Počítačová simulace teoretického pohonu pomocí umělé singularity tvořená 3D virtualizačním a animačním programem Blender.

Poznámka:
Evidentně zcela nereálné a ve skutečnosti naprosto neproveditelné.

Teorie:
Předpokládejme člověkem vytvořenou, 200 tunovou Schwarzschildovu statickou černou díru, která se vypaří za méně než jednu sekundu. Čím menší je černá díra, tím rychleji se vypařuje.
Předpokládejme zařízení v bezpečné vzdálenosti od Země, o hmotnosti 100 tun, která před sebou generuje černou díru. Tato vygenerovaná černá díra musí mít větší hmotnost, než zařízení, aby svou gravitací přitáhlo zařízení k sobě.
Než zařízení dosáhne černé díry, černá díra se zatím vypaří díky kvantovému mechanismu zvanému Hawkingovo záření.
Zařízení generující černou díru by tak pravděpodobně zůstalo v přímém kurzu, mohlo by nabrat velmi vysokou rychlost právě na okraji hroutícího se horizontu událostí vypařující se černé díry. Mohlo by tak generovat více černých děr k dalšímu urychlení letu. Mohlo by tak i brzdit. Mohlo by měnit směr letu.

Určení gravitační síly mezi černou dírou a zařízením.
F = G*(m1*m2)/r^2
F: přitažlivá síla
G: gravitační konstanta
m1: hmotnost černé díry
m2: hmotnost zařízení
r: vzdálenost mezi m1 a m2

Rychlost vypařování černé díry pomocí Hawkingova záření.
Vzorec pro Hawkingovu teplotu:
T = (h*c^3)/(8*k*G*M)
T: teplota černé díry
h: Planckova konstanta
k: Boltzmanova konstanta
c: rychlost světla
G: gravitační konstanta
M: hmotnost černé díry

Zjednodušený vztah pro rychlost, odvozen z kruhového pohybu v omezeném prostoru.
v = √(G*M/r)
v: rychlost
G: gravitační konstanta
M: hmotnost černé díry
r: vzdálenost od středu černé díry (roven poloměru horizontu událostí)

Schwarzschildův poloměr je charakteristická vzdálenost pro každou hmotnost. Je to poloměr koule, do které musí být veškerá hmota o dané hmotnosti stlačena, aby již žádná síla nemohla odvrátit její zhroucení do gravitační singularity.
Rs = 2*G*M/c^2
Rs: Schwarzschildův poloměr
G: gravitační konstanta
M: hmotnost černé díry
c: rychlost světla

Konstanty.
G = (6,674 30 ± 0,000 15)×10^−11 m3·kg^−1·s^−2
h = 6,62607015×10^−34 J·s
k = 1,380649×10^−23 J·K^−1

---

1. Příprava prostředí a základních prvků v Blenderu. https://www.blender.org/
Blender s aktivovaným API pro Python: Blender (verze 2.8 a vyšší) umožňuje programování v Pythonu přímo ve svém prostředí.
Python 3.x: Zajištění, že všechny potřebné moduly jsou nainstalovány (Blender obsahuje svůj vlastní Python, ale některé moduly je možné přidat externě).
a) Zařízení (Generátor): Kvádřík nebo váleček, který generuje černé díry. Bude to hlavní pohyblivý objekt scény.
b) Singularita (Černá díra): Bod s křížkem, představující černou díru. Bude mít odpovídající fyzikální parametry jako hmotnost a horizont událostí.
c) Horizont událostí: Koule kolem černé díry, jejíž poloměr odpovídá Schwarzschildovu poloměru.

2. Uživatelské rozhraní
V Blenderu můžeme vytvořit jednoduché uživatelské rozhraní pomocí skriptů v Pythonu. Tento panel může obsahovat následující prvky:
Vstupní pole pro zadání parametrů:
a) Hmotnost černé díry
b) Počáteční vzdálenost zařízení od černé díry
c) Počáteční rychlost zařízení
Tlačítka:
a) Potvrdit zadání: Uloží hodnoty z polí pro výpočty.
b) Start: Zahájí simulaci, při které se budou aplikovat fyzikální výpočty na zařízení.
c) Reset: Vrátí vše do původního stavu.

3. Implementace fyzikálních vzorců v Pythonu
Na základě fyzikálních vzorců, které jsme již odvodili, vytvoříme funkce, které budou počítat:
a) Poloměr horizontu událostí pro danou hmotnost černé díry.
b) Gravitační zrychlení zařízení způsobené černou dírou.
c) Aktualizaci rychlosti zařízení na základě gravitačního zrychlení.
d) Čas vypaření černé díry podle Hawkingova záření.
Tyto výpočty můžeme implementovat jako funkce v Pythonu a při spuštění simulace budou tyto funkce aktualizovat stav a rychlost objektů na scéně.

4. Animace pohybu zařízení a vypařování černé díry
Pohyb zařízení směrem k černé díře lze simulovat aktualizací jeho pozice v každém časovém kroku simulace. Když se zařízení dostane k horizontu událostí, černá díra se vypaří a zařízení pokračuje dál setrvačností, nebo se generuje nová černá díra pro další urychlení.

5. Výsledkový výstup
Výsledky simulace (rychlost zařízení, vypaření černé díry apod.) můžeme zobrazit v reálném čase na panelu jako textový výstup.

Polotovar Python kódu pro Blender:

```
import bpy
import math

# Fyzikální konstanty
G = 6.674 * 10**-11  # Gravitational constant
c = 3 * 10**8  # Speed of light

# Funkce pro výpočet Schwarzschildova poloměru
def schwarzschild_radius(mass):
    return 2 * G * mass / c**2

# Funkce pro vypočítání rychlosti zařízení ve vzdálenosti r od černé díry
def velocity_near_black_hole(mass, distance):
    return math.sqrt(2 * G * mass / distance)

# Funkce pro čas vypaření (Hawkingovo záření)
def evaporation_time(mass):
    hbar = 1.0545718 * 10**-34
    return (5120 * math.pi * G**2 * mass**3) / (hbar * c**4)

# Uživatelské rozhraní v Blenderu
class SingularDrivePanel(bpy.types.Panel):
    bl_label = "Singulární pohon"
    bl_idname = "OBJECT_PT_singular_drive"
    bl_space_type = 'VIEW_3D'
    bl_region_type = 'UI'
    bl_category = 'Simulace'

    def draw(self, context):
        layout = self.layout
        scene = context.scene

        # Vstupní pole
        layout.prop(scene, "black_hole_mass")
        layout.prop(scene, "device_distance")
        layout.prop(scene, "initial_velocity")

        # Tlačítka
        layout.operator("wm.confirm_setup")
        layout.operator("wm.start_simulation")
        layout.operator("wm.reset_simulation")

        # Výstup
        layout.label(text=f"Current Velocity: {scene.device_velocity:.2f} m/s")
        layout.label(text=f"Black Hole Lifetime: {scene.black_hole_lifetime:.2f} s")

# Tlačítka pro ovládání simulace
class ConfirmSetupOperator(bpy.types.Operator):
    bl_idname = "wm.confirm_setup"
    bl_label = "Potvrdit zadání"

    def execute(self, context):
        scene = context.scene
        mass = scene.black_hole_mass
        distance = scene.device_distance
        scene.black_hole_radius = schwarzschild_radius(mass)
        scene.black_hole_lifetime = evaporation_time(mass)
        return {'FINISHED'}

class StartSimulationOperator(bpy.types.Operator):
    bl_idname = "wm.start_simulation"
    bl_label = "Start"

    def execute(self, context):
        # Zde se bude provádět simulace a aktualizace rychlosti
        return {'FINISHED'}

class ResetSimulationOperator(bpy.types.Operator):
    bl_idname = "wm.reset_simulation"
    bl_label = "Reset"

    def execute(self, context):
        # Obnovení výchozích hodnot
        return {'FINISHED'}

# Registrace tříd a vlastností pro Blender
def register():
    bpy.utils.register_class(SingularDrivePanel)
    bpy.utils.register_class(ConfirmSetupOperator)
    bpy.utils.register_class(StartSimulationOperator)
    bpy.utils.register_class(ResetSimulationOperator)

    bpy.types.Scene.black_hole_mass = bpy.props.FloatProperty(name="Hmotnost černé díry",
default=200000)
    bpy.types.Scene.device_distance = bpy.props.FloatProperty(name="Počáteční vzdálenost",
default=10.0)
    bpy.types.Scene.initial_velocity = bpy.props.FloatProperty(name="Počáteční rychlost",
default=0.0)
    bpy.types.Scene.black_hole_radius = bpy.props.FloatProperty(name="Poloměr horizontu",
default=0.0)
    bpy.types.Scene.black_hole_lifetime = bpy.props.FloatProperty(name="Čas vypaření",
default=0.0)
    bpy.types.Scene.device_velocity = bpy.props.FloatProperty(name="Rychlost zařízení",
default=0.0)

def unregister():
    bpy.utils.unregister_class(SingularDrivePanel)
    bpy.utils.unregister_class(ConfirmSetupOperator)
    bpy.utils.unregister_class(StartSimulationOperator)
    bpy.utils.unregister_class(ResetSimulationOperator)

if __name__ == "__main__":
    register()
```