Scenario 1-1: (Car 1, Terrain 1) Car1.jpg Scene1.jpg

In this scenario, the car behaved exactly as expected. On the first run, the car drove off of the ramp at such a high speed that it hit the midair obstacle and then “fell to its doom”. In the second run, the car applied reverse torque at the Control Step that it failed in the first run. However, this torque didn’t take effect until the car was already in the air after hitting the jump and thus had very little effect on the car’s outcome. In the third run, the change to the Control Plan made a significant difference. At the top of the ramp, the car slowed down enough to pass under the obstacle but still maintained enough speed to get over the gap and ultimately succeed in reaching the goal.

Scenario 1-2: (Car 2, Terrain 1) Car2.jpg Scene1.jpg

This scenario succeeds in the first run. The car hits the obstacle like in Scenario 1, but due to the configuration of the wheels, the robot “slides” underneath the obstacle rather than hitting it flat and falling straight down.

Scenario 1-3: (Car 3, Terrain 1) Car3.jpg Scene1.jpg

This simulation proved to be slightly less efficient at finding a solution. It started out similarly to Scenario 1-1 in that it went too fast and hit the floating obstacle. This happened in the first five runs. In the sixth run, however, the car over compensated and went too slowly to make it across the gap. In the runs that proceeded, the car ended up teetering at the edge of the ramp before falling. In each run, the car tended to teeter longer than it did in the previous run before falling, thus propagating the failure point further into the Control Plan. This behavior of teetering longer in successive runs continued for many runs and thus provided us with a great example of “Error Propagation” as described previously in this report. Eventually, in Run 75, the car teetered in such a way that it pulled itself backwards partly down the ramp and then was able to gain enough forward momentum to jump off the end of the ramp and (just barely) stumble across the gap.

Scenario 2-4 (Car 1, Terrain 2) Car1.jpg Scene2.jpg

In this simulation, the car ends up driving onto the plateau and then nose-diving into the incline after the plateau. After landing on its “nose” the car tips over onto its back and gets stuck. This occurs in the first four runs with nearly identical results. On the fifth run, however, instead of tipping onto its back, it tips the other way, gets back onto its wheels, and continues to the goal after undergoing a flip. This scenario demonstrates a case where the forces generated by freely spinning wheels observably affect the path of the car and (in this case) leads to a successful goal acquisition.

Scenario 2-5 (Car 2, Terrain 2) Car2.jpg Scene2.jpg

As one would expect, this car excelled at completing this course, doing so in just a single run. The car scaled the first incline, drove across the plateau, came off the edge and landed “upside-down”. It continued in this inverted state and launched off of the second incline, did two full flips, and then landed and proceeded to the goal. Being able drive while upside-down makes this car virtually invincible on any terrain that doesn’t have any “bottomless pits”.

Scenario 2-6 (Car 3, Terrain 2) Car3.jpg Scene2.jpg

This scenario gets solved pretty quickly, taking only three Runs. However, this scenario ends up highlighting one of the biggest issues we had with this project. As the output file below depicts, the simulation recognizes that the car gets stuck in Runs 1 and 2, but an observer of the actual simulation would say that the car was not stuck at all. In this simulation, the car easily gets across the plateau, does a flip and then lands on its “nose”. Unfortunately, before being able to turn over back onto its “wheels” (or possibly onto its back) the simulation falsely recognizes that the car is stuck. This problem was addressed in more detail previously in this report. In the third run of this scenario, the Control Plan has changed enough to avoid the false stuck recognition and the car makes it to the goal after a few flips.

Scenario 3-7 (Car 1, Terrain 3) Car1.jpg Scene3.jpg

This simulation is completed in one Run, but it’s a very long Run. This is an example of a scenario in which an exhaustive search would be able to find a much better Run time than the one achieved by the first solution found. In this simulation, the car climbs over the first boulder with the help of upward momentum achieved from launching off of the first incline. It then does a flip and goes into a “wheelie” as it approaches the next boulder. By being in a wheelie when it collides with the second boulder, it easily rolls over it. It then partially flips and ends up landing “upright” (with the back wheel touching the ground and its front wheel in the air). It then struggles to get up the next incline in this position for about ten simulation seconds before finally flipping back onto both of its wheels. Finally, it makes it up the final incline, rams into the third boulder, climbs over it, does one more flip, and proceeds to the goal.

Scenario 3-8 (Car 2, Terrain 3) Car2.jpg Scene3.jpg

Once again, this car traverses the terrain in the first run. One by one, the car plowed into the boulders and climbed over them with relative ease. The car did a half-flip after climbing each boulder and thus finished the finished the course “upside-down”.

Scenario 3-9: Car 3, Terrain 3 Car3.jpg Scene3.jpg

This simulation did not find a solution within 200 Runs, at which point we decided to terminate the simulation. Shown below are the first 100 Runs. At the rate the simulation was going, it would be many hours before any solution was found. In this scenario, at the beginning of each run, the car would jump over the first boulder and then go into somersaults. It would continue somersaulting over the second boulder and then land flat on its back on the incline to the left of the third boulder. Once the stuck condition was recognized, the next run would commence with a new control plan (derived from reverse binary counting as previously described in this report). In this simulation, due to the orientation of the car and aspects of dVC, it appeared that applying reverse torque would cause the car to “jitter” while on its back. As mentioned previously in this report, this jittering can keep the simulation from properly recognizing that the car is stuck. Since the reverse binary counting tends to add Reverse Torque commands to each Control Plan, the car in this simulation would tend to progressively jitter more in each sequential run. This increased jittering in turn caused the stuck recognition to be delayed further in subsequent runs. Thus, this is another example of “Error Propagation” as was described in this report. Even without this Error Propagation we predict that finding a solution to this scenario would still take a very long time. This is because the car starts tumbling early on in the simulation before ultimately getting stuck. While tumbling, modifications to the torques on the “wheels” would presumably have minimal effect. We predict that the simulation would have to modify the control plan previous to when the car starts tumbling to have a significant affect and thus finally lead to a successful control scheme.