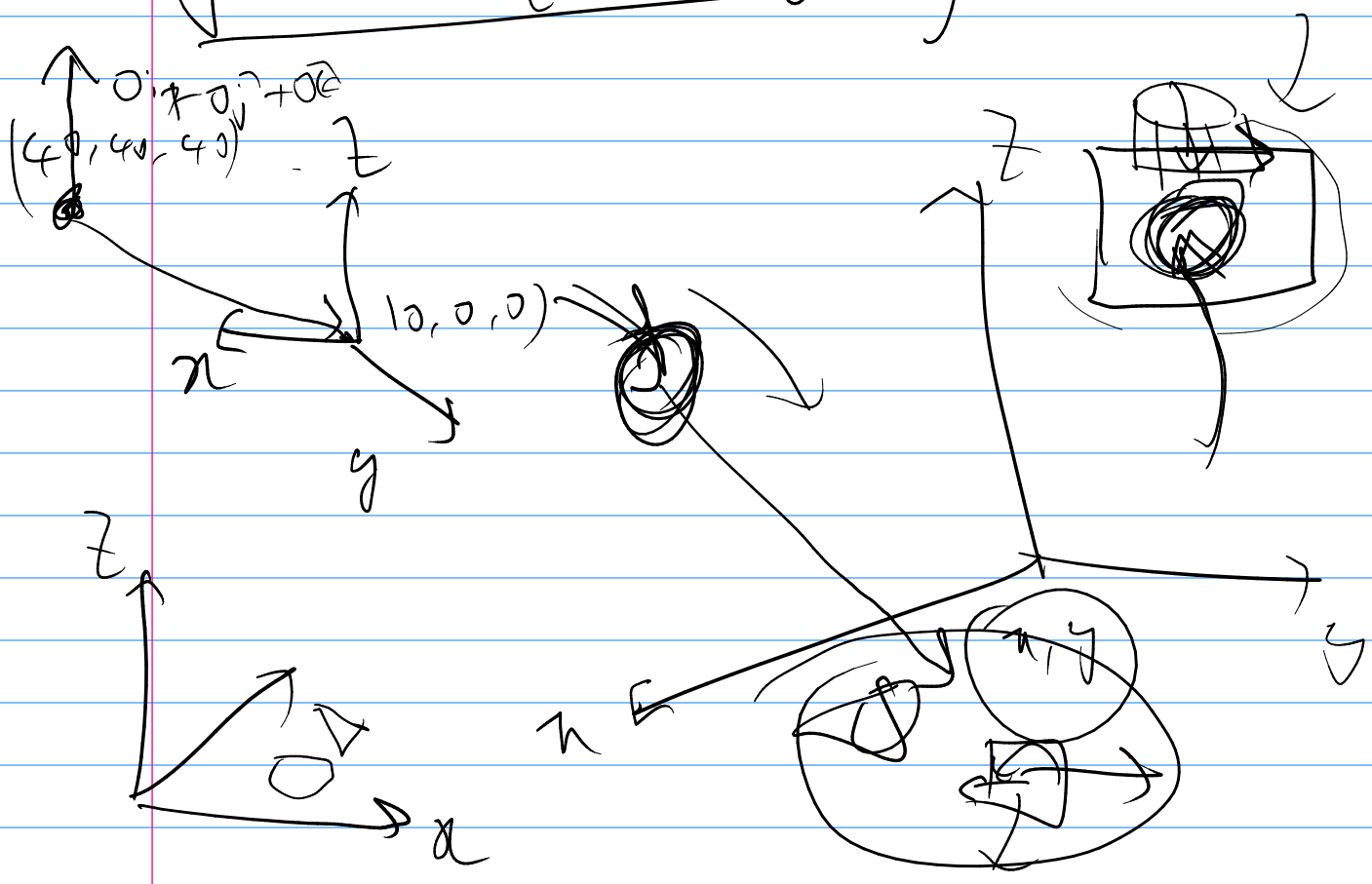
CSE-410 Computer Graphics

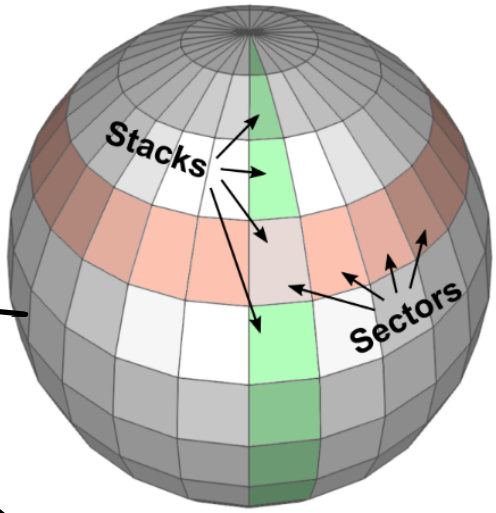
Offline-1 (Task-2) Sessional

No Matrix

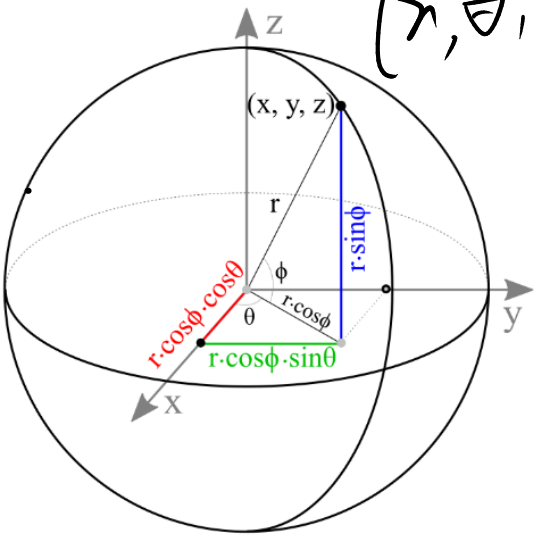
glTranslated(Rx, Ry, Rz)

glRotated(θ, ax, ay, az)





Sectors and stacks of a sphere



A point on a sphere using sector and stack angles

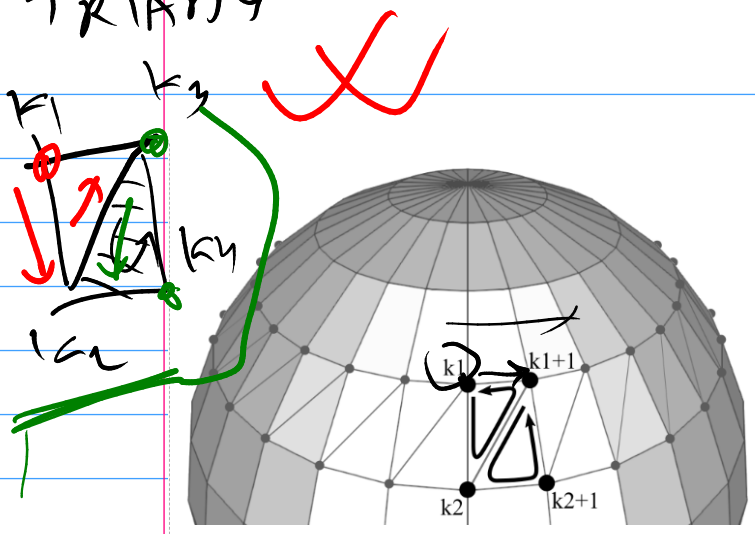
An arbitrary point (*x*, *y*, *z*) on a sphere can be computed by parametric equations with the corresponding sector angle θ and stack angle ϕ.

$$
\begin{aligned}
x &= (r \cdot \cos\phi) \cdot \cos\theta \\
y &= (r \cdot \cos\phi) \cdot \sin\theta \\
z &= r \cdot \sin\phi
\end{aligned}
$$

The range of sector angles is from 0 to 360 degrees, and the stack angles are from 90 (top) to -90 degrees (bottom). The sector and stack angle for each step can be calculated by the following;

$$
\begin{aligned}
\theta &= 2\pi \cdot \frac{\text{sectorStep}}{\text{sectorCount}} \\
\phi &= \frac{\pi}{2} - \pi \cdot \frac{\text{stackStep}}{\text{stackCount}}
\end{aligned}
$$

TRIANG



vertex indices to draw triangles of a sphere

In order to draw the surface of a sphere in OpenGL, you must triangulate adjacent vertices to form polygons. It is possible to use a single triangle strip to render the whole sphere. However, if the shared vertices have different normals or texture coordinates, then a single triangle strip cannot be used.

Each sector in a stack requires 2 triangles. If the first vertex index in the current stack is *k1* and the next stack is *k2*, then the counterclockwise orders of vertex indices of 2 triangles are;
k1 ⟶ *k2* ⟶ *k1+1*
k1+1 ⟶ *k2* ⟶ *k2+1*

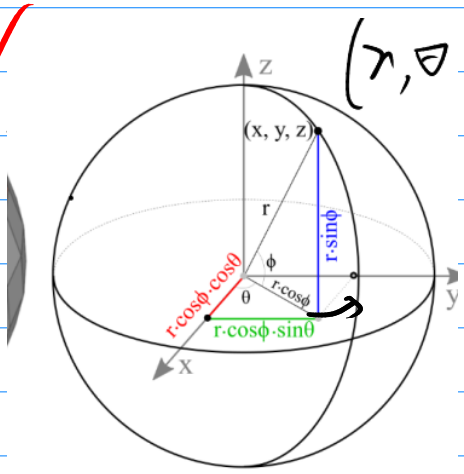
But, the top and bottom stacks require only one triangle per sector. The code snippet to generate all triangles of a sphere may look like;

for(i=0; i<sect; i++)

for(j=0; j<stack; j++)

$\theta_1 = 2\pi \frac{i}{sect}$ $\phi_1 = j$

$\theta_2 = 2\pi \frac{(i+1)}{sect}$ ϕ $j+1$



pushM()

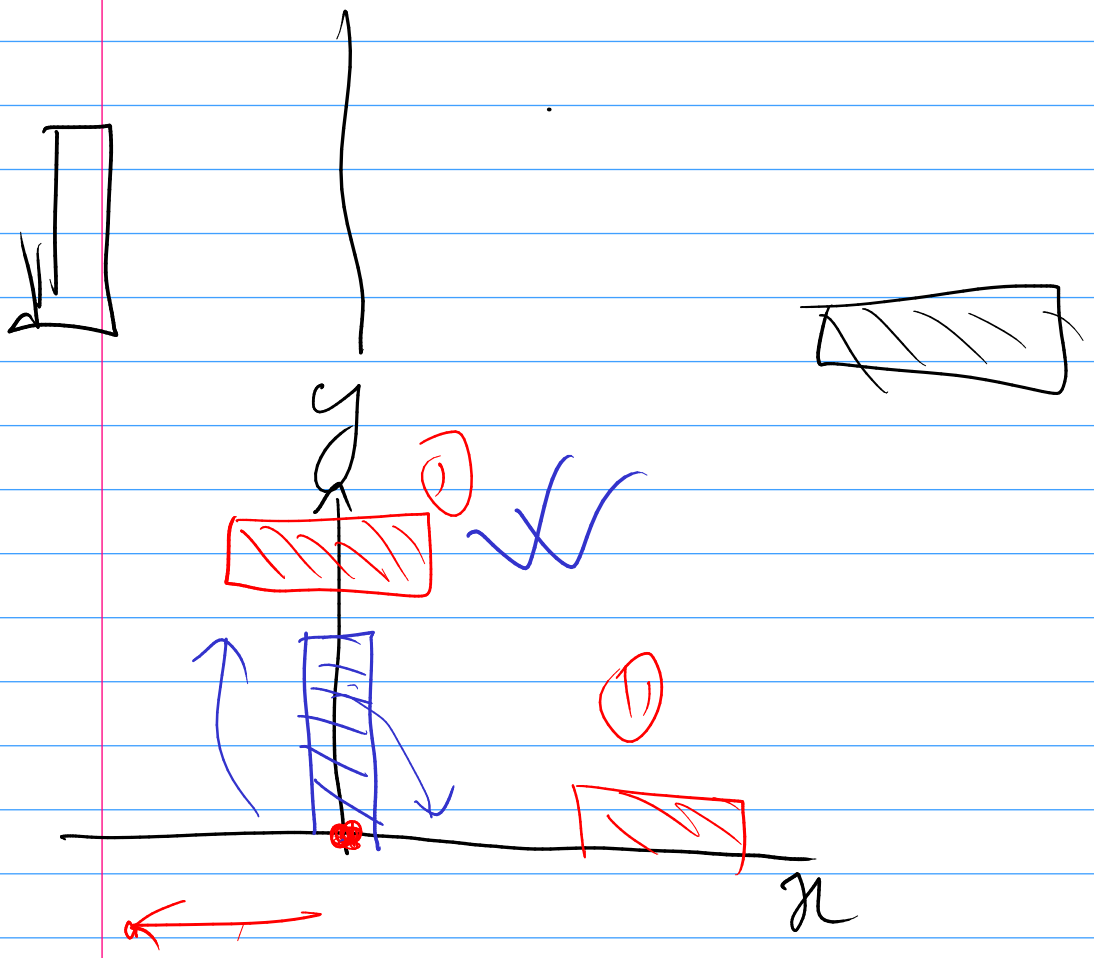
Trans

Rot

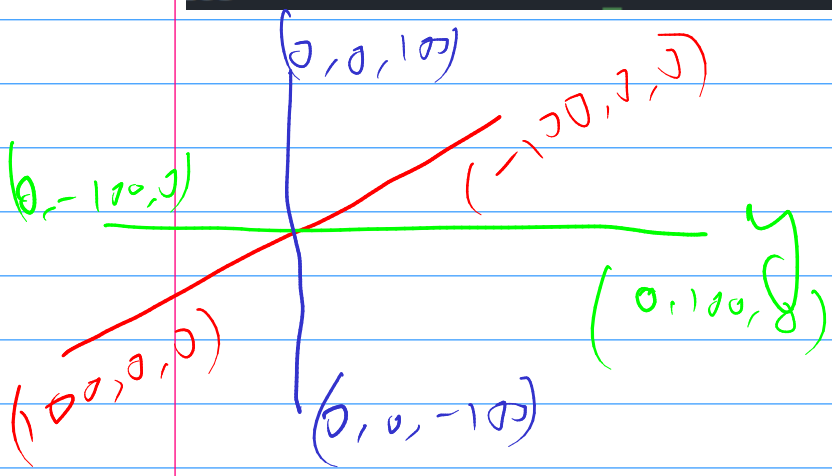
Sphere()

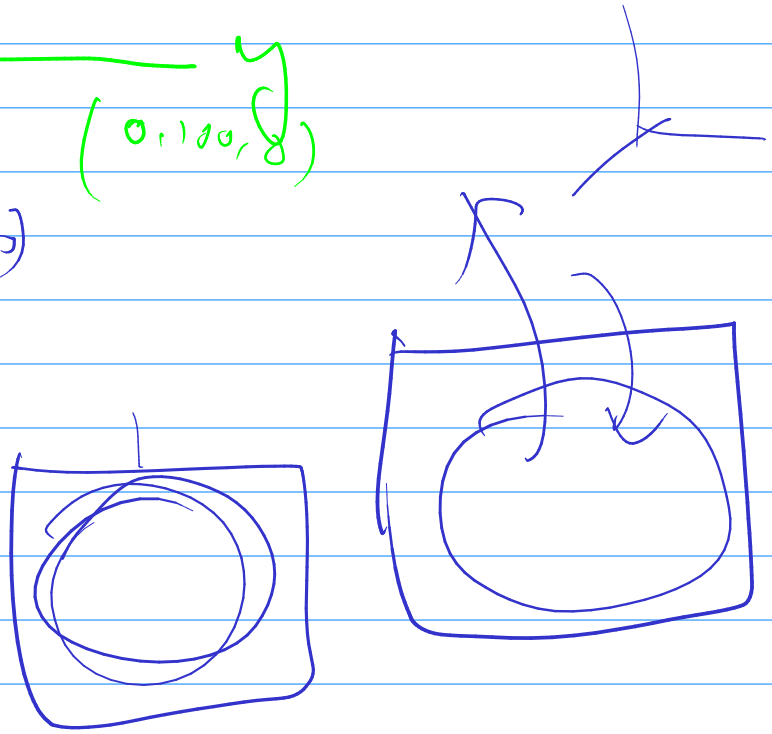
popM()

(cosθ, sinθ, 0)


y
x
Rotate → Translate
Translate → Rotate

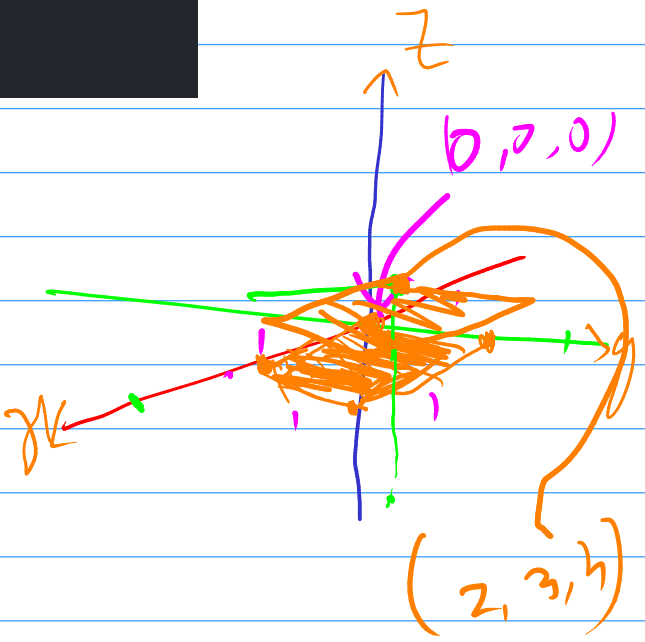
```
188  void axes() {
189    glBegin(GL_LINES);
190    {
191      glColor3f(1.0f, 0.0f, 0.0f); // Red - x axis
192      glVertex3f(-100, 0, 0);
193      glVertex3f(100, 0, 0);
194
195      glColor3f(0.0f, 1.0f, 0.0f); // Green - y axis
196      glVertex3f(0, -100, 0);
197      glVertex3f(0, 100, 0);
198
199      glColor3f(0.0f, 0.0f, 1.0f); // Blue - z axis
200      glVertex3f(0, 0, -100);
201      glVertex3f(0, 0, 100);
202    }
203    glEnd();
204  }
205
```

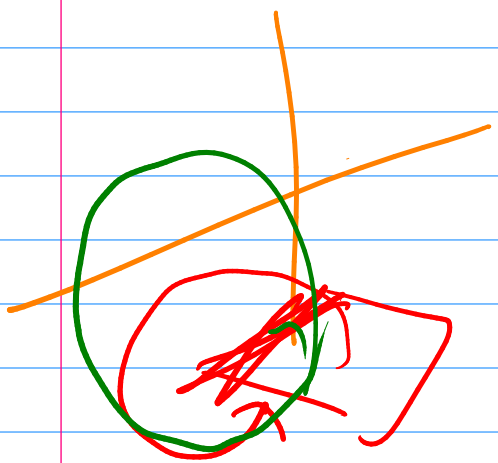




```
void drawSquare(double a) {
  glBegin(GL_QUADS);
  {
    glVertex3f(0, 0, 0);
    glVertex3f(0, a, 0);
    glVertex3f(a, a, 0);
    glVertex3f(a, 0, 0);
  }
  glEnd();
}
```

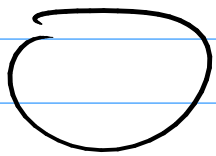
glTranslate(2, 3, 4)

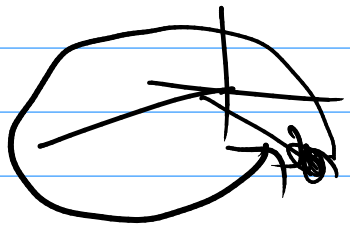


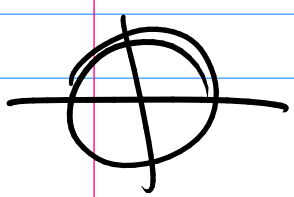


① Rotate – Trans

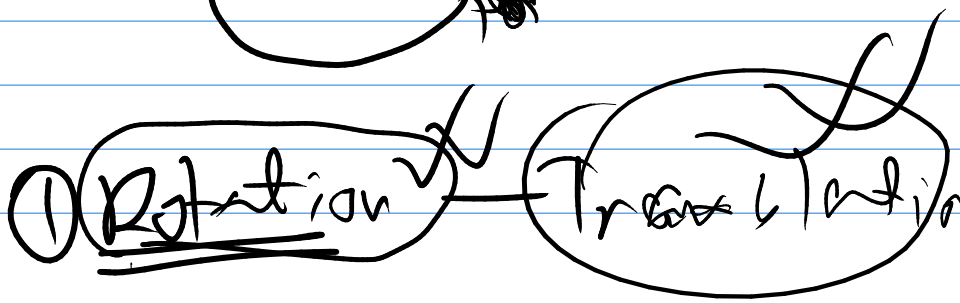
② Translate – Rotate

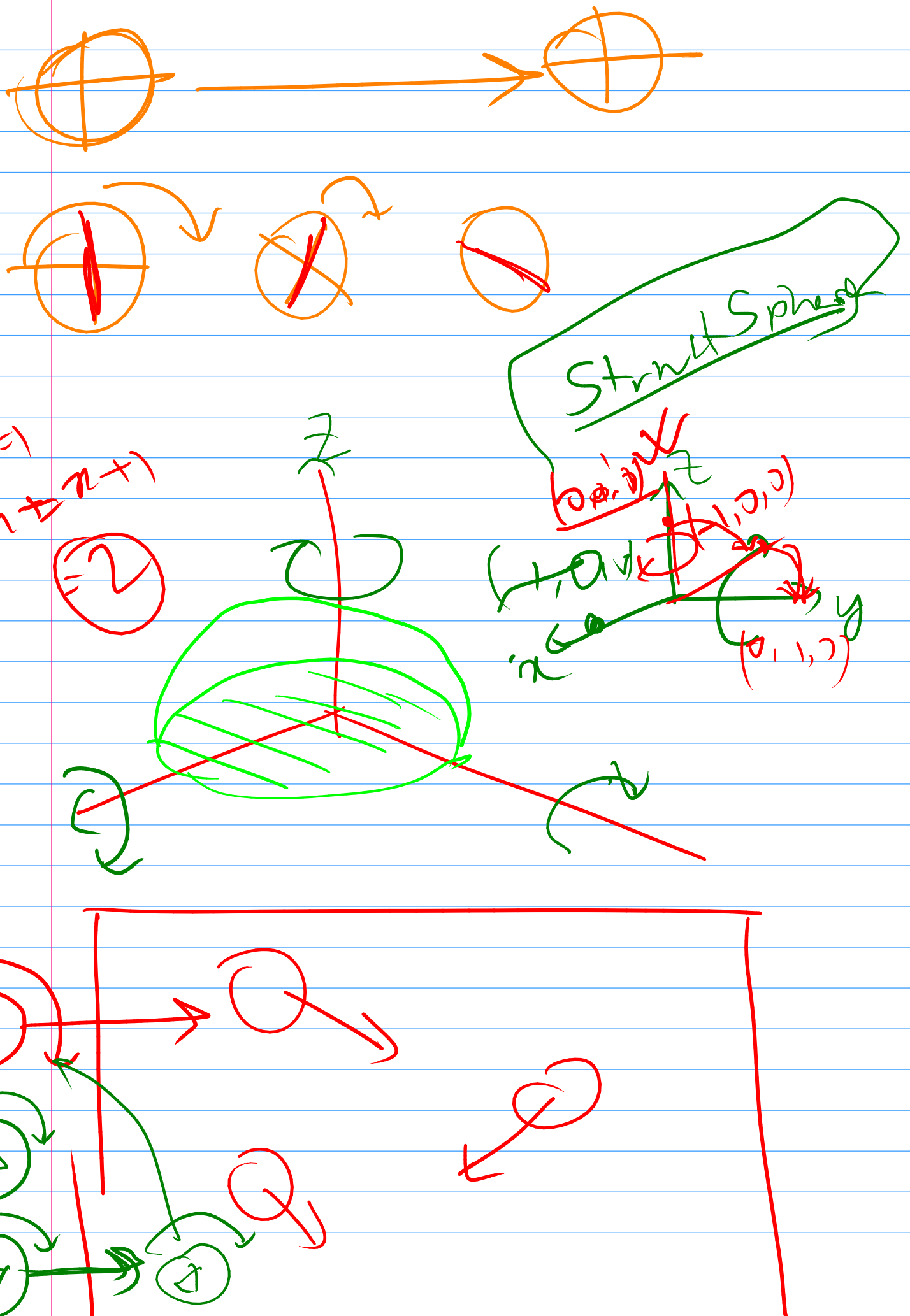


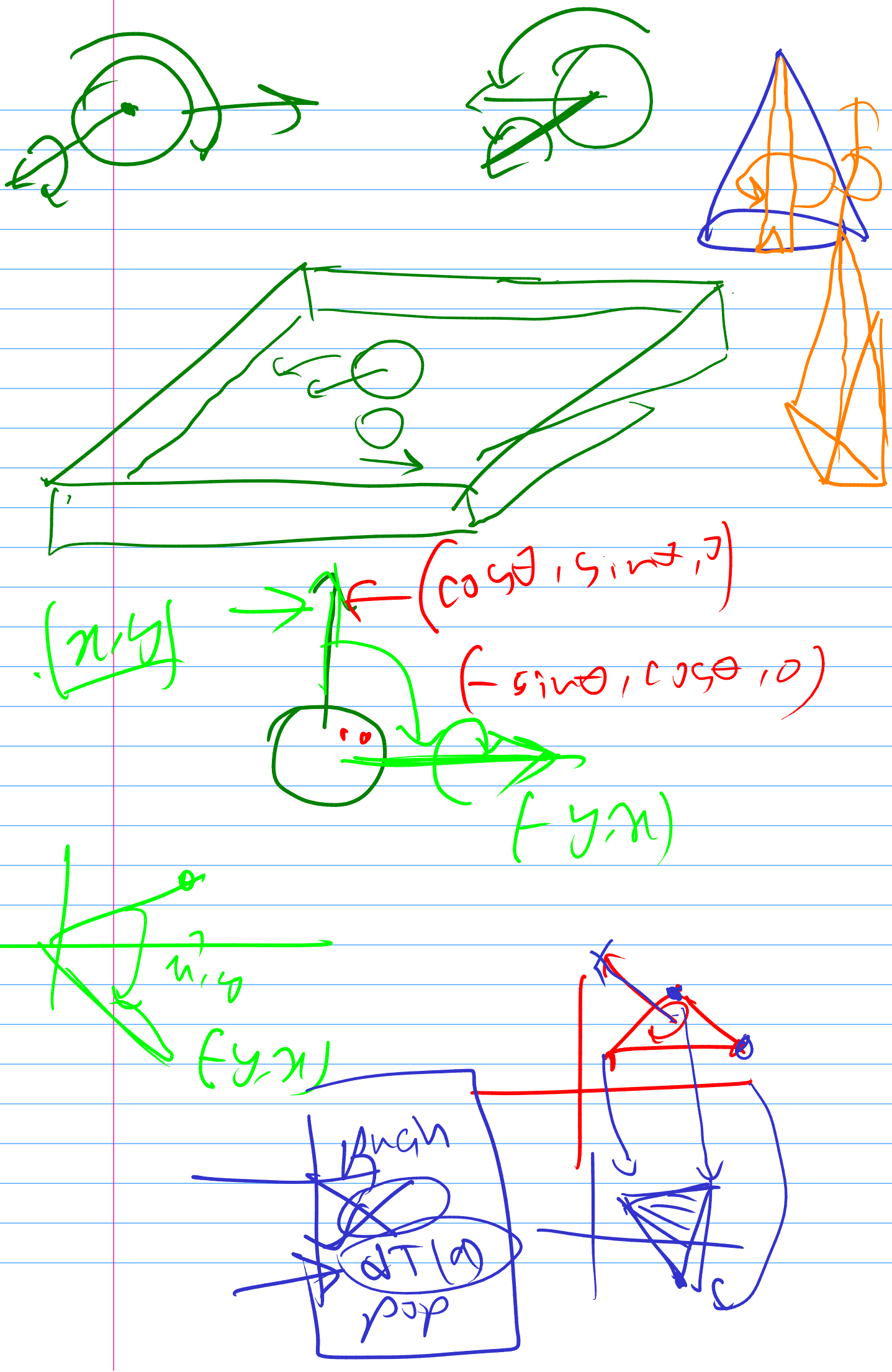


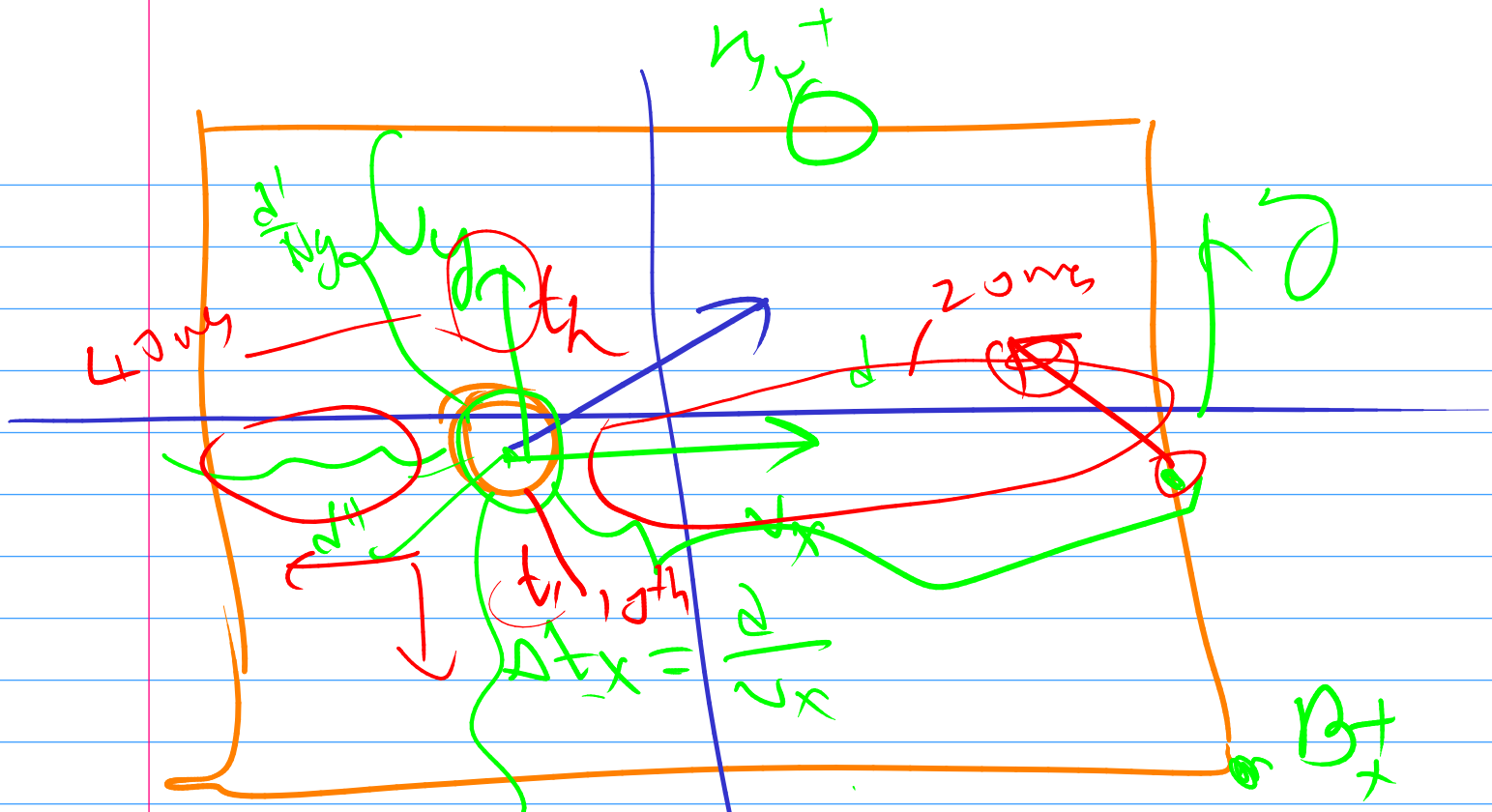


① Rotation — Translation




Strict Sphere
z
y
x
(0,0,1)
(1,0,0)
(0,1,0)


(cosθ, sinθ, 1)
(x, y)
(−sinθ, cosθ, 0)
(−y, x)
x, y
(−y, x)
Bug
dT/dθ
pop



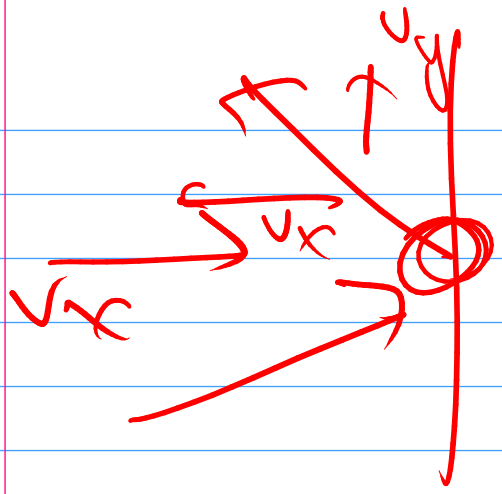
$T + \Delta t_1 = t_1$

$T + \Delta t_2 = t_2$

$T = 10\,ms$

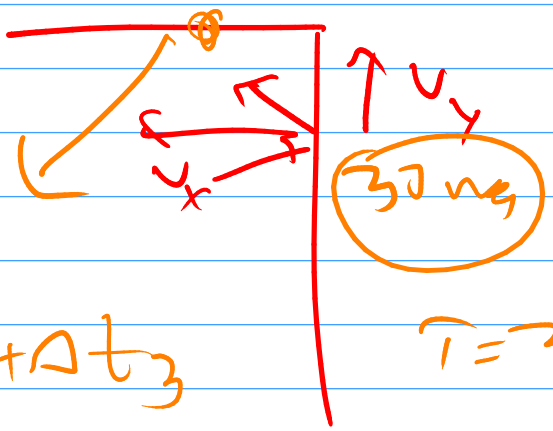
$t_1 \rightarrow 30\,ms$

$t_1 - T = 20\,ms$



collcount = 1

$\Delta t_3 = 2$ ms



$T = 30$ ms

$t_3 = T + \Delta t_3$

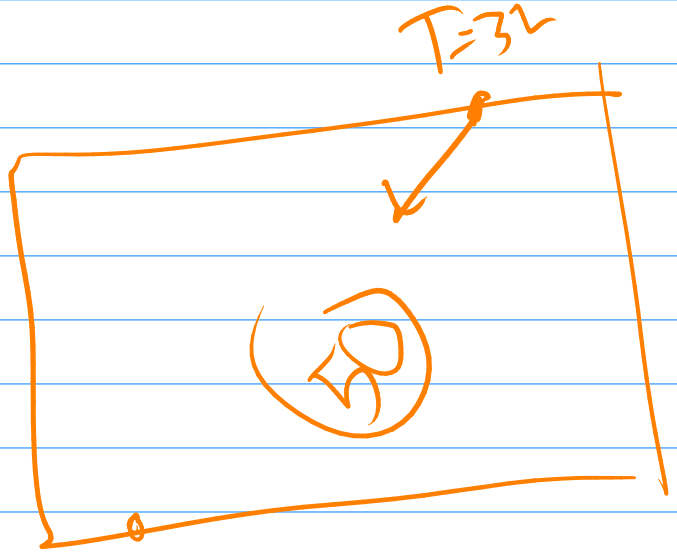
$= 32$ ms

PV: $\langle t_2 \rangle$

$\langle t_2 \rangle$ 50

PV $\langle t_3 \rangle$ 32

cc = 2



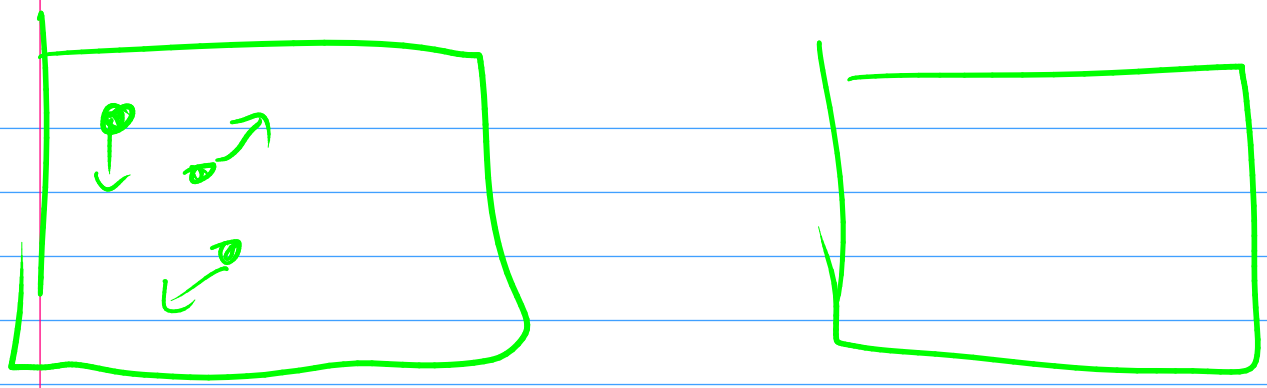
$t_4 = 82$

PV: $\langle t_2 \rangle$ 50 $\langle t_4 \rangle$ 82

0

cc = 2

cc = 0



bcc=00

cc=1

Free-vation

- Schedule-Action

offline X

Schedule-check-Action