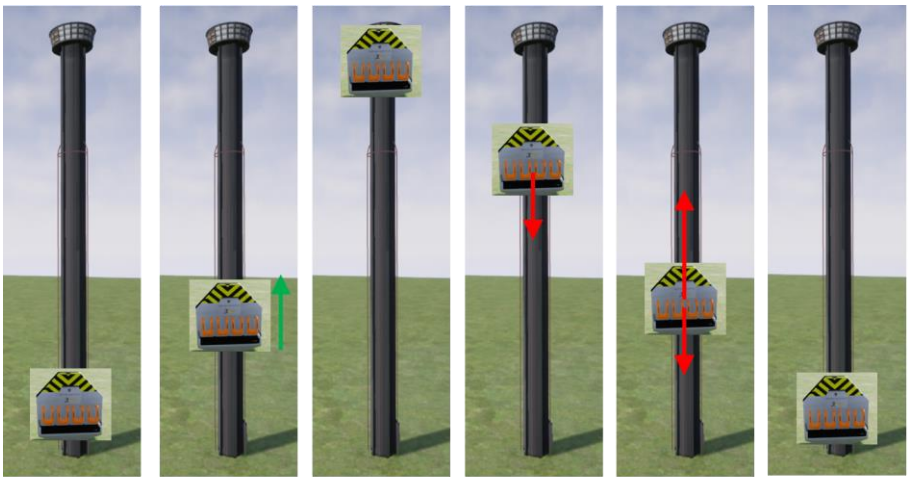
Theory of the Apocalypse

The Apocalypse Finite State Machine

This fairground ride goes through a number of *states*. Each state can specify either the *velocity* of the car or the *force* exerted on the car (by gravity and the brakes). So here's the FSM



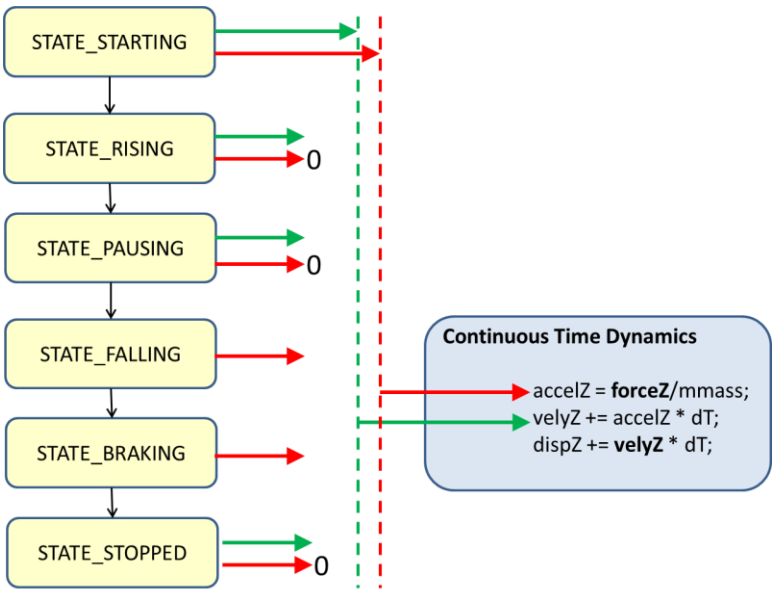
In state STARTING, there is no velocity or force. Transition from this state to the state RISING happens when the simulation is started. In state RISING, the car travels up with a constant velocity. Transition from this state to the state PAUSING occurs when the car has reached the top. In state PAUSING there is no velocity and no force, but the car remains in this state for a certain time. Transition to the state FALLING occurs when this time is complete. In state FALLING there is a force due to gravity on the car (and its velocity is not constrained). Transition to the state BRAKING occurs when the car has reached a certain height. In state BRAKING there are two forces on the car, gravity still exists, so there is a force down, but now there is a braking force upwards. Transition to the state STOPPED occurs when the Car has reached ground level. In this state there are no forces, but the velocity is constrained to be zero.

This description gives us a list of variables and parameters, and suggests some names: *dispZ, velyZ, accelV, mass, gravity, liftVelocity, releaseHeight, brakingHeight, brakingForce, releaseTime.*

How do we combine a finite state machine description with a 'continuous time' (CT) system described by ODEs? Have a look at the diagram below. Think of each state as feeding info into the CT system, providing either a value of force, or a velocity-force pair (where the force part is 0). Only one state is active at any time, so we can think of the state's output(s) as connected to a busbar feeding the CT system. These are shown as vertical dotted lines; forces are red, velocities are blue.

Let's look at the details. In states RISING, PAUSING and STOPPED, the FSM outputs **forceZ = 0** and supplies a **velyZ** appropriate to the state, to the ODE solver. So effectively the velocity is constrained and we are

integrating just one ODE. In the other states, the FSM output values for **forceZ** but no velocity, and the CT system uses this to compute the car's velocity and displacement as usual. There is no constraint on the velocities.



How to code the FSM? You could use a series of if-then-elses, but that immediately shouts out "Use a switch!", something like this

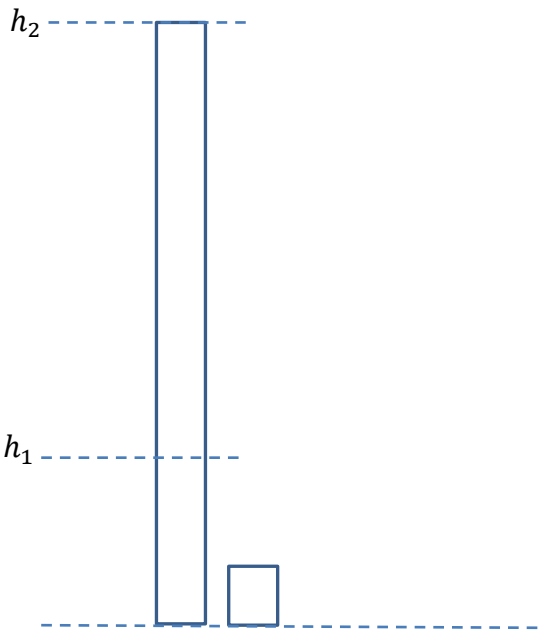
```
switch(state) {
    case RISING:
        set velocity
        set force
        test are we at top?
        if ( yes ) state = PAUSING;
        break;
    case PAUSING:
        test for timeout
        if( yes ) state = FALLING;
        break;
 …
}
```

So you need a variable **int state** to keep track of the state, and some defines like
#define STATE_STARTING 0
Finally, where to place the FSM in the MAS22_Apocalypse template? It should go here, in Computation() before the CT ODE solver is called so that the solver gets the most recent values of force and velocity. You could write the FSM code here directly, or you could place that code in a function and call your function here.

```
void AMAS22_Apocalypse::Computation() { // ------------------------------

      // Other stuff specific to the model may go in here ------------------
```

Now for the Physics



There are two important heights, h_2 is the height at which the car is released (this is 54 metres) and h_1 which is the height at which a braking force *B* is applied. Between these heights the car is in free-fall accelerated by a force due to gravity.

We simply use concepts of work and energy. At the top of the tower when the car is released it is not moving. It has no *kinetic* energy. When it reaches $h_1$it does have some kinetic energy since it now is moving with velocity v_1. This kinetic energy has come from the force on the car, acting over the distance of fall from $h_2\ to\ h_1$. Energy is calculated as the force times the distance the force has been applied. Here the force on the car is just *mg* and the distance moved (dropped) is $(h_2 - h_1)$ so the energy given to the car is just

$$mg(h_2 - h_1)$$

So the car has this kinetic energy at height h_1. Below this height the braking force *B* is applied which removes all of this kinetic energy by the time the car reaches ground level. Again we use the same idea that force times distance gives the energy removed. Here the force is (B – *mg*) upwards, and the distance moved is h_1. So the energy removed is just

$$(B - mg)h_1$$

Almost there. We started with no kinetic energy and we ended up with no kinetic energy, so the energy given to the car and the energy taken away must be the same. So the two expressions above are the same, so we can write

$$(B - mg)h_1 = mg(h_2 - h_1)$$

and this simplifies to $$Bh_1 = mgh_2$$

Finally, let's calculate the *deceleration* experienced by the riders. Deceleration is negative acceleration, so in this case points upwards. In general acceleration is calculated as force divided by the mass of the thing accelerating. So we have

$$a = \frac{F}{m}$$

In this case the force upwards is (B – mg) so the acceleration becomes

$$\begin{aligned} a &= \frac{(B - mg)}{m} \\ &= \frac{\left(mg\frac{h_2}{h_1} - mg\right)}{m} \\ &= g\left(\frac{h_2}{h_1} - 1\right) \end{aligned}$$

and we divide by *g* to express this as a number of *g*'s

$$a(in\ g's) = \frac{h_2}{h_1} - 1$$

and that, guys, is how we do that.