```
GlowScript 2.6 VPython
## 2-Body Problem Version 3 Spring 2013 - Fall 2017 A. O. Hausknecht
## This is a port of a Visual Python program written for the CSUMS seminar.
## This version includes
##   1. A menu to change the initial conditions,
##   2. A Start-Pause button to start and pause the simulation,
##   3. A Reset button to reset the simulation,
##   4. A Show-Hide dynamic velocity labels checkbox,
##   5. A brief statement of the model using laTeX.
##   6. A plot of the speeds of the bodies.

scene = canvas(title = '<b><big>Two-Body Problem Simulation</big></b><br>')
scene.width=500
scene.height=500
scene.center=vector(0,0,4)
scene.range = 15e10
SymmetricMotion = 0
LoopLeftMotion = 1
NearCircularOrbit = 2

gCurrentMotion = 0
gVelocitiesVisible = True
gAnimationRate= 40

gStarted = False
gPaused = True


def setInitCondition(menu):
    s = menu.selected

    if gStarted or gPaused:
        reset()



    if s == 'v1(0) = -v2(0) ≠ 0; m1 = m2':
        initialize(SymmetricMotion)
    elif s == "v1(0) = 0, v2(0) ≠ 0; m1 = m2":
        initialize(LoopLeftMotion)
    else: # s = "v1(0) = 0, v2(0) ≠ 0; m1 >> m2":
        initialize(NearCircularOrbit)


def setInitCondition2(menu):
```

```
    s = menu.selected
    if s == "Symetric":
        initialize(SymmetricMotion)
    elif s == "Loop":
        initialize(LoopLeftMotion)
    else: # s = "v01 = 0, v02 ≠ 0; m1 >> m2":
        initialize(NearCircularOrbit)



def setAnimationRate(menu):
    global gAnimationRate
    s = menu.selected
    if s == "Show":
        gAnimationRate = 10
    elif s == "Medium":
        gAnimationRate = 20
    elif s == "Fast":
        gAnimationRate = 40
    else:
        gAnimationRate = 80

    alert(gAnimationRate)

def reset():
    global gCurrentMotion
    global gPaused
    global gStarted
    gPaused = True
    gStarted = False
    scene.range = 15e10
    startPauseButton.text = "Start"
    startPauseButton.background = color.green
    initialize(gCurrentMotion)



def showHideVelocityLabels():
    global gVelocitiesVisible
    global gVelocity1Label
    global gVelocity2Label

    gVelocitiesVisible = !gVelocitiesVisible
    gVelocity1Label.visible = gVelocitiesVisible
    gVelocity2Label.visible = gVelocitiesVisible
```

```
91
92  def doStartPause():
93      global gStarted
94      global gPaused
95      if not gStarted:
96          gStarted = True
97          gPaused = False
98          startPauseButton.text = "Pause"
99          startPauseButton.background = color.red
100     elif not gPaused:
101          gPaused = True
102          startPauseButton.text = "Resume"
103          startPauseButton.background = color.cyan
104     else:
105          gPaused = False
106          startPauseButton.text = "Pause"
107          startPauseButton.background = color.red
108
109
110
111 scene.append_to_title("<b>Initial Condtitons:<b>")
112 menu(pos=scene.title_anchor, choices=['v1(0) = -v2(0) ≠ 0; m1 = m2',
113                                       "v1(0) = 0, v2(0) ≠ 0; m1 = m2",
114                                       "v1(0) = 0, v2(0) ≠ 0; m1 >> m2"],
115                                  selected = 'v1(0) = -v2(0) ≠ 0; m1 = m2', bind=setIn
116 scene.append_to_title(" ")
117 startPauseButton = button(pos=scene.title_anchor, bind=doStartPause, text="Start", ba
118 scene.append_to_title(" ")
119 resetButton = button(pos=scene.title_anchor, bind=reset, text="Reset", background=col
120
121 scene.append_to_title("\n <b>Show Velocites:<b>")
122 labelsCheckbox = checkbox(pos=scene.title_anchor, bind=showHideVelocityLabels, text='
123
124 scene.append_to_title("  <b>Animation Rate</b> ")
125 menu(pos=scene.title_anchor, choices=["Slow", "Medium", "Fast", "Very Fast"], selecte
126
127
128
129 ## Data Set
130 b1Mass = 1.5e30
131 b1P0   = vector(      0, 4.5e10, 0)
132 b1V0   = vector( 1.0e10,      0, 0) ##Change 1: 1.0e10*0
133 b1Radius = 0.5e10
134 b1Color = color.cyan
135 ##
```

```
b2Mass = 1.5e30  ##Change 2: 1.5e30/1e10
b2P0 = vector(      0, -4.5e10, 0)
b2V0 = vector(-1.0e10,       0, 0)
b2Radius = 0.5e10
b2Color = color.red
#
gT = 0; gDt = 0.1
gMaxTime = 100 ## Time variables
gGravity = 6.67e-11   ## Universal Constant of Gravitation

## Graphics Varaibles
gBody1 = undefined
gBody2 = undefined
gTheGraph = undefined; gCdotsV1 = undefined; gCdotsV2 = undefined
gFinished = False
gVelocity1Label = label(pos = b1P0*1.5, text = "v1 = <0, 0>", height = 14,
                        color = color.cyan, border = 8)
gVelocity2Label = label(pos = b2P0*1.5, text = "v2 = <0, 0>", height = 14,
                        color = color.red, border = 8)


def displayVelocity(v1, p1, v2, p2):
    global gVelocity1Label
    global gVelocity2Label

    vStr = " v1 = <{:5.3}, {:5.3}>".format(v1.x, v1.y)
    gVelocity1Label.text = vStr
    y1 =  p1.y*1.4; y2 = p2.y*1.4
    if (gVelocity1Label.pos.y < y1):
       gVelocity1Label.pos = vector(0, y1, 0)
    elif (gVelocity1Label.pos.y < y2):
       gVelocity1Label.pos = vector(0, y2, 0)

    vStr = " v2 = <{:5.3}, {:5.3}>".format(v2.x, v2.y)
    gVelocity2Label.text = vStr
    if (gVelocity2Label.pos.y > y1):
        gVelocity2Label.pos = vector(0, y1, 0)
    elif (gVelocity2Label.pos.y > y2):
        gVelocity2Label.pos = vector(0, y2, 0)
##displayOutputStr

def initialize(whichMotion):
    global gCurrentMotion
    global gBody1
    global gBody2
```

```
    global gT
    global gDt
    global gTheGraph
    global gCdotsV1
    global gCdotsV2

    gCurrentMotion = whichMotion
    gT = 0; gDt = 0.1  ## Time variables

    if gBody1 == undefined: ##vCreate body1
       gBody1 = sphere( pos = vector(b1P0),  radius = 2*b1Radius, color = b1Color,
                make_trail = True, trail_type = "points", trail_radius=b1Radius*.25)
       gBody1.m = b1Mass

    else: #Reset body1
        gBody1.make_trail = False
        gBody1.clear_trail()
        gBody1.pos = vector(b1P0)
        gBody1.make_trail = True

    if whichMotion == SymmetricMotion:
        gBody1.v = vector(b1V0)
    elif whichMotion == LoopLeftMotion:
        gBody1.v = vector(0.0, 0.0, 0.0)
    elif whichMotion == NearCircularOrbit:
        gBody1.v = vector(0.0, 0.0, 0.0)
    gBody1.a = 0

    if gBody2 == undefined: #Create body2
        gBody2 = sphere(pos = vector(b2P0), radius = 2*b2Radius, color = b2Color,
                    make_trail = True, trail_type = "points",  trail_radius=b2Radius*
    else: #Reset body2
        gBody2.make_trail = False
        gBody2.clear_trail()
        gBody2.pos = vector(b2P0)
        gBody2.make_trail = True

    if whichMotion == SymmetricMotion:
        # 'v02 = -v01; m1 = m2'
        gBody2.m = b1Mass
        gBody2.v = vector(b2V0)
    elif whichMotion == LoopLeftMotion:
        # "v01 = 0, v02 ≠ 0; m1 = m2"
        gBody2.m = b2Mass
        gBody2.v = vector(b2V0)
```

```
    elif whichMotion == NearCircularOrbit:
        # "v01 = 0, v02 ≠ 0; m1 >> m2"
        gBody2.v = vector(b2V0)
        gBody2.m = b2Mass/1e10
    gBody2.a = 0

    if gTheGraph ==undefined:  ##Create plots of the bodies speeds
        gTheGraph = graph( width = 500, height=200, xmin = 0, xmax = 100)
        gCdotsV1 = gcurve(color = b1Color,  dot = True, dot_color=color.black)
        gCdotsV1.plot([0,0])
        gCdotsV2 = gcurve(color = b2Color,   dot = True, dot_color=color.black)
        gCdotsV2.plot([0,0])
    else: ##Clear the plots of the bodies speeds
        gCdotsV1.data = [[0,0]]
        gCdotsV2.data = [[0,0]]



##
def update():
    global gT
    global gBody1
    global gBody2
    global gCdotsV1
    global gCdotsV2

    gT += gDt ## Increment time
    if gT > gMaxTime:
        return

    dirVector = gBody2.pos - gBody1.pos ## Direction Vector
    distSqr = dirVector.mag2       ## Distance between bodies Squared
    norm(dirVector)                ## Normalize the direction vector

    gBody1.a = gGravity*gBody2.m/distSqr*dirVector ## Acceleration of body1
    gBody1.v = gBody1.v + gBody1.a*gDt    ## Approximate velocity of body1
                                          ##  via Euler's Method
    gBody1.pos = gBody1.pos + gBody1.v*gDt    ## Approximate position of body1
                                              ## via Euler's Method
    s1 = gBody1.v.mag## Speed of body1
    if gT <= gMaxTime:
        gCdotsV1.plot([gT, s1])

    gBody2.a = -gGravity*gBody1.m/distSqr*dirVector ## Acceleration of body2
    gBody2.v = gBody2.v + gBody2.a*gDt              ## Approximate velocity of body2
```

```
                                            ## via Euler's Method
    gBody2.pos = gBody2.pos + gBody2.v*gDt ## Approximate position of body2
                                            ## via Euler's Method
    s2 = gBody2.v.mag                       ## Speed of body2
    if gT <= gMaxTime:
        gCdotsV2.plot([gT, s2])

    displayVelocity(gBody1.v, gBody1.pos, gBody2.v, gBody2.pos)


##update



initialize(SymmetricMotion)

##Create a statement of the model using LaTeX:
teXStr = "The Two-Body Model:"
teXStr += "<p>  "
teXStr += "\\(\\mathbf{d}=\\mathbf{p_{2}}-\\mathbf{p_{1}},\\)  "
teXStr += "\\(\\mathbf{a_{1}}=\\frac{G m_{2}}{||\\mathbf{d}||^{3}}\\mathbf{d},"
teXStr += "  \\mathbf{a}_{2}=-\\frac{G m_{1}}{||\\mathbf{d}||^{3}}"
teXStr += "\\mathbf{d}; \\)</p>"
teXStr += "<p>  \\(\\mathbf{v_{i_{new}}}\\approxeq"
teXStr += "\\mathbf{v_{i_{old}}}+\\mathbf{a_{i}}dt\\)"
teXStr += "    and   ";
teXStr += "\\(\\mathbf{p_{i_{new}}}\\approxeq"
teXStr += "\\mathbf{p_{i_{old}}}+\\mathbf{v_{i_{new}}}dt\\)"
teXStr += "   for \\(i = 1,2\\)</p>"
scene.caption = teXStr

MathJax.Hub.Queue(["Typeset",MathJax.Hub])

#An imation loop
while True:
    rate(gAnimationRate, wait)
    if gStarted and not gPaused:
        update()
```