PHYSICS 113 – Recitation Assignment 6

Name

Recitation Assignment # 6

Oct. 25, 2006

You may complete this in class. However, if you are unable to do so, it is expected that you complete this for recitation next time.

Where a box appears, call over Travis to check over your progress. When the sheet is complete, you will hand it in. Also, you are expected to email your final programs to Travis.

Today: Galilean and (maybe) Lorentz Transforms

Today's recitation is in (up to) 2 parts. The first part is required, and should be emailed to Travis. The second part is extra credit, and should be mailed to me.

If you successfully complete the extra credit, you will get up to 10 extra points added on to your midterm! (However, nobody will be allowed to get above 100 – sorry about that.)

For today's assignment, you will need to know two new VPython commands:

- `scene1=display(title="rest frame",range=20,autoscale=0)`

  creates a new window called "scene1". (By default, the window is called "scene"). You can create more than 1 scene. The other scene used in this example will be a moving frame called "scene2".

  You only need to create a scene once in order to use it.

- `scene1.select()`

  Which says that everything after this line should happen in the "scene1" window. You will use this command a lot to switch between your two windows.

Assignment Part 1: Galilean Relativity (Required)

1. Create "scene1". In it, put a yellow sphere (called "ball1") at the origin. The ball should have an initial velocity of $14m/s\hat{j}$, a mass of $2kg$, and a radius of 0.5.

□ 2. Evolve the position and velocity of "ball1" for 10 seconds (assuming the ball is near the surface of the earth), with $dt = 0.01$ sec. Create a yellow trail behind it. Use rate(100).

3. Now, create a second "scene2", and put a ball (also at the origin) called ball2 in it. Pretend that scene2 is moving to the left at $\vec{V} = -5$ m/s. During the evolution part of the code, you should compute the position of the second ball using the Galilean transform relation:

$$\vec{r}_2 = \vec{r}_1 + \vec{V}t$$

   Notice a couple of things. First, the transform uses the total time since the beginning of the simulation (not dt). Secondly, you are NOT to use the $v = v + a \times dt$ relation in the second window. The dynamics are to be determined solely from the transform.

4. Make a green path behind the second ball.

5. To make things easier to see, leave a sphere of radius 0.2 (you can just add a sphere to the scene by issuing a sphere() command) every 0.25 seconds. A handy way to check for this is:

   ```
   if (int((t+dt)/0.25)-int(t/0.25) == 1):
   ```

   Do this for both scenes.

□ 6. Finally, compute analytically how long the code *should* run so that it stops when $y = 0$. You know the projectile motion equations. Simply determine (based on the initial velocity), at what time this should be.

Assignment Part 2: Special Relativity (Extra Credit)

This assignment is tough. I expect that you'll want clarification. Please ask Travis or myself.

Once again, you will be creating a series of events in two windows. In this case, it will be a spacetime diagram.

You may work in teams of two. However, it should be clear to Travis and myself that both students contributed to the work. I may quiz you on how it works.

1. In scene1, you should have a "stationary frame":

   ```
   scene1=display(title='Unprimed Frame',x=1,y=1,width=600,height=600,#
                          center=(0,0,0),range=(10,10,10),autoscale=0)
   ```

   In this, you should create two axes (time and space), and label them appropriately. You should then evolve four events:

   (a) The back of a space-ship at x=0.2

   (b) The front of a spaceship at x=1.2

   (c) A projectile launched from the back of the spaceship and moving toward the right at $0.6c$ (all of your speeds should be in units of the speed of light, and thus, all of your distances will be in ls). When the projectile strikes the front of the ship, it should bounce back with the same speed.

   (d) A student (you) who starts at the origin and heads to the right at the speed 0.8c.

   At every timestep, you will append the "worldlines" of all 4 of these things. The ship (according to me) is stationary, and thus, the front and back will move in time, but not in space. The projectile and student will move in both.

2. Now create a second window and do the transform of this at each "timestep." The primed frame should be moving toward the right at 0.8c (and thus, in this frame you'll know if you're on the right track if the student [you] appears to be moving in time but not space).

   In order to do the transforms, remember:

$$\begin{aligned} \Delta x' &= \gamma \Delta x - v\gamma \Delta t \\ \Delta t' &= \gamma \Delta t - v/c^2 \Delta x \end{aligned}$$

Also, since everything is in units of c, sec., and light-seconds, the number 3×10^8 should NOT appear in your code.

Finally, these Δt, Δx, and so on are compared to the origin, so you could just as easily replace Δx with x, for example.

What do the events look like to the moving observer?

What happens if you change the speed of the projectile to c?