowed different members of the group to focus on different aspects of getting the program to run properly.

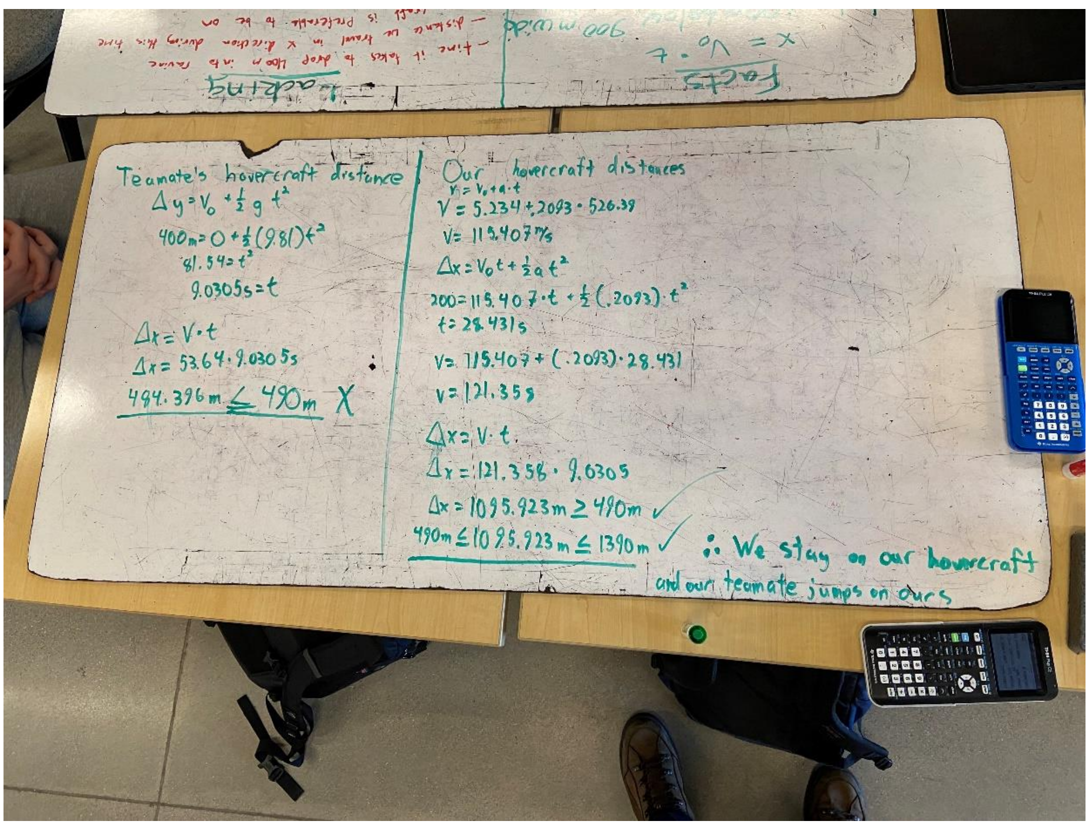
What area(s) could you improve on for next week related to in class and how might you work to improve those next week? What strategies might you try to improve next week?
While our group organization did improve from the week previous in my opinion, we could still get even better moving forward and continue to improve our legibility and workflow. In solving equations, we could also represent our process algebraically before introducing given information to represent our workflow as clearly as possible.

PASTE IMAGES OF YOUR TUESDAY WHITEBOARDS HERE:

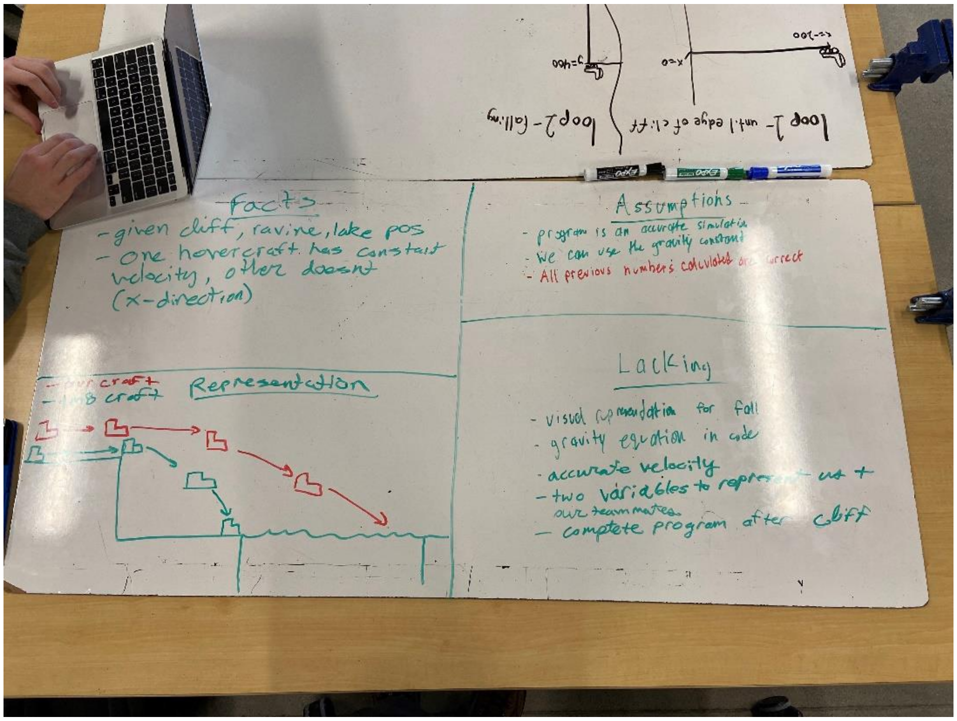


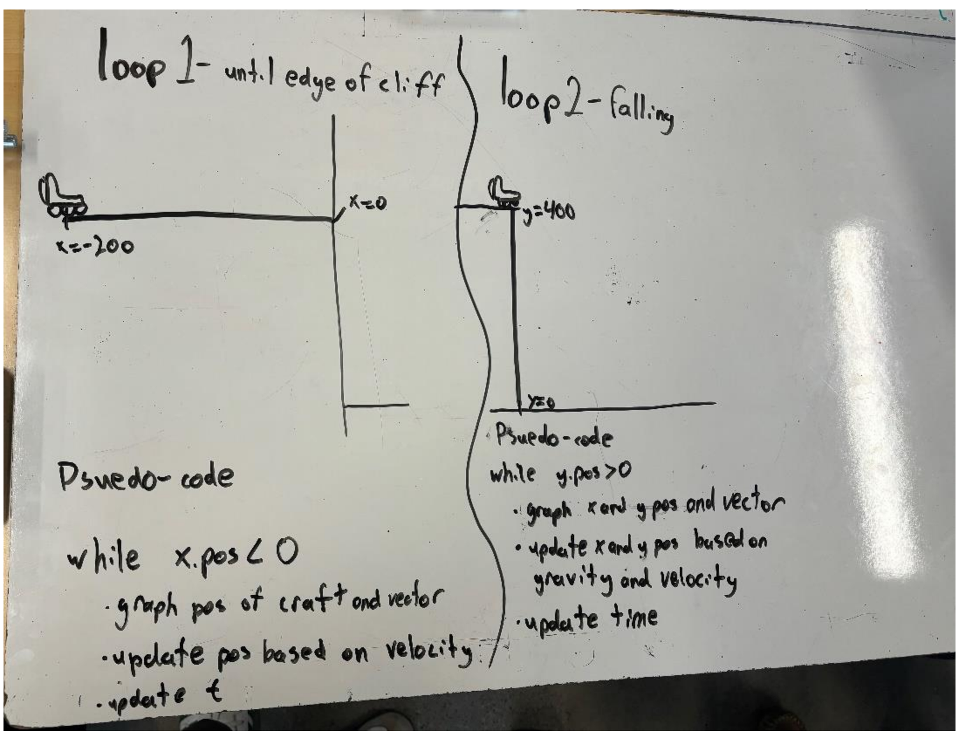



Facts
x = V₀ · t
- Lake = 400m below, 900 m wide
- 490m from Ravine edge to lake
- V = V₀ + at
- Δx = V₀t + ½at² OR X_f = X₀ + V₀t + ½at²
Lacking
- time it takes to drop 400m into ravine
- distance we travel in x direction during this time
- which craft is Preferable to be on
Assump
- If one equation doesn't work the other will
- using velocities & equations from Part A
- Hovercraft stops accelerating in air
Representation
us
200m
400m drop
need to land in system (Pool)
490m (rocks)
900m (Pool)


Teamate's hovercraft distance
Δy = V₀ + ½ g t²
400m = 0 + ½ (9.81) t²
81.54 = t²
9.0305s = t
Δx = V · t
Δx = 53.64 · 9.0305s
484.396m ≤ 490m X
Our hovercraft distances
V = V₀ + a · t
V = 5.234 + .2093 · 526.39
V = 115.407 m/s
Δx = V₀t + ½at²
200 = 115.407 · t + ½ (.2093) · t²
t = 28.431s
V = 115.407 + (.2093) · 28.431
V = 121.358
Δx = V · t
Δx = 121.358 · 9.0305
Δx = 1095.923m ≥ 490m ✓
490m ≤ 1095.923m ≤ 1390m ✓
∴ We stay on our hovercraft and our teamate jumps on ours

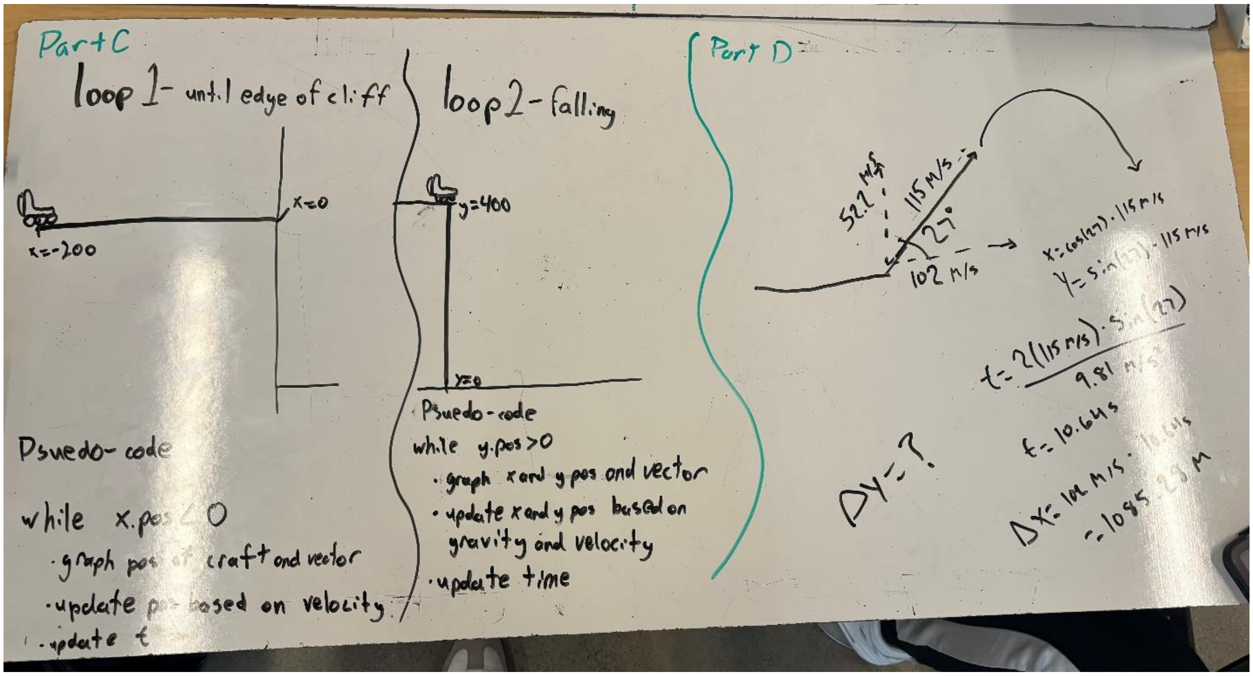
PASTE IMAGES OF YOUR THURSDAY WHITEBOARDS HERE:





```
GlowScript 2.9 VPython

get_library('https://cdn.rawgit.com/PERLMSU/physutil/master/js/physutil.js')

#Window setup
scene.width = 1024
scene.height = 768
scene.center = vector(600,0,0)

#Objects
cliff = box(pos=vector(-100,0,0), size=vec(200,800,0), color=color.white)
ravine = box(pos=vector(245,-200, 0), size=vec(490,400,0), color=color.white)
lake = box(pos=vector(940, -200, 0), size=vec(900,400,0), color=color.blue)
runawaycraft = sphere(pos=vector(-200,400,0), radius=10, color=color.red)
jumpedcraft = sphere(pos=vector(-200,400,0), radius=10, color=color.blue)
#Parameters and Initial Conditions
g = vector(0,-9.81,0)
b = 0  #Drag coefficient

runawaycraftm = 1500
runawaycraftv = vector(53.64,0,0)
runawaycraftp = runawaycraftm*runawaycraftv

jumpedcraftm = 1500
jumpedcraftv = vector(115.407,0,0)
jumpedcraftp = jumpedcraftm*jumpedcraftv
#Time and time step
t=0
tf=20
dt = 0.01

#MotionMap/Graph
runawaycraftMotionMap = MotionMap(runawaycraft, tf, 5, markerScale=1, labelMarkerOrder=False, markerColor=color.orange)
jumpedcraftMotionMap = MotionMap(jumpedcraft, tf, 5, markerScale=1, labelMarkerOrder=False, markerColor=color.pink)

#Calculation Loop for Crashed HoverCraft (on cliff)
while runawaycraft.pos.x<0:
    rate(500)

    Fgrav = runawaycraftm*g
    Fground = -Fgrav
    Fnet = Fgrav + Fground

    runawaycraftp = runawaycraftp + Fnet*dt
    runawaycraft.pos = runawaycraft.pos + (runawaycraftp/runawaycraftm)*dt

    runawaycraftMotionMap.update(t, runawaycraftp/runawaycraftm)
```

```

    t = t + dt


#Calculation Loop for Crashed HoverCraft (falling in the air)
while runawaycraft.pos.y>0:
    rate(500)
    Fgrav = runawaycraftm*g
    Fground = -Fgrav
    Fnet = Fgrav

    runawaycraftp = runawaycraftp + Fnet*dt
    runawaycraft.pos = runawaycraft.pos + (runawaycraftp/runawaycraftm)*dt

    runawaycraftMotionMap.update(t, runawaycraftp/runawaycraftm)

    t = t + dt


#Calculation Loop for Non-Crashed HoverCraft (on cliff)
while jumpedcraft.pos.x<0:
    rate(500)

    Fgrav = runawaycraftm*g
    Fground = -Fgrav
    Fnet = Fgrav + Fground

    jumpedcraftp = jumpedcraftp + Fnet*dt
    jumpedcraft.pos = jumpedcraft.pos + (jumpedcraftp/jumpedcraftm)*dt

    jumpedcraftMotionMap.update(t, jumpedcraftp/jumpedcraftm)

    t = t + dt

#Calculation Loop for Non-Crashed HoverCraft (falling in the air)
while jumpedcraft.pos.y>0:
    rate(500)
    Fgrav = runawaycraftm*g
    Fground = -Fgrav
    Fnet = Fgrav

    jumpedcraftp = jumpedcraftp + Fnet*dt
    jumpedcraft.pos = jumpedcraft.pos + (jumpedcraftp/jumpedcraftm)*dt

    jumpedcraftMotionMap.update(t, jumpedcraftp/jumpedcraftm)

    t = t + dt
```



Part C
loop 1- until edge of cliff
x=0
x=-200
Psuedo-code
while x.pos < 0
·graph pos of craft and vector
·update pos based on velocity
·update t
loop 2- falling
y=400
y=0
Psuedo-code
while y.pos > 0
·graph x and y pos and vector
·update x and y pos based on gravity and velocity
·update time
Part D
52.2 m/s
115 m/s
27°
102 m/s
x=cos(27)·115 m/s
y=sin(27)·115 m/s
t=2(115 m/s)·sin(27) / 9.81 m/s²
t=10.64s
ΔY=?
ΔX=102 m/s·10.64s
=1085.28 m