Proxima Centauri

This is a simulated mission from LEO (altitude approximately 255 miles) to Proxima Centauri. The mission has humans aboard, so the max acceleration is 1.0g. This will be simulated on a Raspberry Pi using an eInk screen and a battery backup, running in real time as if it was the ship's display. I've wanted to do something like this for several years, and now I've decided to do it.

The journey takes part in three phases:

- Acceleration from v_0 to v_t with a maximum acceleration a.
- Coasting at v_c until we reach the deceleration distance d_d.
- Deceleration from v_c to v_e, the in-system cruising speed.

This project doesn't consider masses or anything else, just velocities and distances, as well as relativistic time passing. It also glosses over the technology behind the engines. For example, using some very ballpark numbers, to push the space shuttle (which weighs 81,193 kg) at that speed:

```
In[•]:= F = Quantity[81193, "Kilograms"] * Quantity[9.800665, "Meters" / ("Seconds")^2]
```

```
Out[•]= 795745. kg m/s^2
```

The most powerful rocket engine right now is SpaceX's Raptor engine, which produce 3.5MN of thrust.

```
In[•]:= F_raptor = Quantity[3.5, "Meganewtons"]
```

```
Out[•]= 3.5 MN
```

```
In[•]:= n_engine = Ceiling[F / F_raptor]
```

```
Out[•]= 1
```

So a single engine at a minimum to push the space shuttle at 1G. These have a combined LOX / LCH_4 flow rate of 931.2 kg/s:

```
In[•]:= FlowRate[n_] := Quantity[931.2, "Kilograms" / "Seconds"] * n
```

```
In[•]:= FlowRate[n_engine]
```

```
Out[•]= 931.2 kg/s
```

Later on, we'll see that the acceleration phase alone takes ~354 days, which means just to get the space shuttle going, we'll need approximately

```
In[•]:= m_fuel = FlowRate[n_engine] * Quantity[354, "Days"]
```

```
Out[•]= 2.84813 × 10^10 kg
```

```
In[•]:= UnitConvert[Quantity[1.70888 × 10^11, "Kilograms"], "Tons"]
Out[•]= 1.88372 × 10^8 sh tn
```

For comparison, the external tank on the space shuttle could carry 106,261kg of LH_2 and 629,340 kg of LOX.

```
In[•]:= m_et = Quantity[106 261, "Kilograms"] + Quantity[629 340, "Kilograms"]
Out[•]= 735 601 kg
```

```
In[•]:= m_fuel / m_et
Out[•]= 38 718.4
```

We'd need the equivalent of about 40,000 external fuel tanks to pull this off using our present technology, at least if I did the math right.

```
In[•]:=
```

Basic Kinematics

There are four basic kinematics equations.

1. $v = v_0 + a*t$
2. $v^2 = v_0{}^2 + 2\,a(\Delta x)$
3. $x = x_0 + v_0\,t$
4. $x = x_0 + v_0\,t + \frac{a*t^2}{2}$

```
In[•]:= Acclerate[x₀_, v₀_, a_, t_] := {x₀ + v₀ * t + (a * t^2)/2, v₀ + a * t}
```

Acceleration Phase

We know our maximum acceleration and the altitude at which the ship was assembled.

```
In[•]:= a = Quantity[9.800665, "Meters" / ("Seconds")^2]
Out[•]= 9.80067 m/s^2
```

```
In[•]:= r = UnitConvert[Earth PLANET [equatorial radius], "Meters"] + UnitConvert[Quantity[255, "Miles"], "Meters"]
Out[•]= 6.788520 × 10^6 m
```

In[•]:= μ = 6.674*^-11 m^2N/kg^2 * UnitConvert[Earth PLANET [mass], "Kilograms"]

Out[•]= $3.98585 \times 10^{14}\ m^2N/kg$

In[•]:= $v_0 = \sqrt{\frac{\mu}{r}}$

Out[•]= $7662.54\ \sqrt{m}\ \sqrt{N}/\sqrt{kg}$

The escape velocity is somewhat higher. For the sake of a simplified simulation, we'll assume we've gotten to escape velocity instead.

In[•]:= v_{escape} = UnitConvert[Earth PLANET [escape velocity], "Meters"/"Seconds"]

Out[•]= $1.118 \times 10^4\ m/s$

The distance to Proxima Centauri is 4.247 light years from Earth, give or take a few hundredths of a light year.

In[•]:= d_{ly} = Quantity[4.247, "LightYears"]

Out[•]= $4.247\ ly$

In[•]:= d = UnitConvert[d_{ly}, "Meters"]

Out[•]= $4.01797 \times 10^{16}\ m$

We'll need to get to a target velocity of 0.999*c*.

In[•]:= c = speed of light in m/s

Out[•]= $299\,792\,458\ m/s$

In[•]:= v_{cruise} = 0.999 * c

Out[•]= $2.99493 \times 10^8\ m/s$

How long will it take us to get to our target velocity?

In[•]:= t_a = UnitConvert[$\frac{(v_{cruise} - v_{escape})}{a}$, "Days"]

Out[•]= $353.672\ days$

After nearly a year, we will be at our cruising velocity and the ship will begin rotating to provide gravity. At this point, we will have covered

In[•]:= $d_{accel} = (v_{escape} * t_a) + \frac{1}{2}(a * t_a^2)$

Out[•]= 4.57601×10^{15} m

or

In[•]:=
```
UnitConvert[d_accel, "LightYears"]
```

Out[•]= 0.483685 ly

Deceleration phase

The cruising phase is the gap between the acceleration and the deceleration phases, so the next task is to figure out when to start decelerating. The first step in this task is to figure what our final velocity should be to explore Proxima Centauri. As an assumption, three months to cross one Sol-standard astronomical unit (AU) should be a fast enough velocity to explore the system, but slow enough to not just blow past everything. The exploration velocity is then

In[•]:=
```
v_explore = UnitConvert[
  Quantity[1, "AstronomicalUnit"]/Quantity[3, "Months"], "Meters"/"Seconds"]
```

Out[•]= $\frac{6\,830\,953}{360}$ m/s

How long does it take to get up to speed? We know that $v = v_0 + a * t$, so it follows that $t = (\frac{v - v_0}{a})$.

In[•]:= $t_{accel} = \text{UnitConvert}\left[\frac{(v_{cruise} - v_{escape})}{a}, \text{"Days"}\right]$

Out[•]= 353.672 days

We next need to figure out how far it will take to decelerate to our exploration speed. How long does it take to decelerate from cruising velocity to the exploration velocity? Again, we assume that one gee is the maximum acceleration; we can simulate this by assuming the ship flips around and fires its engines in the reverse direction. We'll want to position ourselves within about 5 AU of the star.

In[•]:= $t_{decel} = \frac{(v_{explore} - v_{cruise})}{-a}$

Out[•]= 3.05565×10^7 s

Now, from the kinematics equations, we know that $x = x_0 + v_0 t + \frac{1}{2} a * t^2$. We'll give our initial distance as 5 AU from the star.

In[•]:= $d_{decel} = \text{Quantity}[5, \text{"AstronomicalUnit"}] + (v_c * t_{decel}) + \frac{1}{2}(-a * t_{decel}^2)$

Out[•]= -4.57468×10^{15} m $+ \left(3.05565 \times 10^7 \text{ s}\right) v_{299\,792\,458 \text{m/s}}$

```
In[•]:= d_startdecel = d - d_decel
Out[•]= 4.47544 × 10^16 m + (-3.05565 × 10^7 s) v_299 792 458m/s
```

This gives us our deceleration point. We should start decelerating

```
In[•]:= UnitConvert[d_startdecel, "LightYears"]
Out[•]= UnitConvert[4.47544 × 10^16 m + (-3.05565 × 10^7 s) v_299 792 458m/s, LightYears]
```

or

```
In[•]:= UnitConvert[d_decel, "LightYears"]
Out[•]= UnitConvert[-4.57468 × 10^15 m + (3.05565 × 10^7 s) v_299 792 458m/s, LightYears]
```

from Proxima Centauri.

Cruising phase

Now that we know how far it takes to accelerate to cruising speed and at what point to decelerate, we know how far we have to cover at cruising speed.

```
In[•]:= d_cruise = d - d_accel - d_decel
Out[•]= 4.01784 × 10^16 m + (-3.05565 × 10^7 s) v_299 792 458m/s
```

```
In[•]:= UnitConvert[d_cruise, "LightYears"]
Out[•]= UnitConvert[4.01784 × 10^16 m + (-3.05565 × 10^7 s) v_299 792 458m/s, LightYears]
```

Knowing our cruising velocity, we can figure out how long the cruising phase will take. From the kinematics equations, we know that $x = x_0 + v * t$. If we assume a zero starting point, then we can compute the cruising time as

```
In[•]:= t_cruise = UnitConvert[d_cruise / v_c, "Years"]
Out[•]= UnitConvert[(4.01784 × 10^16 m + (-3.05565 × 10^7 s) v_299 792 458m/s) / v_299 792 458m/s, Years]
```

Or, to track the number of days:

```
In[•]:= UnitConvert[t_cruise, "Days"]
Out[•]= UnitConvert[UnitConvert[(4.01784 × 10^16 m + (-3.05565 × 10^7 s) v_299 792 458m/s) / v_299 792 458m/s, Years], Days]
```

The total flight time is then

In[•]:= $t_{mission} = t_{accel} + t_{cruise} + t_{decel}$

Out[•]= $6.11137 \times 10^7\ s + \text{UnitConvert}\left[\frac{4.01784 \times 10^{16}\ m + \left(-3.05565 \times 10^7\ s\right) v_{299\,792\,458 m/s}}{v_{299\,792\,458 m/s}}, \text{Years}\right]$

In[•]:= `UnitConvert[t_mission, "Days"]`

Out[•]= $\text{UnitConvert}\left[6.11137 \times 10^7\ s + \text{UnitConvert}\left[\frac{4.01784 \times 10^{16}\ m + \left(-3.05565 \times 10^7\ s\right) v_{299\,792\,458 m/s}}{v_{299\,792\,458 m/s}}, \text{Years}\right], \text{Days}\right]$

In[•]:= `UnitConvert[t_mission, "Years"]`

Out[•]= $\text{UnitConvert}\left[6.11137 \times 10^7\ s + \text{UnitConvert}\left[\frac{4.01784 \times 10^{16}\ m + \left(-3.05565 \times 10^7\ s\right) v_{299\,792\,458 m/s}}{v_{299\,792\,458 m/s}}, \text{Years}\right], \text{Years}\right]$

My goal is to “launch” the mission on my 35th birthday, which means I’d reach Proxima Centauri shortly after turning 40.

Relativistic time effects

During close to light speed travel, time on Earth will pass faster than it will on the ship. The relative time can be calculated using:

In[•]:= $\gamma[v\_] := \sqrt{1.0 - \frac{v^2}{c^2}}$

In[•]:= `Δ_tr[v_, Δt_] := Δt / γ[v]`

An hour and a half of travel at cruising velocity turns into

In[•]:= `Δ_tr[v_c, Quantity[1.5, "Hours"]]`

Out[•]= $\frac{1.5\ h}{\sqrt{1. + \left(-\frac{1}{89\,875\,517\,873\,681\,764}\ s^2/m^2\right) v^2_{299\,792\,458 m/s}}}$

Or, in seconds,

In[•]:= `Δ_tr[v_c, UnitConvert[Quantity[1.5, "Hours"], "Seconds"]]`

Out[•]= $\frac{5400.\ s}{\sqrt{1. + \left(-\frac{1}{89\,875\,517\,873\,681\,764}\ s^2/m^2\right) v^2_{299\,792\,458 m/s}}}$

For a one hour period at a tenth of the speed of light:

```
In[•]:= Δtr[0.1 * c, Quantity[3600, "Seconds"]]
Out[•]= 3618.14 s
```

One thing I wanted to figure out is the local time when the relative drift is 50 years. From the simulation, I know that the drift once the coasting phase starts is 186 days.

```
In[•]:= drift = Quantity[50, "Years"] - Quantity[186, "Days"]
Out[•]= 18 064 days
```

```
In[•]:= γcoast = γ[vcruise]
Out[•]= 0.0447102
```

```
In[•]:= t50 = drift * γcoast + Quantity[186, "Days"]
Out[•]= 993.645 days
```

Flight Update Intervals

How far does the ship travel in a given time period at cruising velocity?

```
In[•]:= UnitConvert[Quantity[1, "Seconds"] * vcruise, "AstronomicalUnit"]
Out[•]= 0.00200198 au
```

```
In[•]:= UnitConvert[Quantity[1, "Minute"] * vcruise, "AstronomicalUnit"]
Out[•]= 0.120119 au
```

```
In[•]:= UnitConvert[Quantity[1, "Hours"] * vcruise, "AstronomicalUnit"]
Out[•]= 7.20715 au
```

A minute between updates is a reasonable interval; this mostly matters for the flight simulator. In the actual flight software, a ticker will update the flight every second.

The Hardware

The simulation will run on a Raspberry Pi with an 18650-powered battery pack and an eInk display. During the acceleration and deceleration phase, it will make sense to update more often, whereas during the cruising phase it may update more slowly.

- Raspberry Pi 3 Model B Rev 1.2
- A 52Pi EP-0136 UPS (with 2x Sony 18650 cells)
- A Pi Supply PaPiRus hat with a 2.7” eInk display

I’ll need to figure out a case probably, but I doubt I can get one made before the launch date.

The Software

The core simulation (the mission itself) is written in Go. I've made the decision to split out the simulation core from the display portion. Source code is on Github.

Figuring out milestones

It would be interesting to figure out when certain milestones will occur. These will generally occur at a set distance; for example, we can assume we pass Mars' orbit when we've gone 0.52 AU.

From the fourth kinematics equation, $x = x_0 + v_0 + t + \frac{a*t^2}{2}$. In this case, x_0 is 0, and if we rearrange the equation, we'll get the equation in quadratic form $a*x^2 + b*x + c = 0$.

$$0 = \frac{a*t^2}{2} + v_0\, t - x$$

If we want to solve this as a quadratic root,

$$x = \frac{-b \pm \sqrt{b^2 - 4\,ac}}{2}$$

we find that $a = \frac{a}{2}$, $b = v_0$, and $c = -x$. Inserting these into the quadratic root equation

$$t = \frac{-v_0 \pm \sqrt{{v_0}^2 - 4*\frac{a}{2}*c}}{2*\frac{a}{2}}$$

This can be simplified:

$$t = \frac{-v_0 \pm \sqrt{{v_0}^2 - 2*a*c}}{a}$$

One of these solutions will be negative; we can remove this solution because it doesn't make sense. Therefore, we can compute the time using

$$t = \max\left(\frac{-v_0 \pm \sqrt{{v_0}^2 - 2*a*c}}{a}\right)$$

By using some common sense, we can see that subtracting the $\sqrt{{v_0}^2 - 2\,ac}$ quantity will probably only

yield a negative value, so we can remove that case.

```
In[ ]:= FlightTime[x_] :=
          UnitConvert[(-vescape + Sqrt[vescape^2 + 2 * a * Quantity[x, "AstronomicalUnit"]]) / a, "Days"]

In[ ]:= FlightTime[0.52]

Out[ ]= 1.44513 days
```

Milestones

```
In[ ]:= LaunchDate = DateObject[]

Out[ ]= Wed 16 Feb 2022 21:18:28 GMT−8

In[ ]:= Milestone[name_, x_] := {name, FlightTime[x], LaunchDate + FlightTime[x]}

In[ ]:= Grid[{Milestone["Mars", 0.52], Milestone["Jupiter", 4.2],
          Milestone["Saturn", 8.58], Milestone["Uranus", 18.2], Milestone["Neptune", 29.05],
          Milestone["Pluto", 38.48], Milestone["Termination shock", 90],
         Milestone["Heliopause", 120], {"Truly alone", t50, LaunchDate + t50}}, Frame → All]
```

Out[ ]=

Mars	1.44513 days	Fri 18 Feb 2022 07:59:27 GMT−8
Jupiter	4.13121 days	Mon 21 Feb 2022 00:27:24 GMT−8
Saturn	5.91033 days	Tue 22 Feb 2022 19:09:20 GMT−8
Uranus	8.61405 days	Fri 25 Feb 2022 12:02:41 GMT−8
Neptune	10.8864 days	Sun 27 Feb 2022 18:34:50 GMT−8
Pluto	12.5313 days	Tue 1 Mar 2022 10:03:33 GMT−8
Termination shock	19.1716 days	Tue 8 Mar 2022 01:25:35 GMT−8
Heliopause	22.1395 days	Fri 11 Mar 2022 00:39:21 GMT−8
Truly alone	993.645 days	Wed 6 Nov 2024 12:46:46 GMT−8