



**Prize Winner**

# **Programming, Apps & Robotics Year 7-8**

**Thomas Palmer**

**Pedare Christian College**



# SpaceX Starship Simulator:

By Thomas Palmer

## Scientific Purpose:

Hundreds of objects are sent into space, and millions of dollars are wasted each year through failures (Aerospace CSIS 2020). *StarshipSim* can be used by potential customers of SpaceX to accurately simulate whether their payloads can withstand the immense stresses an object is subjected to when travelling into space without leaving the ground. Before manufacturing, an individual part, system or the entire vehicle can be simulated 100's of times with no extra cost. Satellites, Space Stations and Spacecraft can all be tested.

*StarshipSim* is designed for the SpaceX Starship launch vehicle only, which is currently under development by SpaceX, with an orbital test expected in the coming weeks. This program will reduce the likelihood of catastrophic incidents, and could save billions of dollars in damages.

*StarshipSim* demonstrates the fundamental laws of physics, thermodynamics, fluid dynamics and drag to achieve high accuracy in the simulation of travelling through the atmosphere, getting into orbit and speeding towards other celestial bodies.

## Using the Program:

After loading, a few questions will appear on the screen. After answering, the vehicle will be ready to launch. It will automatically set itself on a trajectory to achieve a stable orbit, and from there another program is used to simulate interplanetary flight. A detailed method is available under the *Methods* section.

## Loading the Program:

The program runs in Python 3.7 using the PyPi modules: Pygame, Thorpy, OS, and EasyGui.

To open the demonstration video, access Google Drive, open *StarshipSim* and either use the built-in or third-party video player available on the device.

## Entering Payload Specifications:

The first question is 'What is the Payload Weight in kilograms?'. This sets the weight of the product that the consumer wishes to travel into space. The next question is 'What is the Payload G-Force Rating?'. This sets the gravitational force stresses the payload can withstand before critical failure. The final question is 'What is the temperature rating in celsius?'. This sets the temperature that the payload can withstand before critical failure. Submit answers by clicking Enter. After completing these questions, the vehicle is ready to launch.

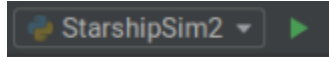
## Controls:

By pressing the UP arrow key, if all prior steps were completed properly, then the vehicle will begin to launch. The movement, time and stress statistics are shown in the bottom left-hand corner of the screen. The vehicle will set itself on a suitable trajectory to achieve orbit. From there, transferring to the program *interplanetary.py* will enable the vehicle to travel to either the Moon or Mars.

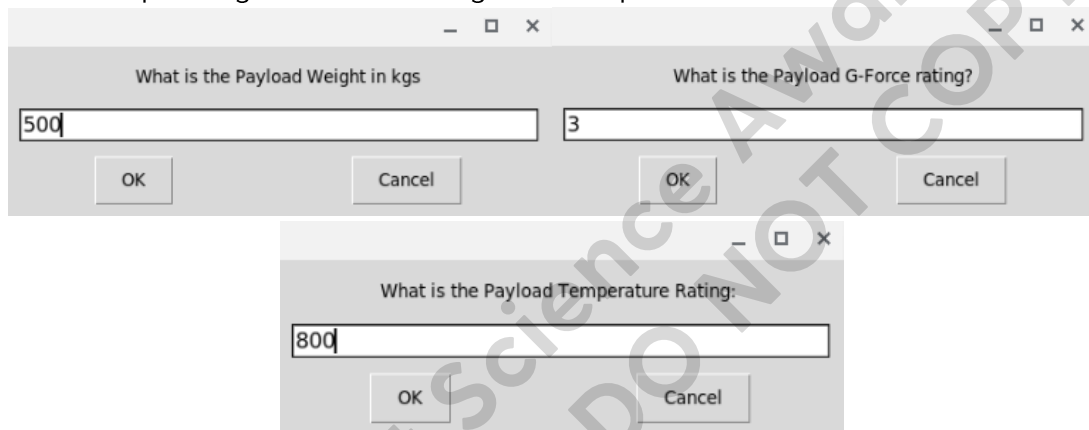
If the payload ratings were exceeded, then the mission will end in a failure and the rocket will explode. This shows that the payload is not robust enough to withstand the stresses of the mission.

**Method:**

1. Load StarshipSim2.py using instructions under “**Loading the Program**”.



2. Answer the three questions ‘What is the Payload Weight in kilograms?’, ‘What is the Payload G-Force Rating’ and ‘What is the temperature rating in celsius?’ by entering a value and pressing **Enter** or selecting OK. Examples are shown below.



The image shows three separate dialog boxes for data entry. The first dialog box asks 'What is the Payload Weight in kgs' with a text input field containing '500' and 'OK'/'Cancel' buttons. The second dialog box asks 'What is the Payload G-Force rating?' with a text input field containing '3' and 'OK'/'Cancel' buttons. The third dialog box asks 'What is the Payload Temperature Rating:' with a text input field containing '800' and 'OK'/'Cancel' buttons.

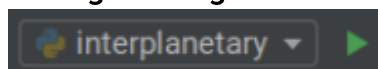
3. The flight data will need to be recorded manually by the user. An automated system is currently under development.
4. To ignite the first stage of the vehicle click the **UP** arrow key
5. Notice in the **left-hand corner** of the screen that vehicle data is displayed in real-time.



6. The vehicle will steer itself on a predetermined course utilising data from the SpaceX Falcon Heavy vehicle launches.



7. To continue on for interplanetary travel, load interplanetary.py using instructions under “Loading the Program”.



8. Answer the questions displayed on the screen, and select your destination.

What is the Payload Weight in kgs

500

OK Cancel

What is the Payload G-Force rating?

3

OK Cancel

What is the destination of this journey?

Mars

Moon

Cancel OK

9. Press the **UP** arrow key to ignite the interplanetary stage.
10. The layout of the screen is nearly identical to the orbital program.



11. Once finished on either program, click **esc** to exit.

### The Program:

The programs were written in Python 3.7 and a copy of the code is displayed below.

#### StarshipSim2.py

```
import pygame, thorpy, os, playsound, easygui
os.environ['SDL_AUDIODRIVER'] = 'dsp'

pygame.init()

# Initial Parameters and Variables / Constants
fuel = 3725 #tonnes
payload = easygui.enterbox("What is the Payload Weight in kgs", "Payload
Weight")
payload2 = easygui.enterbox("What is the Payload G-Force rating?", "Payload
Weight")
payloadw = float(payload)
fuelburnt = 0.112 * (payloadw/2500) #1/10th Second
fuelburnl = 1.12 #1/10th Second
screenHeight = 1000
screenWidth = 1000
screen = pygame.display.set_mode((screenWidth, screenHeight))
pygame.display.set_caption("Spacex Starship Simulator")

# All the characters at different angles
starship = pygame.image.load("/mnt/chromeos/MyFiles/Downloads/emship.png")
# Normal Character
background =
pygame.image.load("/mnt/chromeos/MyFiles/Downloads/embackground.jpg") # For
background
```

```

bellyflops =
pygame.image.load("/mnt/chromeos/MyFiles/Downloads/emship90.png") # For
descent
background2 =
pygame.image.load("/mnt/chromeos/MyFiles/Downloads/interplanetary.jpg") #
For background
explosion =
pygame.image.load("/mnt/chromeos/MyFiles/Downloads/emexplosion.png")
#Change to png / For explosion

# Miscellaneous Constants
yvel0 = 0
xvel = 0
width = 64 # In Pixels
height = 64 # In Pixels

# Time Constants and Variables
time_delta = 0.1 # In seconds
time = pygame.time.get_ticks()
t = 0
counter = 0

# Coordinate Constants for Starship
xx = 0 # Background set coordinate
yy = 0 # Background set coordinate
x0 = 900 # Starship set coordinate
y0 = 800 # Starship set coordinate

# Acceleration Constants
accscale = 100
dacceleration = 0.03535
lacceleration = 0.0264
vacceleration = -0.0023 # Metres per second 2
xacceleration = -0.000023

# Acceleration Equivalents
dacc = deceleration
vacc = vacceleration
lacc = lacceleration
xacc = xacceleration
x = x0
yvel = yvel0
y = y0
running = True

# The program loop
while running:
    pygame.time.delay(100)

    # Setting up the screen
    screen.fill((0, 0, 0), (200, 0, 770, 770))
    screen.blit(background, (xx, yy))

```

```

# Enabling key recognition
pygame.event.get()
keys = pygame.key.get_pressed()

# Landing sequence for the vehicle
if keys[pygame.K_1] and 800 > y > 793:
    screen.fill((0, 0, 0), (0, 0, 1000, 1000))
    screen.blit(background, (xx, yy))
    screen.blit(starship, (x, y))
    currveld = dacc
    currveld += (-currvel/1000)
    yold = y
    # The equation for vertical position
    y = yold + (yvel * t) + 0.5 * currveld * (t * t)
    currvel = (y - yold) / time_delta
    if t < 1:
        t = 1
    print(currvel, yveld, y)
    currvel = (y - y0) / t
    if y >= 800.1:
        y = 800.12

# To exit the program
elif keys[pygame.K_ESCAPE]:
    running = False

# To achieve orbit
elif keys[pygame.K_UP] and 805 > y > 0:
    takeoff = True
    while takeoff:
        pygame.event.get()
        keys = pygame.key.get_pressed()
        while fuel < 200:
            currvel = 0
            pygame.time.delay(3000)
            if currvel >= 0:
                fuel = 201
                takeoff = False
        if keys[pygame.K_ESCAPE]:
            takeoff = False
            running = False
        if y < 0:
            y = 0
            takeoff = False

    xold = x
    yold = y
    y = yold + (yvel * t) + 0.5 * vacc * (t * t)
    x = xold + (xvel * t) + 0.5 * xacc * (t * t)
    screen.fill((0, 0, 0), (200, 0, 770, 770))
    screen.blit(background, (xx, yy))

    currvel = (y - yold) / time_delta

```

```

t += time_delta
currvel = (y - y0) / t
print("Average velocity is: ", currvel, " at time ", t, " and
position y ", y, "and position x", x)
if currvel <= -20:
    currvel = -20
if y >= 800:
    y = 800
fuel -= fuelburnt
if fuel < 200 and currvel >= 0:
    fuel = 201
    takeoff = False

yvel = round(currvel * -100, 1)

# Setting up the Statistics display
yvel = round(currvel * -100, 1)
fuel = round((fuel/3725)*100, 1)
fueldisplay = str(fuel)
rapid = thorpy.Element("RapidRockets")
yveldisplay = str(yvel)
velocity = thorpy.Element(yveldisplay)
text = thorpy.Element("Velocity = ")
elev = round((800 - y) * 150)
if elev > 119000:
    pygame.time.delay(1000)
    currvel = 0.0001
    takeoff = False

eveldisplay = str(elev)
elevation = thorpy.Element(eveldisplay)
text2 = thorpy.Element("Elevation = ")
text3 = thorpy.Element("Time = ")
text4 = thorpy.Element("Fuel% = ")
text5 = thorpy.Element("G-Force =")
tround = round(t*2)
if tround == 0:
    tround = 1
gforce = round((yvel/tround) / 9.8 + 1, 2)
gforcedisplay = str(gforce)
time22 = str(tround)
time2 = thorpy.Element(time22)
rocketfuel = thorpy.Element(fueldisplay)
rocketg = thorpy.Element(gforcedisplay)
box = thorpy.Box([rapid, text4, rocketfuel, text5, rocketg,
text3, time2, text, velocity, text2, elevation])
velocity.set_main_color((105, 105, 105))
text.set_main_color((105, 105, 105))
text2.set_main_color((105, 105, 105))
text3.set_main_color((105, 105, 105))
text4.set_main_color((105, 105, 105))
text5.set_main_color((105, 105, 105))
rocketg.set_main_color((105, 105, 105))

```



```

rocketfuel.set_main_color((105, 105, 105))
time2.set_main_color((105, 105, 105))
elevation.set_main_color((105, 105, 105))
velocity.set_font_color((255, 255, 255))
text.set_font_color((255, 255, 255))
text3.set_font_color((255, 255, 255))
text4.set_font_color((255, 255, 255))
text5.set_font_color((255, 255, 255))
rocketg.set_font_color((255, 255, 255))
rocketfuel.set_font_color((255, 255, 255))
time2.set_font_color((255, 255, 255))
text2.set_font_color((255, 255, 255))
elevation.set_font_color((255, 255, 255))
box.set_main_color((0, 0, 0))
box.set_center_pos([50, 700])
box.set_size([90, 400])
box.set_font("century")
box.update()

screen.blit(starship, (x, y))

box.blit()

# Setting up the trajectory
if elev >= 1000:
    starship =
pygame.image.load("/mnt/chromeos/MyFiles/Downloads/emship-5.png")
    xacc -= 0.0000005

    if elev >= 3000:
        starship =
pygame.image.load("/mnt/chromeos/MyFiles/Downloads/emship-10.png")
        xacc -= 0.0000000005
    if elev >= 5000:
        starship =
pygame.image.load("/mnt/chromeos/MyFiles/Downloads/emship-20.png")
        xacc -= 0.000000005
    if elev >= 10000:
        starship =
pygame.image.load("/mnt/chromeos/MyFiles/Downloads/emship-35.png")
        xacc -= 0.000000125
    if elev >= 20000:
        starship =
pygame.image.load("/mnt/chromeos/MyFiles/Downloads/emship-45.png")
        xacc -= 0.000000015
    if elev >= 79000:
        starship =
pygame.image.load("/mnt/chromeos/MyFiles/Downloads/emship-70.png")
        xacc -= 0.000000175

    if elev >= 118000:

```

```

        starship =
pygame.image.load("/mnt/chromeos/MyFiles/Downloads/emship.png") # Normal
Character

        xacc = 0

        if x < 1:
            x = 1
        pygame.display.update()

# Simulating orbit and re-entry
elif 0 < y <= 799 and not keys[pygame.K_l]:
    screen.blit(bellyflops, (x, y))
    if elev > 100000:
        currveld = 0.000001
    else:
        currveld = -currvel/1000
    yold = y
    y = yold + (yvel * t) + 0.5 * currveld * (t * t)
    currvel = (y - yold) / time_delta
    if elev < 80000:
        t += 0.05
    if t < 1:
        t = 1
    print(currvel, yveld, y)
    currvel = (y - y0) / t
    if currvel > -1.91:
        currvel = -1.917
    print(currvel, yveld, y)

    yveld = round(-currvel * 100)
    rapid = thorpy.Element("RapidRockets")
    yveldisplay = str(yveld)
    velocity = thorpy.Element(yveldisplay)
    text = thorpy.Element("Velocity = ")
    elev = round((800 - y) * 150)
    eveldisplay = str(elev)
    elevation = thorpy.Element(eveldisplay)
    text2 = thorpy.Element("Elevation = ")
    box = thorpy.Box([rapid, text, velocity, text2, elevation])
    velocity.set_main_color((105, 105, 105))
    text.set_main_color((105, 105, 105))
    text2.set_main_color((105, 105, 105))
    elevation.set_main_color((105, 105, 105))
    velocity.set_font_color((255, 255, 255))
    text.set_font_color((255, 255, 255))
    text2.set_font_color((255, 255, 255))
    elevation.set_font_color((255, 255, 255))
    box.set_main_color((0, 0, 0))
    box.set_center_pos([50, 840])
    box.set_size([90, 220])
    box.set_font("century")

    box.update()

```

```

        box.blit()

        if (currvel < -0.02 and y > 800.1) or fuel == 0:
            screen.fill((0, 0, 0), (200, 0, 800, 869))
            screen.blit(background, (xx, yy))
            screen.blit(explosion, (x, y))
            pygame.time.delay(3000)
            running = False

    elif y < 1:
        y = 1
        screen.blit(starship, (x, y))

    elif y >= 795:
        y = 800
        screen.blit(starship, (x, y))
        yvel = yvel0

    pygame.display.update()
pygame.quit()

```

### **interplanetary.py**

```

import pygame, thorpy, os, playsound, easygui

os.environ['SDL_AUDIODRIVER'] = 'dsp'

pygame.init()

# Initial Parameters and Variables / Constants
fuel = 3725 #tonnes
payload = easygui.enterbox("What is the Payload Weight in kgs?", "Payload Weight")
payload2 = easygui.enterbox("What is the Payload G-Force rating?", "Payload Weight")
destination = easygui.choicebox("What is the destination of this journey?", "Destination", ["Mars", "Moon"])
payloadw = float(payload)
screenHeight = 1000
screenWidth = 1000
screen = pygame.display.set_mode((screenWidth, screenHeight))
pygame.display.set_caption("Spacex Starship Simulator")

# All the characters at different angles
starship = pygame.image.load("/mnt/chromeos/MyFiles/Downloads/emship.png")
# Normal Character
background =
pygame.image.load("/mnt/chromeos/MyFiles/Downloads/embackground.jpg") # For background
bellyflops =
pygame.image.load("/mnt/chromeos/MyFiles/Downloads/emship90.png") # For descent

```

```

mobbackground =
pygame.image.load("/mnt/chromeos/MyFiles/Downloads/earthmoon.jpg") # For
background
mabackground =
pygame.image.load("/mnt/chromeos/MyFiles/Downloads/earthmars.png") # For
background
explosion =
pygame.image.load("/mnt/chromeos/MyFiles/Downloads/emexplosion.png")
#Change to png / For explosion

# Miscellaneous Constants
yvel0 = 0
xvel = 0
width = 64 # In Pixels
height = 64 # In Pixels

# Time Constants and Variables
time_delta = 0.1 # In seconds
time = pygame.time.get_ticks()
t = 0
counter = 0

# Coordinate Constants for Starship
xx = 0 # Background set coordinate
yy = 0 # Background set coordinate
x0 = 120 # Starship set coordinate
y0 = 800 # Starship set coordinate

# Acceleration Constants
accscale = 100
dacceleration = 0.03535
lacceleration = 0.0264
vacceleration = -0.01 # Metres per second 2
xacceleration = -0.00000655

# Acceleration Equivalents
dacc = deceleration
vacc = vacceleration
lacc = lacceleration
xacc = xacceleration
x = x0
yvel = yvel0
y = y0

fuelburnt = 0.112 * (payloadw/2500) # 1/10th Second
screenHeight = 1000
screenWidth = 1000
screen2 = pygame.display.set_mode((screenWidth, screenHeight))
pygame.display.set_caption("SpaceX Starship Simulator Interplanetary")
running2 = True

# Setting up the running loop
while running2:

```

```

pygame.time.delay(100)

pygame.event.get()
keys = pygame.key.get_pressed()

if keys[pygame.K_UP] and 805 > y > 0:
    takeoff = True
    while takeoff:
        pygame.event.get()
        keys = pygame.key.get_pressed()
        while fuel < 200:
            currvel = 0
            pygame.time.delay(3000)
            if currvel >= 0:
                fuel = 201
                takeoff = False
        if keys[pygame.K_ESCAPE]:
            takeoff = False
            running = False
        if y < 0:
            y = 0
            takeoff = False

        xold = x
        yold = y
        y = yold + (yvel * t) + 0.5 * yacc * (t * t)
        x = xold + (xvel * t) + 0.5 * xacc * (t * t)

        currvel = (y - yold) / time_delta
        t += time_delta
        currvel = (y - y0) / t

        screen.fill((0, 0, 0), (100, 0, 1000, 1000))
        if destination == "Moon":
            screen.blit(mobackground, (xx, yy))
            tround = round(t * 9910)

        elif destination == "Mars":
            screen.blit(mabackground, (xx, yy))
            tround = round(t * 35480)

        currvel -= 21
        print("Average velocity is: ", currvel, " at time ", t, " and
position y ", y,
            "and position x", x)
        if y >= 800:
            y = 800
        fuel -= fuelburnt
        if fuel < 200 and currvel >= 0:
            fuel = 201
            takeoff = False

        currvel = -0.001

```

```

if currvel <= -90:
    currvel = -90

# Setting up the Statistics display
yveld = round(currvel * -1000, 1)
fuel = round((fuel / 3725) * 100, 1)
fueldisplay = str(fuel)
rapid = thorpy.Element("RapidRockets")
yveldisplay = str(yveld)
velocity = thorpy.Element(yveldisplay)
text = thorpy.Element("Velocity = ")
if destination == "Moon":
    elev = round((((800 - y) * 720000) + 119000) / 1000)
    if elev > 310000:
        takeoff = False
        running2 = False
elif destination == "Mars":
    elev = round((((800 - y) * 653040000) + 119000) / 1000)
    if elev > 325000000:
        takeoff = False
        running2 = False
elevdisplay = str(elev)
elevation = thorpy.Element(elevdisplay)
text2 = thorpy.Element("Elevation = ")
text3 = thorpy.Element("Time = ")
text4 = thorpy.Element("Fuel% = ")
text5 = thorpy.Element("G-Force = ")
if tround == 0:
    tround = 1
gforce = round((yveld / tround) / 9.8 + 1, 2)
gforcedisplay = str(gforce)
time22 = str(tround)
time2 = thorpy.Element(time22)
rocketfuel = thorpy.Element(fueldisplay)
rocketg = thorpy.Element(gforcedisplay)
box = thorpy.Box([rapid, text4, rocketfuel, text5, rocketg,
text3, time2, text, velocity, text2, elevation])
velocity.set_main_color((105, 105, 105))
text.set_main_color((105, 105, 105))
text2.set_main_color((105, 105, 105))
text3.set_main_color((105, 105, 105))
text4.set_main_color((105, 105, 105))
text5.set_main_color((105, 105, 105))
rocketg.set_main_color((105, 105, 105))
rocketfuel.set_main_color((105, 105, 105))
time2.set_main_color((105, 105, 105))
elevation.set_main_color((105, 105, 105))
velocity.set_font_color((255, 255, 255))
text.set_font_color((255, 255, 255))
text3.set_font_color((255, 255, 255))
text4.set_font_color((255, 255, 255))
text5.set_font_color((255, 255, 255))

```

```

rocketg.set_font_color((255, 255, 255))
rocketfuel.set_font_color((255, 255, 255))
time2.set_font_color((255, 255, 255))
text2.set_font_color((255, 255, 255))
elevation.set_font_color((255, 255, 255))
box.set_main_color((0, 0, 0))
box.set_center_pos([50, 700])
box.set_size([90, 400])
box.set_font("century")
box.update()

screen.blit(starship, (x, y))

box.blit()

pygame.display.update()

```

### Acknowledgment:

James Palmer clarified a few elements within the program, but the entire program was written by Thomas Palmer.

### Video Simulation:

The video enters payload characteristics similar to a small space station or a space telescope. If the G-Force or Temperature ratings were to be exceeded by simulated figures, then the mission would be unsuccessful, characterised by an on-screen explosion.

The video first showed a payload weight of 50 tonnes, a G-Force rating of 3G, and a temperature rating of 2000 degrees celsius. Then the **UP** arrow key was pressed for ignition. After attaining orbit, the program was shut off by pressing the **esc** key and interplanetary.py was loaded. From there similar characteristics were entered along with the destination, and a time warped simulation was shown. The video concluded by shutting off interplanetary.py by pressing the **esc** key.

### Bibliography:

SpaceX 2021, *Starship SN10 High Altitude Test Flight*, 17 March, viewed 20 July, <<https://www.youtube.com/watch?v=gA6ppby3JC8>>.

SpaceX 2021, *Starship SN15 High Altitude Test Flight*, 6 May, viewed 6 May, <<https://www.youtube.com/watch?v=z9eoubnO-pE>>.

SpaceX 2021, *Starship SN15 Flight Test Recap*, 14 May, viewed 21 July, <<https://www.youtube.com/watch?v=7CZTLogln34>>.

Felix Zemdegs, F 2021, *What about it!?*, 14 May, viewed 21 July, <<https://www.youtube.com/user/ZSSolomon>>.

C-bass Productions 2021, *Starship and Super Heavy Orbital Test Animation*, 5 June, viewed 21 July, <[https://www.youtube.com/watch?v=iFt\\_LsFRFEQ&t=122s](https://www.youtube.com/watch?v=iFt_LsFRFEQ&t=122s)>.

Aerospace CSIS 2020, *Space Environment: Total Payloads Launched by Country*, viewed 22 July 2021, <<https://aerospace.csis.org/data/space-environment-total-launches-country/>>.

2021 Oliphant Science Award  
Student Work - DO NOT COPY