Ray tracing prototypes

Referencias:
http://spiro.fisica.unipd.it/~antonell/schwarzschild/
http://rantonels.github.io/starless/
Orbits of massless particles in the Schwarzschild metric: Exact solutions American Journal of Physics 82, 564 (2014)

Derivación del modelo

Se supondrá que c = 1 y $r_s = 1$. La métrica está dada por:

$$ds^2 = \left(1-\frac{1}{r}\right)dt^2 - \left(1-\frac{1}{r}\right)^{-1}dr^2 - r^2 d\theta^2 - r^2 \operatorname{sen}(\theta)^2 d\phi^2$$

si nos restringimos al plano $\theta = \frac{\pi}{2} \Rightarrow d\theta = 0$

$$ds^2 = \left(1-\frac{1}{r}\right)dt^2 - \left(1-\frac{1}{r}\right)^{-1}dr^2 - r^2 d\phi^2 \quad (1)$$

cuyo tensor métrico es

$$g_{\mu\nu} = \begin{pmatrix} \left(1-\frac{1}{r}\right) & 0 & 0 & 0 \\ 0 & -\left(1-\frac{1}{r}\right)^{-1} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -r^2 \end{pmatrix}$$

$$\Rightarrow \frac{ds^2}{d\tau^2} = \left(1-\frac{1}{r}\right)\frac{dt^2}{d\tau^2} - \left(1-\frac{1}{r}\right)^{-1}\frac{dr^2}{d\tau^2} - r^2\frac{d\phi^2}{d\tau^2}$$

Para una partícula sin masa $ds^2 = 0$

$$\Rightarrow \left(1-\frac{1}{r}\right)\left(\frac{dt}{d\tau}\right)^2 - \left(1-\frac{1}{r}\right)^{-1}\left(\frac{dr}{d\tau}\right)^2 - r^2\left(\frac{d\phi}{d\tau}\right)^2 = 0 \quad (2)$$

Dado que la métrica es independiente de t y ϕ existen dos vectores de Killing

$$K^0 = (1, 0, 0, 0) \quad \text{y} \quad K^3 = (0, 0, 0, 1)$$

cuya forma covariante es:

$$K_0 = g_{\mu\nu}K^0 = \left(1-\frac{1}{r}, 0, 0, 0\right)$$

$$K_3 = g_{\mu\nu}K^3 = \left(0, 0, 0, -r^2\right)$$

La existencia de los vectores de Killing implica que se debe cumplir

$$\frac{d}{d\tau}\left(K_\mu \frac{dX^\mu}{d\tau}\right) = 0$$

Al aplicar esta condición en (1) se obtiene

$$\left(1-\frac{1}{r}\right)\frac{dt}{d\tau}=e \quad y \quad r^2\frac{d\phi}{d\tau}=\ell$$ donde e y l son constantes

$$\Rightarrow \left(\frac{dt}{d\tau}\right)^2=e^2\left(1-\frac{1}{r}\right)^{-2}, \quad \left(\frac{d\phi}{d\tau}\right)^2=\frac{\ell^2}{r^4}$$

Al sustituir en (2) queda

$$e^2\left(1-\frac{1}{r}\right)^{-1}-\left(1-\frac{1}{r}\right)^{-1}\left(\frac{dr}{d\tau}\right)^2-\frac{\ell^2}{r^2}=0 \quad \Rightarrow \left(1-\frac{1}{r}\right)^{-1}\left(\frac{dr}{d\tau}\right)^2=e^2\left(1-\frac{1}{r}\right)^{-1}-\frac{\ell^2}{r^2}$$

$$\Rightarrow \left(\frac{dr}{d\tau}\right)^2=e^2-\left(1-\frac{1}{r}\right)\frac{\ell^2}{r^2}$$

$$\left(\frac{dr}{d\tau}\right)^2=\left(\frac{dr}{d\phi}\frac{d\phi}{d\tau}\right)^2=\underbrace{\left(\frac{\ell}{r^2}\frac{dr}{d\phi}\right)^2=e^2-\left(1-\frac{1}{r}\right)\frac{\ell^2}{r^2}}$$

$$\frac{\ell^2}{r^4}\left(\frac{dr}{d\phi}\right)^2=e^2-\left(1-\frac{1}{r}\right)\frac{\ell^2}{r^2}$$

$$\left(\frac{dr}{d\phi}\right)^2=\frac{e^2r^4}{\ell^2}-\left(1-\frac{1}{r}\right)r^2$$ Haciendo $b=\frac{\ell}{e}$

$$\left(\frac{dr}{d\phi}\right)^2=\frac{r^4}{b^2}-\left(1-\frac{1}{r}\right)r^2$$

Haciendo $u=\frac{1}{r} \Rightarrow r=\frac{1}{u}$, $\dot{r}=-\frac{1}{u^2}\dot{u}$

$$\left(-\frac{1}{u^2}\frac{du}{d\phi}\right)^2=\frac{1}{b^2u^4}-\left(1-u\right)\frac{1}{u^2}$$

$$\frac{1}{u^4}\left(\frac{du}{d\phi}\right)^2=\frac{1}{b^2u^4}-\frac{1-u}{u^2}$$

$$\left(\frac{du}{d\phi}\right)^2=\frac{1}{b^2}-u^2(1-u)$$

Se deriva todo respecto a ϕ

$$2\cancel{\dot{U}(\phi)}\,\ddot{U}(\phi) = -2\,(1-U(\phi))\,U(\phi)\,\cancel{\dot{U}(\phi)} + U(\phi)^2\,\cancel{\dot{U}(\phi)}$$

$$2\ddot{U}(\phi) = -2(1-U(\phi))U(\phi) + U(\phi)^2$$

$$\ddot{U} = \frac{U^2}{2} - (1-U)U = \frac{U^2}{2} - U + U^2$$

$$= \frac{3}{2}U^2 - U$$

$\Rightarrow$ se llega a la ec. de mov

$$\ddot{U}(\phi) = -U\left(1 - \frac{3}{2}U\right)$$

$$\ddot{U}(\phi) + U(\phi) = \frac{3}{2}U^2(\phi)$$

Esto es una ecuación de Binet

$$\ddot{U} + U = -\frac{1}{mh^2U^2}F(U)$$

$\Rightarrow$

$$-\frac{1}{mh^2U^2}F(U) = \frac{3}{2}U^2$$

$$F(U) = -\frac{3}{2}h^2U^4$$

haciendo el cambio de variable

$$U = \frac{1}{r}$$

$$F(r) = -\frac{3}{2}h^2\frac{1}{r^4}$$

En forma vectorial

$$F(\vec{r}) = -\frac{3}{2}h^2\frac{\hat{r}}{|\vec{r}|^5}$$

Mundo a simular

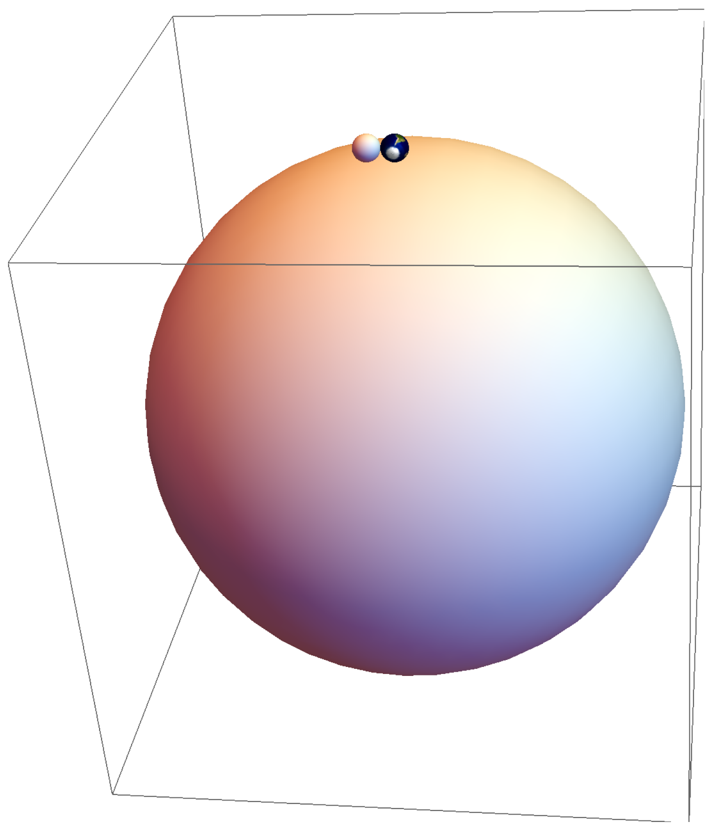
In[ ]:=
```
earthTexture = Import@
   FileNameJoin[{ParentDirectory[NotebookDirectory[]], "Textures", "earthmap.jpg"}];
skyTexture = Import@FileNameJoin[{ParentDirectory[NotebookDirectory[]],
     "Textures", "starbackground.jpg"}];
```

In[ ]:=
```
planet = SphericalPlot3D[0.5 , {u, 0, Pi}, {v, 0, 2 Pi},
   Mesh → None,
   TextureCoordinateFunction → ({#5, 1 - #4} &),
   PlotStyle → Directive[Specularity[White, 10], Texture[earthTexture]],
   Lighting → "Neutral",
   Axes → False,
   RotationAction → "Clip",
   Boxed -> False
  ];
bigAssSphere = Graphics3D[Sphere[{0, 0, 0}, 10], Boxed -> False];
```

```
In[•]:= solidSphere = Graphics3D[Sphere[{0, 0, 0}, 0.5], Boxed -> False];

In[•]:= Graphics3D[{Translate[planet[[1]], {0, 0, -1}],
          Translate[solidSphere[[1]], {1, 0, -1}], Translate[bigAssSphere[[1]], {0, -10.5, -1}]}]
```

Out[•]=

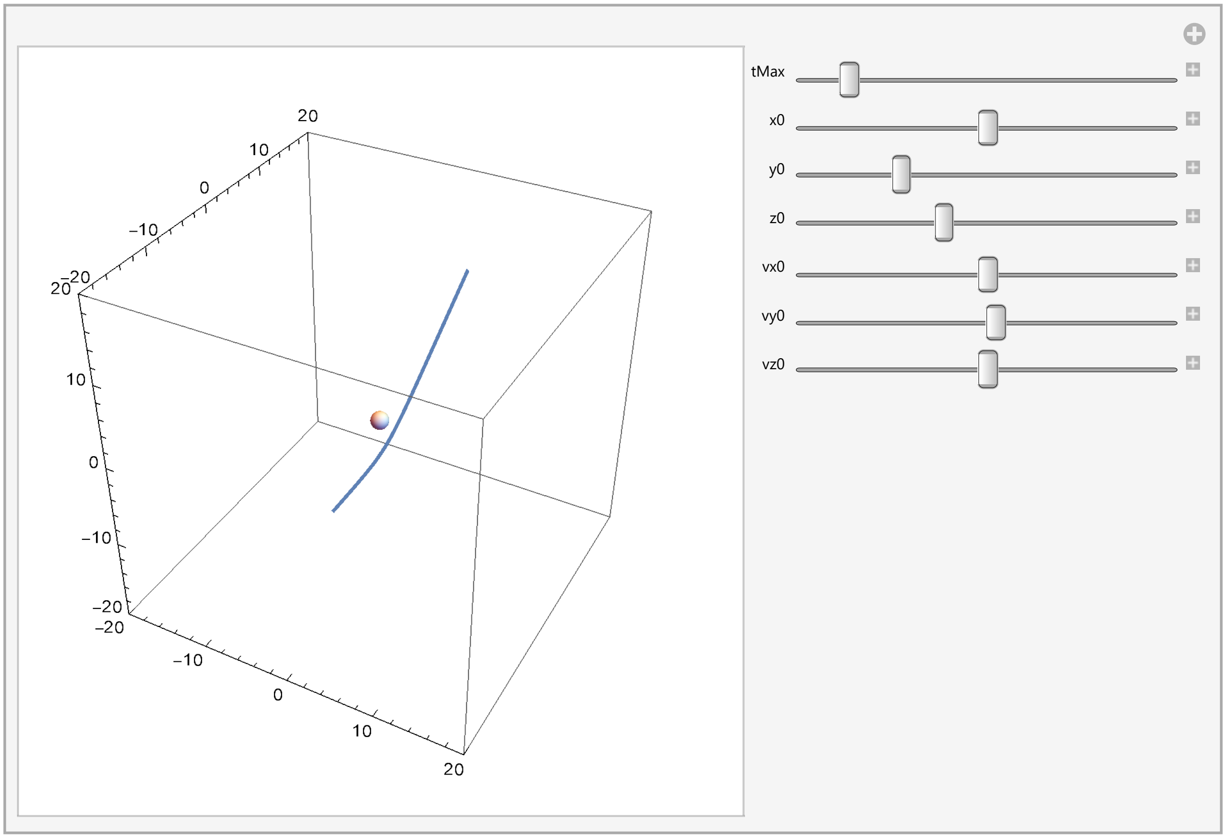


Desviación del rayo en presencia de cuerpo masivo

In[ ]:=

```
SquaredNorm[vec_] := vec.vec;
DynamicModule[{func, h2},
 Manipulate[
  h2 = SquaredNorm[Cross[{x0, y0, z0}, {vx0, vy0, vz0}]];
  func = NDSolveValue[
    {
     x''[t] == -1.5 h2 x[t] / SquaredNorm[{x[t], y[t], z[t]}]^(5/2),
     y''[t] == -1.5 h2 y[t] / SquaredNorm[{x[t], y[t], z[t]}]^(5/2),
     z''[t] == -1.5 h2 z[t] / SquaredNorm[{x[t], y[t], z[t]}]^(5/2),
     x[0] == x0, y[0] == y0, z[0] == z0,
     x'[0] == vx0, y'[0] == vy0, z'[0] == vz0
    },
    {x[t], y[t], z[t]},
    {t, 0, tMax}
   ];
  Show[
   ParametricPlot3D[func, {t, 0, tMax}, PlotRange → {{-20, 20}, {-20, 20}, {-20, 20}}],
   Graphics3D[Sphere[{0, 0, 0}, 1]]
  ],
  {{tMax, 100}, 1, 1000},
  {{x0, 0}, -20, 20},
  {{y0, -10}, -20, 20},
  {{z0, -5}, -20, 20},
  {{vx0, 0}, -20, 20},
  {{vy0, 1}, -20, 20},
  {{vz0, 0}, -20, 20},
  SaveDefinitions → True
 ]
]
```

Out[•]=



In[•]:=

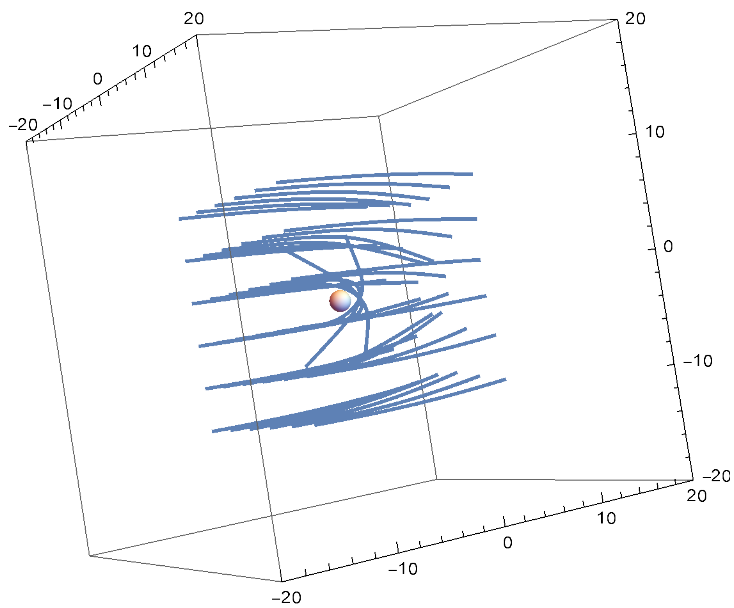
```
RayTrajectory[{x0_, y0_, z0_}, {vx0_, vy0_, vz0_}, tMax_] := Module[{x, y, z, h2},
   h2 = SquaredNorm[Cross[{x0, y0, z0}, {vx0, vy0, vz0}]];
   func = NDSolveValue[
     {
      x''[t] == -1.5 h2 x[t] / SquaredNorm[{x[t], y[t], z[t]}]^(5/2),
      y''[t] == -1.5 h2 y[t] / SquaredNorm[{x[t], y[t], z[t]}]^(5/2),
      z''[t] == -1.5 h2 z[t] / SquaredNorm[{x[t], y[t], z[t]}]^(5/2),
       x[0] == x0, y[0] == y0, z[0] == z0,
       x'[0] == vx0, y'[0] == vy0, z'[0] == vz0
     },
     {x[t], y[t], z[t]},
     {t, 0, tMax}
    ];
   Return[func];
  ];
```

In[•]:=

```
rays = Table[If[x ≠ 0 && z ≠ 0, RayTrajectory[{x, -10, z}, {0.0, 1, 0.0}, 20], Nothing],
   {z, -10, 10, 4}, {x, -10, 10, 4}];
```

In[◦]:=
```
Show[
 ParametricPlot3D[rays, {t, 0, 20}, PlotRange → {{-20, 20}, {-20, 20}, {-20, 20}}],
 Graphics3D[Sphere[{0, 0, 0}, 1]]
]
```

Out[◦]=




```
WhenEvent[SquaredNorm[{x, y, z}] < 1, "StopIntegration"]
```

```
SquaredNorm[{x - offsetX, y - offsetY, z - offsetZ}]
```

In[∘]:=

```
PositionableRayTrajectory[{offsetX_, offsetY_, offsetZ_},
   {x0_, y0_, z0_}, {vx0_, vy0_, vz0_}, tMax_] := Module[{x, y, z, h2, func},
   h2 = SquaredNorm[Cross[{x0, y0, z0}, {vx0, vy0, vz0}]];
   func = NDSolveValue[
     {
      x''[t] == -1.5 h2 (x[t] - offsetX) / SquaredNorm[{x[t] - offsetX, y[t] - offsetY, z[t] - offsetZ}]^(5/2),
      y''[t] == -1.5 h2 (y[t] - offsetY) / SquaredNorm[{x[t] - offsetX, y[t] - offsetY, z[t] - offsetZ}]^(5/2),
      z''[t] == -1.5 h2 (z[t] - offsetZ) / SquaredNorm[{x[t] - offsetX, y[t] - offsetY, z[t] - offsetZ}]^(5/2),
      x[0] == x0, y[0] == y0, z[0] == z0,
      x'[0] == vx0, y'[0] == vy0, z'[0] == vz0,
      WhenEvent[SquaredNorm[{x[t] - offsetX, y[t] - offsetY, z[t] - offsetZ}] < 1,
       "StopIntegration"]
     },
     {x[t], y[t], z[t]},
     {t, 0, tMax},
     MaxStepSize → 0.01
    ];
   Return[func];
  ];
```

In[∘]:=

```
PositionableRayTrajectory[{0, 0, 0}, {2, -10, 0}, {0.0, 1, 0.0}, 20]
```

Out[∘]=

```
{InterpolatingFunction[ Domain: {{0., 10.4}} Output: scalar ][t],
 InterpolatingFunction[ Domain: {{0., 10.4}} Output: scalar ][t],
 InterpolatingFunction[ Domain: {{0., 10.4}} Output: scalar ][t]}
```

In[ ]:=
```
DynamicModule[{rays},
 Manipulate[
  rays = Table[If[x ≠ 0 && z ≠ 0, PositionableRayTrajectory[{offsetX, offsetY, offsetZ},
      {x, -10, z}, {0.0, 1, 0.0}, tMax], Nothing], {z, -10, 10, 4}, {x, -10, 10, 4}];
  Show[
   ParametricPlot3D[rays, {t, 0, tMax}, PlotRange → {{-20, 20}, {-20, 20}, {-20, 20}}],
   Graphics3D[Sphere[{offsetX, offsetY, offsetZ}, 1]]
  ]
  ,
  {{offsetX, 0}, -10, 10},
  {{offsetY, 0}, -10, 10},
  {{offsetZ, 0}, -10, 10},
  {{tMax, 20}, 0.1, 50}
 ]
]
```

Out[ ]=