Welcome to our presentation

សកលវិទ្យាល័យភូមិន្ទភ្នំពេញ

Royal university of Phnom Penh

Foundation Year , Semester2

Course : Technical Drawing
Lecturer : Toem Theara

Topic : Solar System With Python Turtle

Member

ស សុវណ្ណាយ

សុខ អេងឆាយ

សៀ មុនីឧត្តម

Content

Introduction

Earth

- ភពផែនដីជាភពទី៣នៃព្រះអាទិត្យ
- ចម្ងាយពីផែនដីទៅព្រះអាទិត្យស្មើប្រហែល **149.6 លានគីឡូម៉ែត្រ**
- ផែនដីបង្វិលជុំវិញព្រះអាទិត្យ (revolution around the Sun) ប្រហែល 365.25 ថ្ងៃ
- ផែនដីបង្វិល ជុំខ្លួនវា (rotation on its axis) ប្រហែល 24 ម៉ោង

Moon

- ទំហំ: ប្រហែល 3,474 គីឡូម៉ែត្រ (diameter)
- ចម្ងាយពីផែនដី: ប្រហែល 384,400 គីឡូម៉ែត្រ
- រយៈពេលបង្វិលជុំផែនដី: ប្រហែល 27.3 ថ្ងៃ
- រយៈពេលបង្វិលជុំខ្លួន: ផ្នែកមុខនៃ Moon តែងតែបង្ហាញទៅផែនដី, ហើយវាបង្វិលជុំខ្លួនប្រហែល 27.3 ថ្ងៃ (synchronous rotation)

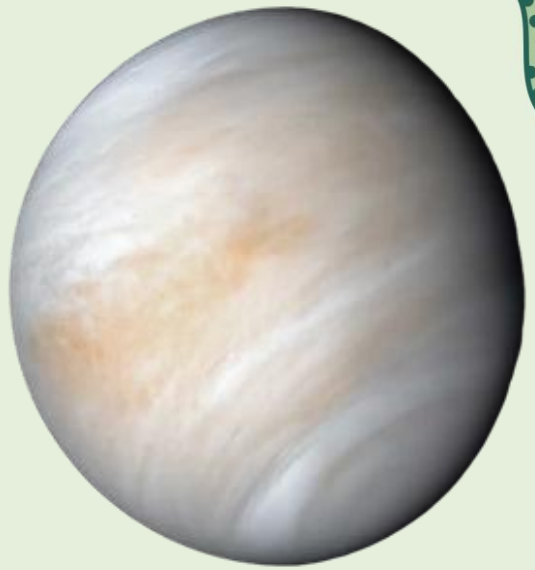
Venus

- Venus (វីនុស) – ភពទី 2 ពីព្រះអាទិត្យ
- ឈ្មោះខ្មែរ: វីនុស
- ទំហំ: អាគុយ ~12,104 គីឡូម៉ែត្រ
- ចម្ងាយពីព្រះអាទិត្យ: ~108.2 លានគីឡូម៉ែត្រ
- បង្វិលជុំខ្លួន: ~243 ថ្ងៃផែនដី (ពេលបង្វិលវាថយក្រោយ)
- បង្វិលជុំព្រះអាទិត្យ: ~225 ថ្ងៃផែនដី (មួយឆ្នាំវីនុស)

លក្ខណៈពិសេស:

មានពពកក្រាស់ និងសីតុណ្ហភាពខ្ពស់ (~465°C)

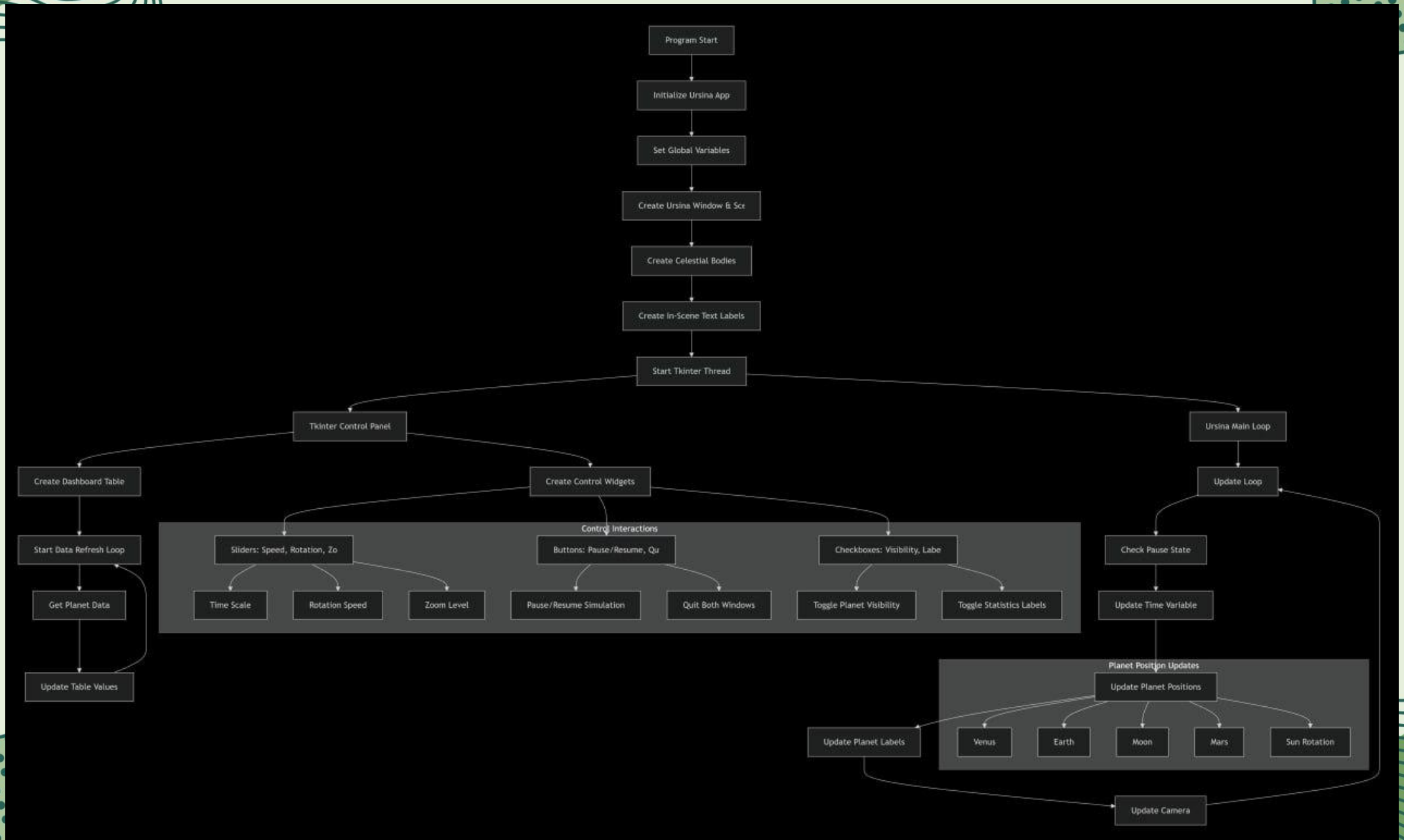
បង្វិលថយក្រោយ (retrograde rotation), ថ្ងៃនៅវីនុសវែងជាងឆ្នាំ



Mars

- Mars (ម៉ាស) – ភពទី 4 ពីព្រះអាទិត្យ
- ទំហំ: ~6,779 គីឡូម៉ែត្រ
- ចម្ងាយពីព្រះអាទិត្យ: ~227.9 លានគីឡូម៉ែត្រ
- បង្វិលជុំខ្លួន: ~24.6 ម៉ោង (ប្រហែល 1 ថ្ងៃផែនដី)
- បង្វិលជុំព្រះអាទិត្យ: ~687 ថ្ងៃផែនដី (មួយឆ្នាំម៉ាស)

Draw Flow Chart



Soft Code

Title

```python
from ursina import *
import math
import threading
import tkinter as tk
from tkinter import ttk
import random

# ----------------------------------
# 1. Global Shared Control Variables
# ----------------------------------
app = Ursina()
window.icon = 'my_icon.ico'
window.size = (864, 1536)
window.position = (192, 108)
window.fullscreen = False

time_scale = 1.0
is_paused = False
show_moon = True
rotation_speed_mult = 1.0
show_earth = True
show_mars = True
show_venus = True
zoom_level = 20
t = -math.pi
show_stats = True

ORBIT_SPEED_MULTIPLIER = 0.5

# ----------------------
# 2. Ursina Scene Setup
# ----------------------
window.color = color.black
window.title = "Ursina Solar System"
window.borderless = False
EditorCamera()
camera.z = zoom_level

# --- Background Starfield ---
for _ in range(300):
    Entity(model='sphere', color=color.white, scale=0.1,
           position=(random.uniform(-120, 120),
                     random.uniform(-120, 120),
                     random.uniform(-120, 120)),
           unlit=True)

```

Title

```python
# --- Celestial bodies ---
sun = Entity(model='sphere', texture='textures/sun.jpg', scale=2, position=(0, 0, 0))
venus = Entity(model='sphere', texture='textures/2K_venus_surface.jpg', scale=1)
earth = Entity(model='sphere', texture="textures/2k_earth_daymap.jpg", scale=1)
mars = Entity(model='sphere', texture='textures/2k_mars.jpg', scale=1)
moon = Entity(model='sphere', texture='textures/2k_moon.jpg', scale=0.4)

# --- Labels (for in-scene display) ---
# D: Distance from Sun (or Earth for Moon). V: Velocity (relative).
venus_text = Text("", origin=(0, 0), scale=1, background=True, billboard=True, color=color.white, unlit=True)
earth_text = Text("", origin=(0, 0), scale=1, background=True, billboard=True, color=color.white, unlit=True)
moon_text = Text("", origin=(0, 0), scale=1, background=True, billboard=True, color=color.white, unlit=True)
mars_text = Text("", origin=(0, 0), scale=1, background=True, billboard=True, color=color.white, unlit=True)

# ----------------------
# Planet Info Function (Data Source for Dashboard)
# ----------------------
def get_planet_data():
    """Calculates current distance (to Sun) and orbit speed for visible bodies."""
    data = {}

    # Venus
    if show_venus and venus.enabled:
        dist = math.sqrt(venus.x**2 + venus.y**2)
        speed = 5.2 * 1.6 * ORBIT_SPEED_MULTIPLIER * time_scale
        data["Venus"] = (dist, speed)

    # Earth
    if show_earth and earth.enabled:
        dist = math.sqrt(earth.x**2 + earth.y**2)
        speed = 8.0 * 1.0 * ORBIT_SPEED_MULTIPLIER * time_scale
        data["Earth"] = (dist, speed)

    # Moon (Distance relative to Earth)
    if show_moon and moon.enabled and show_earth:
        dist = math.sqrt((moon.x - earth.x)**2 + (moon.y - earth.y)**2)
        speed = 1.5 * 8.0 * ORBIT_SPEED_MULTIPLIER * time_scale
        data["Moon"] = (dist, speed)

    # Mars
    if show_mars and mars.enabled:
        dist = math.sqrt(mars.x**2 + mars.y**2)
        speed = 12.0 * 0.8 * ORBIT_SPEED_MULTIPLIER * time_scale
        data["Mars"] = (dist, speed)

    return data

```

Title

```python
# -----------------------
#  Ursina Update (Physics and Rendering Loop)
# -----------------------
def update():
    global t, time_scale, show_stats

    # Time progression
    if not is_paused:
        t += time.dt * ORBIT_SPEED_MULTIPLIER * time_scale

    angle = math.pi * 40 / 100
    spin = time.dt * 20 * rotation_speed_mult

    # --- Venus ---
    if show_venus:
        venus.enabled = True
        target_x = math.cos(t * 1.6 + angle) * 5.2
        target_y = math.sin(t * 1.6 + angle) * 5.2
        # Use lerp for smooth motion
        venus.x = lerp(venus.x, target_x, 4 * time.dt)
        venus.y = lerp(venus.y, target_y, 4 * time.dt)
        venus.rotation_y += spin

        venus_dist = math.sqrt(venus.x**2 + venus.y**2)
        venus_speed = 5.2 * 1.6 * ORBIT_SPEED_MULTIPLIER * time_scale
        venus_text.text = f"D: {venus_dist:.2f}\nV: {venus_speed:.2f}"
        venus_text.position = venus.world_position + Vec3(0, 1.5, 0)
        venus_text.enabled = show_stats
    else:
        venus.enabled = False
        venus_text.enabled = False

    # --- Earth ---
    if show_earth:
        earth.enabled = True
        target_x = math.cos(t * 1.0 + angle * 2) * 8.0
        target_y = math.sin(t * 1.0 + angle * 2) * 8.0
        # Use lerp for smooth motion
        earth.x = lerp(earth.x, target_x, 4 * time.dt)
        earth.y = lerp(earth.y, target_y, 4 * time.dt)
        earth.rotation_y += spin

        earth_dist = math.sqrt(earth.x**2 + earth.y**2)
        earth_speed = 8.0 * 1.0 * ORBIT_SPEED_MULTIPLIER * time_scale
        earth_text.text = f"D: {earth_dist:.2f}\nV: {earth_speed:.2f}"
        earth_text.position = earth.world_position + Vec3(0, 1.5, 0)
        earth_text.enabled = show_stats
    else:
        earth.enabled = False
        earth_text.enabled = False

```

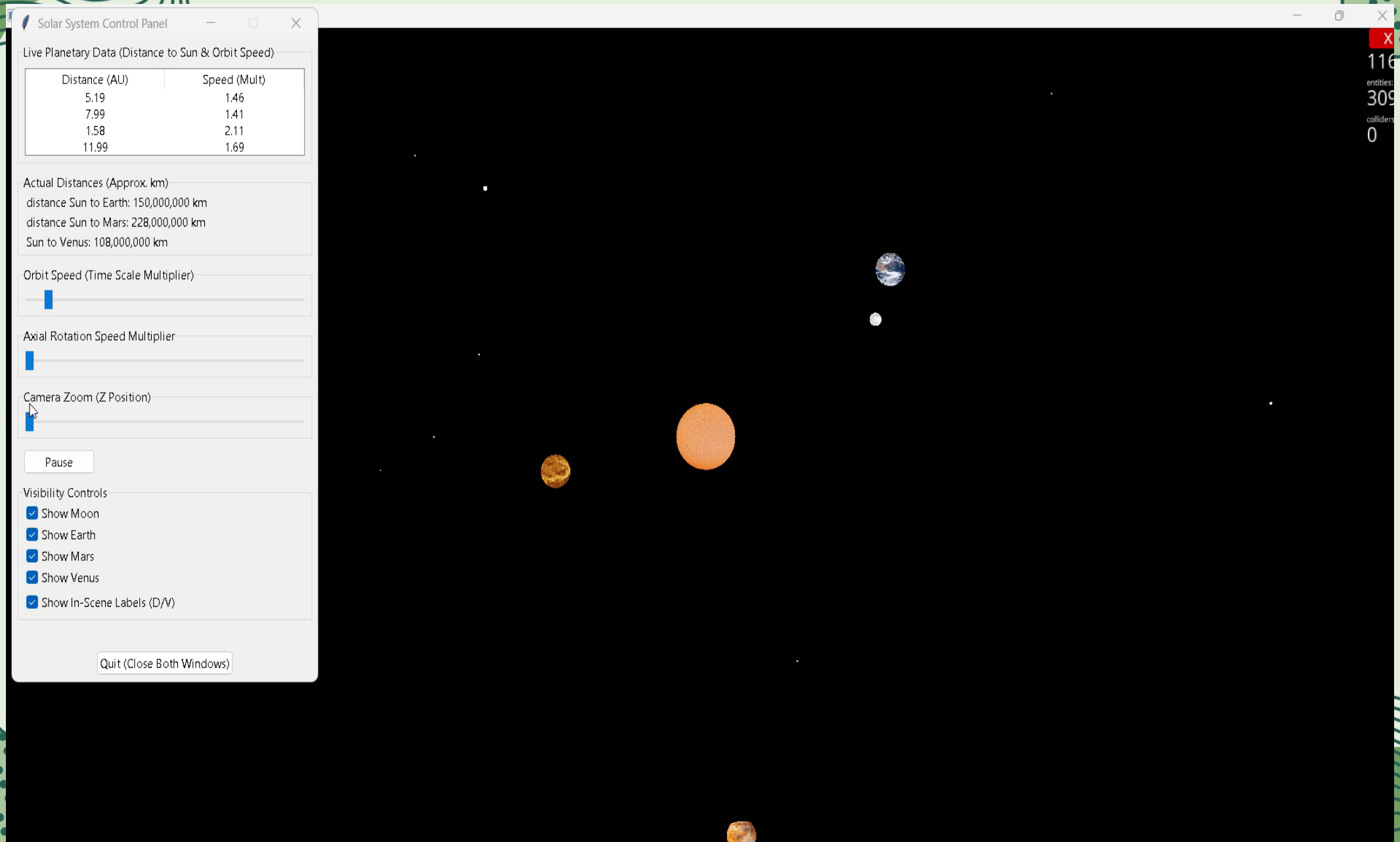
Title

```python
# --- Moon ---
    if show_moon and show_earth:
        moon.enabled = True
        target_x = earth.x + math.cos(t * 8.0) * 1.5
        target_y = earth.y + math.sin(t * 8.0) * 1.5
        # Use lerp for smooth motion
        moon.x = lerp(moon.x, target_x, 6 * time.dt)
        moon.y = lerp(moon.y, target_y, 6 * time.dt)
        moon.rotation_y += spin

        moon_dist_rel_earth = math.sqrt((moon.x-earth.x)**2 + (moon.y-earth.y)**2)
        moon_speed_rel_earth = 1.5 * 8.0 * ORBIT_SPEED_MULTIPLIER * time_scale
        moon_text.text = f"D_E: {moon_dist_rel_earth:.2f}\nV_E: {moon_speed_rel_earth:.2f}"
        moon_text.position = moon.world_position + Vec3(0, 0.8, 0)
        moon_text.enabled = show_stats
    else:
        moon.enabled = False
        moon_text.enabled = False

    # --- Mars ---
    if show_mars:
        mars.enabled = True
        target_x = math.cos(t * 0.8 + angle * 3) * 12.0
        target_y = math.sin(t * 0.8 + angle * 3) * 12.0
        # Use lerp for smooth motion
        mars.x = lerp(mars.x, target_x, 4 * time.dt)
        mars.y = lerp(mars.y, target_y, 4 * time.dt)
        mars.rotation_y += spin

        mars_dist = math.sqrt(mars.x**2 + mars.y**2)
        mars_speed = 12.0 * 0.8 * ORBIT_SPEED_MULTIPLIER * time_scale
        mars_text.text = f"D: {mars_dist:.2f}\nV: {mars_speed:.2f}"
        mars_text.position = mars.world_position + Vec3(0, 1.5, 0)
        mars_text.enabled = show_stats
    else:
        mars.enabled = False
        mars_text.enabled = False

    # Sun and Camera
    sun.rotation_y += spin
    camera.z = lerp(camera.z, zoom_level, 3 * time.dt)


```

```
# Initial table population
 for name in ["Venus", "Earth", "Moon", "Mars"]:
     table.insert("", "end", iid=name, text=name, values=("N/A", "N/A"))

 def refresh_table():
     """Updates the Treeview with live data."""
     planet_data = get_planet_data()

     # Define order and names for display
     for name in ["Venus", "Earth", "Moon", "Mars"]:
         if name in planet_data:
             # FIX: Use actual dist and speed variables
             dist, speed = planet_data[name]
             table.item(name, values=(f"{dist:.2f}", f"{speed:.2f}"))
         else:
             # Planet is disabled
             table.item(name, values=("-", "-"))

     # Schedule the next refresh
     root.after(200, refresh_table) # Refresh every 200ms

 # Start the dashboard refresh loop
 refresh_table()

 # === Real-World Data Display ===
 real_data_frame = ttk.LabelFrame(root, text="Actual Distances (Approx. km)")
 real_data_frame.pack(fill="x", padx=8, pady=6)

 ttk.Label(real_data_frame, text=f"distance Sun to Earth: 150,000,000 km").pack(anchor="w", padx=8)
 ttk.Label(real_data_frame, text=f"distance Sun to Mars: 228,000,000 km").pack(anchor="w", padx=8)
 ttk.Label(real_data_frame, text=f"Sun to Venus: 108,000,000 km").pack(anchor="w", padx=8, pady=(0, 4))

 # === SLIDERS AND CONTROLS ===

 # Orbit Speed
 speed_frame = ttk.LabelFrame(root, text="Orbit Speed (Time Scale Multiplier)")
 speed_frame.pack(fill="x", padx=8, pady=6)
 speed_slider = ttk.Scale(speed_frame, from_=0.0, to=5.0, orient="horizontal", command=set_speed)
 speed_slider.set(time_scale)
 speed_slider.pack(fill="x", padx=8, pady=6)

 # Rotation Speed
 rot_frame = ttk.LabelFrame(root, text="Axial Rotation Speed Multiplier")
 rot_frame.pack(fill="x", padx=8, pady=6)
 rot_slider = ttk.Scale(rot_frame, from_=0.1, to=5.0, orient="horizontal", command=set_rot)
 rot_slider.set(rotation_speed_mult)
 rot_slider.pack(fill="x", padx=8, pady=6)

 # Zoom
 zoom_frame = ttk.LabelFrame(root, text="Camera Zoom (Z Position)")
 zoom_frame.pack(fill="x", padx=8, pady=6)
 zoom_slider = ttk.Scale(zoom_frame, from_=-150, to=100, orient="horizontal", command=set_zoom)
 zoom_slider.set(zoom_level)
 zoom_slider.pack(fill="x", padx=8, pady=6)

 # --- Pause/Resume ---
 control_frame = ttk.Frame(root)
 control_frame.pack(fill="x", padx=8, pady=6)
 pause_btn = ttk.Button(control_frame, text="Pause", command=toggle_pause)
 pause_btn.pack(side="left", padx=8)

```

```
 # --- Planet Visibility ---
  check_frame = ttk.LabelFrame(root, text="Visibility Controls")
  check_frame.pack(fill="x", padx=8, pady=6)

  var_moon = tk.BooleanVar(value=show_moon)
  var_earth = tk.BooleanVar(value=show_earth)
  var_mars = tk.BooleanVar(value=show_mars)
  var_venus = tk.BooleanVar(value=show_venus)
  var_stats = tk.BooleanVar(value=show_stats)

  ttk.Checkbutton(check_frame, text="Show Moon", variable=var_moon,
                  command=lambda: toggle_planet('moon', var_moon)).pack(anchor="w", padx=8)
  ttk.Checkbutton(check_frame, text="Show Earth", variable=var_earth,
                  command=lambda: toggle_planet('earth', var_earth)).pack(anchor="w", padx=8)
  ttk.Checkbutton(check_frame, text="Show Mars", variable=var_mars,
                  command=lambda: toggle_planet('mars', var_mars)).pack(anchor="w", padx=8)
  ttk.Checkbutton(check_frame, text="Show Venus", variable=var_venus,
                  command=lambda: toggle_planet('venus', var_venus)).pack(anchor="w", padx=8)
  ttk.Checkbutton(check_frame, text="Show In-Scene Labels (D/V)", variable=var_stats,
                  command=lambda: toggle_stats(var_stats)).pack(anchor="w", padx=8, pady=(4, 8))

  # --- Quit Button ---
  def quit_all():
      root.quit()
      try:
          app.quit()
      except Exception as e:
          print(f"Error during Ursina quit: {e}")

  quit_btn = ttk.Button(root, text="Quit (Close Both Windows)", command=quit_all)
  quit_btn.pack(side="bottom", pady=8)

  root.mainloop()
# ----------------------
# 4. Threading and Run
# ----------------------
tk_thread = threading.Thread(target=start_tkinter, daemon=True)
tk_thread.start()
app.run()

```

Video Display



Problem :

- ឧស្សាហ៍ error code ច្រើន
- មានមុខងារខ្លះដែលយើងមិនសូវយល់ច្បាស់
- ការគណនាល្បឿនរបស់ភពនីមួយៗ

Solution :

- សួរAiខ្លះ
- ស្វែងរកព័ត៌មានក្នុងប្រព័ន្ធInternet

Github

Link github :
https://github.com/Sokengchhay/Solar-System-With-Python-Turtle-RUPP-.git

Google Drive

Link Google Drive :
https://drive.google.com/drive/folders/1jD3S9LlYiRbYEYRfIuk729gek9-5oAvY?usp=sharing


Thank
you