Computers Come of Age

JOURNEYS IN FILM
educating for global understanding

JOURNEYS IN FILM™
educating for global understanding

Table of Contents

Click the Chapter heading to be taken to that page

About *Journeys in Film*

Founded in 2003, ***Journeys in Film*** operates on the belief that teaching with film has the power to prepare students to live and work more successfully in the 21st century as informed and globally competent citizens. Its core mission is to advance global understanding among youth through the combination of age-appropriate films from around the world, interdisciplinary classroom materials coordinated with the films, and teachers' professional-development offerings. This comprehensive curriculum model promotes widespread use of film as a window to the world to help students to mitigate existing attitudes of cultural bias, cultivate empathy, develop a richer understanding of global issues, and prepare for effective participation in an increasingly interdependent world. Our standards-based lesson plans support various learning styles, promote literacy, transport students around the globe, and foster learning that meets core academic objectives.

Selected films act as springboards for lesson plans in subjects ranging from math, science, language arts, and social studies to other topics that have become critical for students, including environmental sustainability, poverty and hunger, global health, diversity, and immigration. Prominent educators on our team consult with filmmakers and cultural specialists in the development of curriculum guides, each one dedicated to an in-depth exploration of the culture and issues depicted in a specific film. The guides merge effectively into teachers' existing lesson plans and mandated curricular requirements, providing teachers with an innovative way to fulfill their school districts' standards-based goals.

Why use this program?

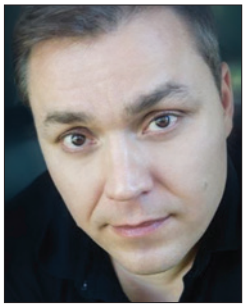
To be prepared to participate in tomorrow's global arena, students need to gain an understanding of the world beyond their own borders. ***Journeys in Film*** offers innovative and engaging tools to explore other cultures and social issues, beyond the often negative images seen in print, television, and film media.

For today's media-centric youth, film is an appropriate and effective teaching tool. ***Journeys in Film*** has carefully selected quality films that tell the stories of young people living in locations that may otherwise never be experienced by your students. Students travel through these characters and their stories: They drink tea with an Iranian family in *Children of Heaven,* play soccer in a Tibetan monastery in *The Cup*, find themselves in the conflict between urban grandson and rural grandmother in South Korea in *The Way Home,* watch the ways modernity challenges Maori traditions in New Zealand in *Whale Rider,* tour an African school with a Nobel Prize-winning teenager in *He Named Me Malala,* or experience the transformative power of music in *The Music of Strangers: Yo-Yo Ma & the Silk Road Ensemble.*

In addition to our ongoing development of teaching guides for culturally sensitive foreign films, ***Journeys in Film*** brings outstanding documentary films to the classroom. ***Journeys in Film*** has identified exceptional narrative and documentary films that teach about a broad range of social issues in real-life settings such as famine-stricken and war-torn Somalia, a maximum-security prison in Alabama, and a World War II concentration camp near Prague. ***Journeys in Film*** guides help teachers integrate these films into their classrooms, examining complex issues, encouraging students to be active rather than passive viewers, and maximizing the power of film to enhance critical thinking skills and to meet the Common Core Standards.

Journeys in Film is a 501(c)(3) nonprofit organization.

A Letter From Theodore Melfi



When you find a career you love, fame is far from your mind. Passion, excitement, and challenging work are instead the driving factors that motivate on a daily basis. Such is the case for Katherine G. Johnson, Dorothy Vaughan, and Mary Jackson—the brilliant trio of African-American women working at NASA in the early 1960s—who helped serve as the brains behind one of the greatest operations in history: the Mercury space missions, culminating in the launch of astronaut John Glenn into orbit.

For decades, until the publication of Margot Lee Shetterly's book *Hidden Figures*, the story of Johnson, Vaughan, and Jackson, NASA's *so-called "human computers,"* went untold. But when their story crossed my path—a story that blurs gender, race, and professional lines—I knew this was a part of history that had to be told. Fifty-five years later, *Hidden Figures* is a rich and moving true story that deserves a spot in our collective consciousness.

The backdrop for the movie is one of the most defining, complex periods in American history: the high-stakes Cold War, the space race, the Jim Crow South and the birth of the civil rights movement. Exploring these historic events serves as a reminder that we must learn from our past experiences while continuing to catapult ourselves forward.

It was also important for me, as a son raised by a single mother and as the father of two daughters, to explore the importance of STEM as a compelling and viable career choice for young girls. The media, cinema, and other public discourse often do society a disservice by not presenting strong, independent women in the fields of science, technology, engineering and math on a regular basis. Drawing attention to these figures, often hidden in plain sight, will hopefully help to chart a new course for female students and change the composition of these vital industries.

At its core, *Hidden Figures* is the story of three remarkable women who overcame every obstacle stacked against them, despite gender, race, and the political landscape of the time. Illuminating this universal experience for the next generation was critical. My goal was to showcase how skill and knowledge are equalizers, how hard work and determination are the cornerstones to every pursuit, and how uniting under a common goal is more powerful than staying divided.

Johnson, Vaughan, and Jackson were pioneers who broke down commonly held perceptions and achieved something phenomenal. Their legacy of persistence serves to empower people of all circumstances and teaches us, as NASA points out in its webpage on Katherine Johnson,

- To love learning.
- To follow your passion.
- To accept the help you're given, and help others when you can.
- To follow new leads and don't give up. Keep trying.
- To go beyond the task at hand; ask questions; be inquisitive. Let yourself be heard.
- To do what you love, and love what you do.

I hope that through the exploration of *Hidden Figures*—and your own passions—you, too, will achieve the seemingly impossible.

Theodore Melfi

Director, *Hidden Figures*

Introducing *Hidden Figures*

Space exploration in the modern age is entering a new phase, replete with private space companies, prospective lunar tourism, and even projected travel to Mars, the closest planet in our solar system. It is fitting, therefore, to pause to look back at the early years of the United States space program, and particularly the early efforts to launch astronauts into orbit, a preliminary step toward a moon landing.

Hidden Figures tells us about a generally unheralded group of women whose brilliance and dedication provided a foundation for the space program—the Black women known as "human computers" who worked at the NASA Center in Langley, Virginia. Faced with obstacles to their own education and to job prospects because of race and gender, these women succeeded in earning places and eventually respect in a workplace dominated by male supervisors and colleagues, many of whom were reluctant to hire women, and marked by segregated facilities, from office to restroom, that reflected the pre-civil rights era.

Katherine Johnson, physicist and mathematician, calculated the orbits, trajectories, and launch windows that would put John Glenn and others into space and bring them back safely. Dorothy Vaughan, another mathematician, became the first African-American supervisor at NASA, learning the computer language FORTRAN on her own and teaching it to her staff. Mary Jackson, an aerospace engineer as well as a mathematician, had to go to court to earn the right to take graduate-level courses at a previously all-white school; she eventually also served as a program officer helping other women succeed at NASA.

Their story is also the story of the world in which they lived and worked—the racism and segregation that made their lives more difficult; the beginnings of the civil rights movement in the South; the Cold War with Russia that gave such impetus to the drive for superiority in space; and the space race itself. The film weaves these events into the dramatic personal stories with skill and accuracy, making it an ideal film for the classroom. It is sure to serve as inspiration to many young women considering a career in science and mathematics.

Hidden Figures has been nominated for many awards, including the Academy Awards, BAFTA, the Golden Globes, the NAACP Image Awards, the Screen Actors Guild, and the African-American Film Critics Association.

Film credits

Director: Theodore Melfi

Screenplay: Allison Schroeder and Theodore Melfi, based on the book with the same title by Margot Lee Shetterly

Producers: Donna Gigliotti, Peter Chernin, Jenno Topping, Pharrell Williams, Theodore Melfi

Actors: Taraji P. Henson, Octavia Spencer, Janelle Monáe, Kirsten Dunst, Jim Parsons, Mahershala Ali, Aldis Hodge, Glen Powell, Kimberly Quinn, Kevin Costner, Olek Krupa

Computers Come of Age

Enduring Understandings

- A nonzero net force on an object causes a change in its momentum. (Newton's second law)
- Newton's laws apply at vastly different scales.
- Computer programs can simulate complex phenomena with relatively simple rules.
- Automation has long played a deciding role in job market shifts, and will probably continue to do so for the foreseeable future.

Essential Questions

- What quantitative effect do forces have on the motion of an object?
- How did early NASA workers, including Katherine Johnson and her fellow "human computers" depicted in *Hidden Figures*, use repetitive (iterative) techniques to model the paths of rockets?
- How has automation affected employment in the past, and how is it likely to affect it in the near future?

Notes to the Teacher

The programming problems in this lesson are designed to be done in the classroom under supervision, especially at the beginning. As the students get more comfortable with GlowScript, some programming problems may be done for homework. Calculation problems, on the other hand, are best assigned as homework problems. The students then present their own work and respond to the presentations of others. The role of the teacher is to guide the discussion and ask probing questions.

Before the lesson, work through all programming exercises that you will use in the classroom, even if you are familiar with Python or VPython. Much of class time will be spent helping student identify minor typos, misplaced parentheses, and the like. Having the code fresh in your mind will make this process easier. Students who finish early or who have prior experience with coding can also help identify bugs in the work of their peers.

Instead of having students watch the introductory videos independently in problem 2, you may wish to walk the students through each exercise in the videos in class on a projector or large screen. Then give them the challenge at the end of each video to do on their own, in class, or for homework. This gives you a chance to address the typical problems with typos, spaces, bad formatting, and so on that arise frequently with students new to programming.

The 30-minute video *Thinking Iteratively*, at **https://www.youtube.com/watch?v=e-shsRZQsi4**, presented by Bruce Sherwood and Ruth Chabay, gives a good overview of the reasoning behind using iterative programming in teaching physics. (This video is a useful tool for you, rather than for

your students.) Sherwood and Chabay's textbook, *Matter and Interactions*, uses VPython extensively in its presentation and exercises.

Any modern computer with Internet access can run GlowScript. If GlowScript does not run for some reason, try a different Web browser; Mozilla Firefox and Google Chrome usually work well. Tablets and even phones may be used, although the input interface is more awkward, especially for inserting tabs and copying and pasting code.

Students can work alone or in pairs. More than two students in a group rarely works, unless you have highly motivated students. An especially effective system is for students to work in pairs, but with each member of the pair entering code into his or her own device. They mainly write the same code, but each student in the pair is free to try different things on a whim. Each student also gets the benefit of the thrill that comes with finally making it work. It's impossible to overstate the importance of that. This is one of the best ways to learn to code.

Note that many problems require the use of scientific notation. If your students are not strong in this topic, you may wish to have them do the first part of Lesson 5. Scientific notation is nominally a middle school standard in most schools. However, discomfort with exponents, and in particular with scientific notation, is widespread at all ages. (Note that the concept of significant digits, though sometimes taught hand-in-hand with scientific notation, is not explicitly used in any of the problems in Lessons 5 or 6.) Additional practice exercises in scientific notation can be found in most high school chemistry or physics textbooks, as well as many places on the Internet.

While it may seem heresy to some, as a STEM educator you are probably already aware that Wikipedia is an excellent, reliable resource for quickly obtaining numerical data about planets and space missions. Some problems require students to obtain orbital data on their own. Don't discourage the use of Wikipedia for this. But this is also a great opportunity to have a debate about the difference between "primary source" and "one of your primary resources."

A good tool for enabling discussions in a large class is the portable whiteboard. The American Modeling Teachers Association (AMTA, at http://modelinginstruction.org/) has information on obtaining inexpensive whiteboards, for example the document *Whiteboards*, by Jane Jackson (**http://modeling.asu.edu/modeling/whiteboards2008.doc** and **http://legacy.modelinginstruction.org/wp-content/uploads/2013/05/whiteboards2008.doc**). AMTA has many resources on how to use whiteboards for effective classroom discussion, for example at **http://modeling.asu.edu/Projects-Resources.html**. Kelly O'Shea also has practical advice on using whiteboards in the classroom at **https://kellyoshea.blog/tag/whiteboards/**.

To introduce the problems at the end of the lesson on automation, you may wish to show the video *Tesla Self-Driving Demonstration* (**https://www.tesla.com/videos/autopilot-self-driving-hardware-neighborhood-long**) to your students. Although it is a promotional video, it provides a striking visual hook to get students interested in the topic. (Note that as time passes from the publication date of these videos and autonomous cars continue to become more common, it may be necessary to do a web search of a few supplementary articles and videos that speak to the current state of autonomous cars.) Another good introductory video is *Take A Ride In One Of The World's First Self Driving Trucks* (**https://www.youtube. com/watch?v=TWHgajaEMM8**) by VICE News.

A good resource for connecting the mathematics of conic sections with gravitational physics is *Six Ideas That Shaped Physics: Unit N — Laws of Physics Are Universal* by Thomas Moore.

On a (somewhat) reassuring note for the teacher using this lesson, the paper by Carl Frey and Michael Osborne—"The Future of Employment: How Susceptible Are Jobs to Computerisation?—calculates the probability of automating most teaching professions as less than 20 percent (and less than one percent, in some cases). It is available here: **http://www.oxfordmartin.ox.ac.uk/downloads/academic/The_Future_of_Employment.pdf**

A final alert: Modeling physics in GlowScript is fun. Many students who would normally be frustrated or otherwise unmotivated by school are suddenly captivated by the challenge of creating a working reality. The immediacy of the feedback is something usually found in sports or music. You do it wrong, it doesn't work. Over and over again. You finally do it right, and it works beautifully. People like that sort of thing.

Common Core Math Standards addressed by this lesson

Many concepts in this lesson fall under a number of different math standards. However, vectors in particular are central to understanding and creating many of the programs in this lesson:

CCSS.Math.Content.HSN.VM.A.1
Recognize vector quantities as having both magnitude and direction. Represent vector quantities by directed line segments, and use appropriate symbols for vectors and their magnitudes (e.g., *v*, *|v|*, *||v||*, *v*).

CCSS.Math.Content.HSN.VM.A.2
Find the components of a vector by subtracting the coordinates of an initial point from the coordinates of a terminal point.

CCSS.Math.Content.HSN.VM.A.3
Solve problems involving velocity and other quantities that can be represented by vectors.

CCSS.Math.Content.HSN.VM.B.4
Add and subtract vectors.

CCSS.Math.Content.HSN.VM.B.5
Multiply a vector by a scalar.

Next Generation Science Standards addressed by this lesson

HS-PS2-1.
Analyze data to support the claim that Newton's second law of motion describes the mathematical relationship among the net force on a macroscopic object, its mass, and its acceleration.

HS-PS2-4.
Use mathematical representations of Newton's Law of Gravitation and Coulomb's Law to describe and predict the gravitational and electrostatic forces between objects.

HS-ESS1-4.
Use mathematical or computational representations to predict the motion of orbiting objects in the solar system.

HS-ETS1-1.
Analyze a major global challenge to specify qualitative and quantitative criteria and constraints for solutions that account for societal needs and wants.

HS-ETS1-4.
Use a computer simulation to model the impact of proposed solutions to a complex real-world problem with numerous criteria and constraints on interactions within and between systems relevant to the problem.

Duration of the Lesson

Each major programming problem, if done in class, will probably use most of one 50- to 60-minute class period. If that seems long, remember that a large part of class time will be devoted to spotting typos and bugs. As students get more proficient this time can possibly be shortened by instituting code review sessions, a typical feature of professional software development firms, where one pair of students looks at the code of another pair for typos and bugs.

You may wish to assign calculation problems for homework, and then have the students discuss these in the next class. Programs that require only small modifications to programs written in class can also be assigned for homework. There are six major programs with code given in the problem, if you count the introduction videos as one. So if you work through all problems in the lesson, it will take a minimum of six class periods. However, discussing the calculation problems, even if done for homework, will take up additional class time, so you should figure on about eight class periods to cover all the problems.

That is far too long for most teachers. However, your students do not need to discuss every problem they do. In addition, you do not need to assign every problem. Be aware, however, that the programming problems build off previous programs. You are, of course, the best judge of your own students' needs and abilities. You may use all the problems or one of the problems, or anything in between. Or you may simply use them for reference when writing your own problems and lesson plans.

Assessment

If you structure your class so that students learn as they work through the exercises, you will be continuously checking on students' code, troubleshooting problems, and asking probing questions. Therefore, you will get continuous information about where your students are in their understanding.

Some of the longer problems and programs can also work as extended written assignments, rather than as topics for classroom debate. In particular, the analysis questions at the end on autonomous driving and workplace automation are intended to be in-depth research problems.

Materials

Handout 1: Using GlowScript to Solve Problems

Handout 2: Employment and Automation

Computer access

Procedure

1. Read through the problems on **Handout 1: Using GlowScript to Solve Problems**. After determining how much class time you have available to cover these topics, select the problems that are most relevant to your course goals, the length of your class period, and the age and ability of your students. There is ample opportunity for differentiation.

2. Assign a reasonable number of problems to be worked on for homework in preparation for class discussion. (See Notes to the Teacher.) Talk may veer off in any number of directions, as you guide the class to look for connections between ideas. This is, of course, an essential part of the process.

3. Copy the appropriate pages of the handout, marking the problems to be completed for homework. Alternatively, you can copy and paste the problems for each assignment onto a handout of your own.

4. In class each day, discuss each problem in turn, having students present their work and asking them questions about their reasoning. Give students time to ask questions until you are satisfied that they understand the basic concepts underlying the problems.

5. To conclude the lesson, ask students to think about the invention of the automobile and to list the jobs that were eliminated or greatly reduced when the automobile replaced horses as a primary means of transportation. (Horse breeders, blacksmiths, livery stables operators, farmers of hay and oats, carriage makers, stagecoach drivers, saddle makers.) Were any jobs created as a result of this invention? (Assembly line workers in automobile manufacturing plants, gas station operators, highway construction workers, auto mechanics, motel staff, fast food and convenience store workers.)

6. Distribute **Handout 2: Employment and Automation.** Have students watch suggested videos if desired (see Notes to the Teacher) and then work through the suggested readings and questions on the handout. Then conduct a class discussion on their conclusions.

Additional Resources

She's Coding (See their resource page for additional links)
http://shescoding.org/

Black Girls Code (#FutureKatherineJohnsons)
http://www.blackgirlscode.com/

Girls Who Code
https://girlswhocode.com/

Women Who Code
https://www.womenwhocode.com/

Girl Develop It
https://www.girldevelopit.com

Anita Borg Institute
https://anitaborg.org/

Raspberry Pi Coding
https://www.raspberrypi.org/education/

Handout 1 ▸ P. 1

Using GlowScript to Solve Problems

Directions:

In the exercises below, you will create 3D models on a computer of spacecraft trajectories, planet orbits, and other physical phenomena. You will need a device that connects to the Internet. A desktop or laptop computer is ideal. A tablet or mobile phone will be a little harder to use, but doable.

1. The application you will use is called GlowScript (glowscript.org). In order to create, edit, and save your own programs, you must have a Gmail account. If you do not have a Gmail account, go to the Gmail website (**https://www.google.com/Gmail**) and create one. Once you have an account (or if you have one already), log in to your account in a Web browser. In the same browser, go to the GlowScript website (**http://www.glowscript.org**).

You may need to click on the link labeled "Sign in." Then click on the link in "Your programs are here."

2. GlowScript is a Web-based implementation of VPython, which is itself a package for the Python programming language. One of the best resources to get started with VPython is a series of videos produced by Ruth Chabay and Bruce Sherwood (**https://www.youtube.com/user/VPythonVideos/videos**). A search for "VPython videos" should bring them up.

Watch videos 1 through 5, copying the illustrative, or sample, programs in GlowScript as you go. Do the exercises given at the end of each video. Note: Do not start each program with `from visual import *`. Instead, a GlowScript program must have `GlowScript 2.4 VPython` as its first line. This line is made automatically whenever you start a new GlowScript program.

3. While working through the five videos noted above, figure out how to change the viewpoint and zoom in and out on the GlowScript display. The method varies depending on the type of device and the operating system you are using.

4. "Position update." Imagine a lab cart moving along a low-friction track at a constant velocity. If we divide time into equal intervals, then the cart's position changes by the same amount during each interval. Suppose the cart moves with a constant velocity $\vec{v} = [0.63, 0, -0.16]$ m/s, starting at a position of $\vec{r}_0 = [3.4, 0, 1.0]$ m at $t = 0$ s. How much does the cart's position in the x,y, and z coordinates change during every time interval of $\Delta t = 0.5$ s? What will be the cart's position $\vec{r}$ at $t = 0.5$ s? At $t = 1.0$ s? At $t = 1.5$ s? At $t = 20.0$ s?

Handout 1 ▸ P.2

Using GlowScript to Solve Problems

5. The process you used in the previous problem to update the position $\vec{r}$ can be summarized in the equation $\vec{r}_f = \vec{r}_i + \vec{v}\Delta t$. Show how this equation can be rearranged to be a calculation of the velocity $\vec{v}$. Explain why the equation $\vec{r} = \vec{r}_i + \vec{v}t$ accurately calculates the position $\vec{r}$ of the cart in the previous problem for any time t. Are there conditions under which it would not work?

6. In the next problem you will run a program that models an object moving at constant velocity. All units will be in the International System, abbreviated SI. (In France, where the system was developed, it is called *Système international*, which is why the abbreviation is SI instead of IS.) What are the SI units for mass? For length? For time? For force?

7. The following code models the motion of a lab cart along a 2.2 m track. Create a new GlowScript program called "constant velocity," copy this code into the program, and run it.

```
GlowScript 2.4 VPython

track = box(pos=vec(0,0,0), length=2.2, height=0.02, width=0.1,

color=color.red)# create the track

car = box(pos=vec(-1.00,0.03,0), length=0.14, height=0.04, width=0.08,color=color.
  green) #create the car

v = vec(0.8,0,0)# create the initial velocity vector

dt = 0.01 # choose the time increment

while car.pos.x < 1.00: # loop as the car is at an x position less than 1 rate(1/dt)
  car.pos = car.pos + v*dt
```

a. Which line of the code updates the position? What is the difference between `car.pos` and `car.pos.x`?

b. The variable `dt` sets the increment of time for updating the position of the cart. Explain why the `rate()` statement has `1/dt` as its argument. Hint: you may need to look up what rate() does in Glowscript online.

c. Modify the code to cause the cart to start in the center of the track, move to the right at 0.8 m/s, then immediately start moving the left at half its initial speed when it reaches the end of the track. It should stop when it reaches the other end. (Hint: One way is to use two `while` loops.)

Handout 1 ▸ P.3

Using GlowScript to Solve Problems

8. The following code models the motion of the lab cart along a track, but this time there is a constant force applied to it. (One way to do this in the lab is to attach a small fan to the cart.) Create a new GlowScript program called "constant force," copy this code into the program, and run it.

```
GlowScript 2.4 VPython

track = box(pos=vec(0,0,0), length=2.2, height=0.02, width=0.1, color=color.red)

car = box(pos=vec(-1.00,0.03,0), length=0.14, height=0.04, width=0.08, color=color.
  green)

m = 1                          # mass of car, kg
v = vec(0.8,0,0)               # initial velocity of car, m/s
F = vec(-0.25,0,0)             # net force on the car, N
dt = 0.01
while car.pos.x < 1.00 and car.pos.x >= -1.00:
  rate(1/dt)
  a=F/m
  v = v + a*dt
  car.pos = car.pos + v*dt
```

a. Why does the cart slow down and turn around? Use a force/free body diagram for the cart in your explanation.

b. This code includes comments, which appear after # marks. Comments are ignored by GlowScript when running the program. So why would anyone use them? Comment the uncommented lines after the while statement.

Handout 1 ▸ P.4

Using GlowScript to Solve Problems

9. Create a new program called "rock drop" that models the fall of a 3 kg rock from a height of 12 m. Remember that the magnitude of the force of gravity near the Earth's surface is $F_{\text{grav}} = mg$, where $g \approx 9.8$ N/kg. You will need to manually put code in to make the rock stop when it hits the ground. Modeling a rock bouncing and tumbling when it hits the Earth is quite hard and beyond the scope of this exercise.

"Universal Gravitation." As you may have learned in your physics or science classes, all objects with mass attract each other. Larger masses exert a stronger gravitational force on each other, but larger distances between objects result in a smaller gravitational force. For two objects, labeled 1 and 2, the magnitudes of the force exerted by 1 on 2 and the force exerted by 2 on 1 are equal and given by

$$F_{12} = G\frac{m_1 m_2}{r^2}$$

The masses of the objects are m_1 and m_2, and r is the distance between them. The constant G is called the universal gravitational constant. Its value is $G \approx 6.67\times10^{-11}$ as measured in SI units.

10. Use the form of the equation for universal gravitation to determine the units for G.

11. Looking up any values necessary, use the equation for universal gravitation to calculate the size of the following gravitational forces:

a. The force of the sun on Jupiter

b. The force of the sun on an 80 kg human, who is standing on the surface of the Earth

c. The force of the Earth on an 80 kg human, who is standing on the surface of the Earth

d. The force of the same 80 kg human on the Earth

The gravitational force has a direction as well as a magnitude—it is a vector. Here is the expression for the force of the sun on the planet Jupiter:

$$\vec{F}_{\text{S on J}} = -G\frac{m_S m_J}{\left|\vec{r}_{SJ}\right|^2}\hat{r}_{SJ}$$

Handout 1 ▸ P. 5

Using GlowScript to Solve Problems

12. The ^ or "hat" over the last r_{SJ} means that it is a "unit vector." A unit vector is a vector that has a magnitude of 1. Unit vectors are useful in situations where direction is important, but magnitude is not. Identify which of the following are unit vectors:

a. $[0, -1, 0]$

b. $[1, 1, 1]$

c. $\left[-\frac{6}{7}, \frac{3}{7}, \frac{2}{7}\right]$

d. $\left[-\frac{1}{2}, 0, -\frac{\sqrt{3}}{2}\right]$

13. A unit vector can be created by dividing a vector by its magnitude. When this process is carried out, the arrow over the vector is replaced by the hat. For example, $\hat{v} = \frac{\vec{v}}{|\vec{v}|}$. The symbol $\hat{v}$ is pronounced "vee hat." Calculate the unit vector versions of each of the following vectors:

a. $\vec{r} = [3, 4, 0]$ m

b. $\vec{p} = [1, 2, 3]$ kg•m/s

14. What are the units of a unit vector?

15. The vector $\vec{r}_{SJ}$ is the vector pointing from the sun to Jupiter. Make a sketch of the sun and Jupiter, and show the following vectors: $\vec{F}_{S\text{ on }J}$, $\vec{F}_{J\text{ on }S}$, $\vec{r}_{SJ}$, and $\vec{r}_{JS}$. Which way does $\hat{r}_{SJ}$ point? Why is there a negative sign in the equation?

16. Create a new GlowScript program called "Sun-Jupiter," copy this code into the program, and run it. Verify that you see a yellow sphere representing the sun and a red sphere representing Jupiter.

```
GlowScript 2.4 VPython
scalefactor = 100
G = 6.67E-11      # universal
                  gravitational constant,
                  SI units
mS = 1.99E30      # mass of Sun, kg
mJ = 1.90E27      # mass of Jupiter, kg
RJ = 6.99E7       # radius of Jupiter, m
```

Handout 1 ▸ P.6

Using GlowScript to Solve Problems

```
RS = 6.96E8                    # radius of Sun, m
dSJ = 7.78E11                  # mean distance of Jupiter from Sun, m
T = 11.86* 365.25*24*60*60     # orbital period of Jupiter
speed = 2*pi*dSJ/T
Sun = sphere(pos=vec(0,0,0), radius=RS*scalefactor, color=color.yellow)
Jupiter = sphere(pos=vec(dSJ,0,0), radius=RJ*scalefactor, color=color.red,
  make_trail=True)
```

a. Change the value of `scalefactor` to 50, and run the program again. Then change it to 10, and run the program. What value of `scalefactor` displays a representation to scale of the sun and Jupiter? What does the program display when you use this value?

b. The orbital period of Jupiter is 11.86 years. This is the time is take Jupiter to orbit the sun once. Explain the other factors in the calculation of `T` in the program.

c. Describe the speed that the line `speed = 2*pi*dSJ/T` is calculating. Why does it work?

Add the following code to your Sun-Jupiter program.

```
v = vec(0,speed,0)
dt = 0.1*24*60*60
while True:
rate(5E7/dt)
  r = Jupiter.pos-Sun.pos
  rhat = r/mag(r)
  F = -G*mS*mJ/mag(r)**2 * rhat
  a = F/mJ
  v = v+a*dt
  Jupiter.pos = Jupiter.pos + v*dt
```

d. How many days is each time interval in this program? Why is it OK for it to be so long?

e. What is the purpose of the `5E7` in the `rate()` function?

f. You can double the mass of the sun by changing its value to `mS = 2*1.99E30`. What is the effect of doing this? What about cutting the sun's mass in half? Does the same thing happen if you change the mass of Jupiter instead? Investigate the effect of changing other parameters, such as `G`, `dSJ`, and `speed`.

Handout 1 ▸ P.7

Using GlowScript to Solve Problems

17. (Note: This problem requires some familiarity with the mathematical properties of ellipses.) Johannes Kepler published work in the early 17th century that summarized three observational facts about the motion of the planets. The first of these facts is that the orbit of a planet traces out an ellipse, where the sun is located at one of the foci of the ellipse. At the time, he did not provide any reasonable hypotheses about why this might be the case. However, about 50 years later, Isaac Newton showed that the fact that planetary orbits are ellipses follows from the law of universal gravitation.

Although this is a fairly straightforward derivation for a circular orbit, for an elliptical orbit the calculation is mathematically complex. However, you can check Kepler's prediction for a specific orbit that you've generated in VPython. By varying one of the parameters in part (f) of the previous problem, you almost certainly generated an orbit that resembles an ellipse. Take a screenshot of a complete orbit, and paste the image into an image program, and make it full screen. Using a ruler placed across the screen, measure the perihelion and aphelion for the orbit. The sun is at one focus of the (purported) ellipse. Using symmetry, you should be able to locate the other focus. Place the mouse cursor at the other focus. Pick a random point on the ellipse, and measure the distance of that point from each focus . If the orbit is indeed an ellipse, how do you expect the sum of these two distances to be related to the perihelion and aphelion? Do this for several other points. Is the orbit an ellipse?

18. Find a way to do the calculations from the previous problem, but in the code of the program itself. This should give more concrete evidence that the orbit is, in fact, an ellipse.

19. Create a new GlowScript program called "Freedom7test," copy this code into the program, and run it. This code represents a short (and quite unsafe) "test flight" of the Freedom7 capsule near the Earth's surface (so we can again replace Newton's inverse square law for the gravitational force with the approximation F_g=-mg . As it is now, the capsule misses the "target."

```
GlowScript 2.4 VPython
g = 9.8
m = 1400
Freedom7 = cone(pos=vector(0,0,0), axis=vector(0,2,0), radius=1, color=color.cyan,
  make_trail=True)
ground = box(pos=vector(0,0,0), length=100, height=1, width=100,
  color=color.yellow)
pad = cylinder(pos=vector(0,-0.6,0), axis=vector(0,1.2,0), radius=2,
  color=color.red)
target = cylinder(pos=vector(-40,-0.6,0), axis=vector(0,1.2,0), radius=2,
  color=color.green)
```

Handout 1 ▸ P.8

Using GlowScript to Solve Problems

```
dt = 0.01
F = vector(0,-m*g,0)
v = vector(10,16,10)

while Freedom7.pos.y >= 0:
  rate(1/dt)
  a=F/m
  v = v + a*dt
  Freedom7.pos = Freedom7.pos + v*dt
```

In your physics class, you may have learned how to find algebraic solutions for situations involving Newton's second law. This usually means that you solve for the motion of an object explicitly: You find a single equation that gives the quantity of interest in terms of the parameters of the problem. Find an algebraic solution for the initial velocity of the Freedom 7 capsule in terms of the position of the target. You may need to make some simplifying assumptions. In addition, there may be more than one possible answer. Once you have obtained a satisfactory solution, calculate a value for the initial velocity, and test it in the GlowScript program above.

20. The previous problem is a highly simplified version of the problem that Katherine Johnson and the other NASA workers were trying to solve in *Hidden Figures*. The next problem explores a more realistic, but still simplified version of the trajectory problem. Create a new GlowScript program called "Freedom7flight," copy this code into the program, and run it. (You do not need to copy all the comments.)

```
GlowScript 2.4 VPython
scalefactor = 1E4
G = 6.67E-11                  # universal gravitational constant, SI units
mE = 5.972E24                 # mass of Earth, kg
mc = 1400                     # mass of Freedom 7 capsule, kg
RE = 6.371E6                  # radius of Earth, m

# These two lines set up a "zoomed in" camera viewpoint. If you're curious
# about what they mean, read the GlowScript help menu under "Canvases."
scene.range=RE/10
scene.camera.pos = vec(0,RE,RE/10)
```

Handout 1 ▸ P.9

Using GlowScript to Solve Problems

```
# The next several lines set up the target area of the ocean. The target is shown
# to scale — approximately 20 square miles. Note that the target appears to
# float above the surface of the Earth. This is because GlowScript spheres are
# not perfectly smooth.
targetrange = 487.3E3          # The Freedom 7 capsule traveled 487.3 km
                               # across the Earth's surface.
targetangle = targetrange/RE   # angular distance traveled, radians

targetpos = RE*vec(cos(pi/2-targetangle),cos(targetangle),0)
targetaxis = 4E3*vec(cos(pi/2-targetangle),cos(targetangle),0)
target = cylinder(pos=targetpos, axis=targetaxis, radius=4E3,color=color.yellow)
Earth = sphere(pos=vec(0,0,0), radius=RE, color=color.cyan)
Freedom7 = cone(pos=vector(0,RE,0), axis=vector(0,2,0)*scalefactor,
  radius=1*scalefactor, color=color.green, make_trail=True)
```

Write a program that uses physics to model the flight of the Freedom 7. Since the capsule moves over a significant portion of the Earth, you must use the universal law of gravitation, as you did with modeling the orbit of Jupiter, rather than the near Earth approximation in the previous problem. The capsule must travel approximately 487 km to the target area, and reach an "apogee" (maximum height above the Earth) of 187.5 km.

As before, this program makes some simplifying assumptions. Only the capsule is launched, not the lower stage. Also, you can assume the rocket reaches its maximum speed right away—very unhealthy for the astronaut! In addition, it ignores the rotation of the Earth itself. (Katherine Johnson definitely could *not* afford to ignore the Earth's motion in her calculations. It would have led to a significant error in the result, and Alan Shepard would likely have been lost at sea.)

21. (Challenging) Write a new GlowScript program, called "Glennflight," that models astronaut John Glenn's orbital flight. As mentioned in *Hidden Figures*, this involves transitioning the spacecraft from roughly parabolic motion, as in Shepard's flight, to circular motion. Then to return to Earth, the opposite procedure must be carried out. In order to do this effectively, you must model the force of the rocket engines on Glenn's capsule, Friendship 7, during the launch. You will have to choose an appropriate size of the force exerted by the engines. You must also apply it in an appropriate direction (which may change during the flight) and for a sufficient period of time.

Handout 1 ▸ P. 10

Using GlowScript to Solve Problems

22. In general, Katherine Johnson used "iterative methods" for solving equations with derivatives in them (called "differential equations") numerically. This was an important step forward in predicting the trajectories of spacecraft. Research the difference between the two methods. The programs above that simulate trajectories use iterative methods. Identify in the code the iterative part. Why are iterative methods especially suited to solutions carried out on a machine computer?

23. Up until this point, you have been learning how to simulate physical reality by writing code yourself and running it immediately. However, when machine computers (as opposed to "human computers") were first invented, programmers had to write their code beforehand, carefully and diligently check and recheck it for errors, and then finally feed it into the machine. Results could take minutes, hours, or even longer. Recall that Dorothy Vaughan learned the programming language FORTRAN from a textbook, which she went to great lengths to obtain. How did she teach her coworkers the language? These women did not have much of a choice, but these days there are many ways to learn to program a computer. What are some possible advantages to learning programming from reading a textbook or attending a lecture? Are there any advantages to "learning by doing," as you have just done? Are there situations where one is preferable to the other?

24. (Challenging) A collection of at least 22 papers that Katherine Johnson wrote or co-wrote can be found on the NASA Technical Reports Server at **https://ntrs.nasa.gov/search.jsp?N=4293878840** and at **https://ntrs.nasa.gov/search.jsp?N=4293471801**. Pick one of her papers that is specifically on spaceflight and read it. It will probably be difficult, but focus on the parts you do understand. Look for things that are similar to the analyses you did in GlowScript, such as iteration, gravitational calculations, or use of vectors. State in your own words the problem the paper was addressing. Was it successful in solving that problem? How did Katherine Johnson (and her co-authors, if applicable) limit the scope of the problem to make it easier to analyze? In other words, what approximations did they make? Did they use a mechanical computer in their analysis?

Handout 2 ▸ P.1

Employment and Automation

Analysis 1: Autonomous Driving and Truck Drivers

Somewhere down the line, a human being is going to have to push the buttons.

—Dorothy Vaughan in *Hidden Figures*

Trucking is commonly cited as an industry that will soon be done mostly by automated driving systems, replacing a large number of humans currently employed as drivers. (See, for example, "Self-driving trucks: what's the future for America's 3.5 million truckers?" at **https://www.theguardian.com/technology/2016/jun/17/self-driving-trucks-impact-on-drivers-jobs-us**.)

Because it is always wise to check statistics, it would be reasonable to look at just how many people actually are employed as truck drivers. Read these two articles, starting with the NPR piece.

"Map: The Most Common Job In Every State"

http://www.npr.org/sections/money/2015/02/05/382664837/map-the-most-common-job-in-every-state

"No, 'truck driver' isn't the most common job in your state"
http://www.marketwatch.com/story/no-truck-driver-isnt-the-most-common-job-in-your-state-2015-02-12

What does Mr. Nutting, the author of the second article, mean by the statement, "It's one of those examples where a statistic can be absolutely correct and utterly wrong at the same time." How can something be both correct and wrong at the same time?

Explain how the two articles arrived at different conclusions from the same data. In order to get an unbiased view, it is important to examine the data itself directly from the United States Department of Labor, Bureau of Labor Statistics, at **https://www.bls.gov/oes/current/area_emp_chart/area_emp_chart.htm**. (Click on "View Chart Data" for the full dataset.) In your analysis, you may wish to load the raw data into a spreadsheet and calculate the statistics claimed in each article yourself. The raw data can be found here at **https://www.bls.gov/oes/current/oes_nat.htm**.

Handout 2 ▸ p.2

Employment and Automation

Analysis 2

Take computers. In the early 20th century a "computer" was a worker, or a room of workers, doing mathematical calculations by hand, often with the end point of one person's work the starting point for the next. The development of mechanical and electronic computing rendered these arrangements obsolete. But in time it greatly increased the productivity of those who used the new computers in their work.

—Quoted from the *Economist* article cited below

Regardless of the actual number of truck drivers in the world, future advances in automation will almost certainly not be limited to self-driving vehicles. The following article examines how technological changes have affected employment in the past, and discusses whether the lessons of the past are truly applicable to the future.

"The future of jobs: The onrushing wave," at **http://www.economist.com/news/briefing/21594264-previous-technological-innovation-has-always-delivered-more-long-run-employment-not-less**.

1. Which jobs are likely to be automated in the future? Describe some of the characteristics that separate jobs that are easier to automate from those that are much more difficult for machines to do. (You may find it useful to consult the article listed in question 2 below, skipping the more mathematical bits.) More disconcertingly, if you had a certain career in mind for yourself already, how do you view those plans after reading this article?

2. The article references a study done in 2013 by Carl Frey and Michael Osborne, "*The Future of Employment: How susceptible are jobs to computerisation?* (**http://www.oxfordmartin.ox.ac.uk/publications/view/1314**). Unfortunately, to critically examine the methods used in the paper requires a working knowledge of advanced probability theory. However, it will probably be helpful for your analysis to consult the appendix of this study, which is readily understandable. It lists the probability of automation that the authors calculated for each of the occupations listed by the Bureau of Labor Statistics. Since the authors use the same numerical code as the Bureau, you can fairly easily compare employment numbers in a profession with its probability of automation (as claimed by the authors of the study).

   Choose a few specific careers, preferably with a range of automation probabilities. Estimate the number of people likely to be affected by automation in the careers you examine. What are some possible societal impacts of such a shift? Will this be like previous technological advances, in which new, different jobs were created to replace the old? Or will this shift be toward significantly different work? If automation does lead to large-scale unemployment, what are some ways society might reasonably cope with it?

Teacher Resource 1

Using GlowScript to Solve Problems – Answer Sheet

1. For in-class activities, it is recommended that you work along with the students on a projector screen, if available, especially for the earlier problems. Keep in mind that your personal Gmail address will be visible while doing this. If this is not something you want to happen, make a throwaway Gmail account just for the class.

2. Instead of showing the videos, it can be useful to demonstrate similar examples yourself, and then assign the challenges as in-class problems immediately after each example.

3. On a PC, holding down the right mouse button while moving the mouse rotates the view. Hold down the left and right button simultaneously while moving the mouse to zoom in or out. The scroll wheel should also control zooming. On a phone or tablet, one and two finger motions should control panning and zooming. Challenge your students to figure out how to pan and zoom—they'll figure it out fairly quickly.

4. For an object moving at constant velocity $\vec{v}$, the position is given by $\vec{r} = \vec{r}_0 + \vec{v}\Delta t$. Or each step can be obtained iteratively, using the equation $\vec{r}_f = \vec{r}_i + \vec{v}\Delta t$ repeatedly.
For $\Delta t = 0.5$ s, $\Delta\vec{r} = \vec{r} - \vec{r}_0 = [0.315, 0, 0.08]$ m.
At $t = 0.5$ s, $\vec{r} = [3.715, 0, 0.92]$ m.
At $t = 1.0$ s, $\vec{r} = [4.03, 0, 0.84]$ m.
At $t = 1.5$ s, $\vec{r} = [4.345, 0, 0.76]$ m.
At $t = 20.0$ s, $\vec{r} = [16, 0, -2.2]$ m.

5. $\vec{v} = \Delta\vec{r}/\Delta t$
$\vec{r} = \vec{r}_i + \vec{v}t$ works when the start time is $t_i = 0$ s.

6. This takes a single Google search to look up. If you ask your class to tell you, you should get the correct answer —kilograms, meters, seconds, newtons—fairly quickly.

7a. `car.pos = car.pos + v*dt` updates the position. `car.pos` is the full position vector `car.pos.x` is the x coordinate of the position vector.

7b. `dt` is the time for each iteration, in s/step. The `rate()` method accepts an argument formatted as iterations per time increment, i.e., units of steps/s.

Teacher Resource 1

Using GlowScript to Solve Problems – Answer Sheet

7c. Here is one possible program:

```
GlowScript 2.4 VPython
track = box(pos=vec(0,0,0), length=2.2, height=0.02, width=0.1,color=color.red)
car = box(pos=vec(0,0.03,0), length=0.14, height=0.04, width=0.08, color=color.
  green)
v = vec(0.8,0,0)
dt = 0.01
while car.pos.x < 1.00:
  rate(1/dt)
  car.pos = car.pos + v*dt
v = -v/2
while car.pos.x > -1.00:
  rate(1/dt)
  car.pos = car.pos + v*dt
```

8a. A free body diagram for the cart should show not only the force from the fan, but also the gravitational force and normal force. Since the net force points in the negative x direction, Δp_x at each step is negative.

8b. Comments clarify the purpose of your code, not just for others, but also for you.

9. Here is one working program:

```
GlowScript 2.4 VPython
rock = sphere(pos=vec(0,12,0), radius=0.2, color=color.red)
ground = box(pos=vec(0,0,0), length=5, height=0.1, width=5, color=color.green)
g = 9.8             # gravitational field strength, N/kg
m = 3               # mass of rock, kg
v = vec(0,0,0)      # initial velocity of rock, m/s
p = m*v             # initial momentum of rock, kg*m/s
F = vec(0,-m*g,0)   # net force on the rockar, N
dt = 0.01
while rock.pos.y>0:
  rate(1/dt)
  p = p + F*dt
  v = p/m
  rock.pos = rock.pos + v*dt
```

Of course, it's possible to get more accurate with the final position, but the main focus should be on getting the physics right.

Teacher Resource 1

Using GlowScript to Solve Problems – Answer Sheet

10. The units for G are $\mathrm{N} \cdot \mathrm{m}^2/\mathrm{kg}^2$.

11a. 4.2×10^{23} N

11b. 0.49 N

11c. 784 N

11d. 784 N (by Newton's third law)

12. a, c, and d are unit vectors. Choice (b) may seem attractive to some students because each component is one, but the magnitude of this vector is not 1.

13a. $\hat{r} = \left[\frac{3}{5}, \frac{4}{5}, 0\right]$

13b. $\hat{p} = \left[\frac{1}{\sqrt{14}}, \frac{2}{\sqrt{14}}, \frac{3}{\sqrt{14}}\right]$

14. Unit vectors have no units: the units cancel when a vector is divided by its magnitude, since a vector has the same units as its magnitude.

15. The unit vector $\hat{r}_{SJ}$ points from the sun to Jupiter. The negative sign reverses the direction of $\hat{r}_{SJ}$, ensuring that the gravitational force on Jupiter points from Jupiter to the sun.

16a. The scale factor increases the size of the sun and Jupiter, without changing their relative distances. A scale factor of 1 gives a model of the two objects to scale, but renders both too small to see on most screens, without zooming in. This gives an idea of just how large the solar system is.

16b. This calculation changes the units of the orbital period from years to seconds. You might want to discuss why this is more readable than simply calculating the period in seconds beforehand and typing in that value directly.

16c. This calculates the orbital speed of Jupiter in m/s, but dividing the circumference of the circle it traces out by the orbital period. This assumes that the orbit is a circle, which of course is not quite true.

16d. The time interval is 0.1 day. Since this is a tiny fraction of the orbital period, the simulation is still quite accurate.

16e. This represents that the simulation is sped up by a factor of 5×10^7. Formatting the `rate()` method argument in this way—as a multiple of `1/dt` —just makes explicit the factor by which time is sped up.

16f. Students should be able to obtain both elliptical and hyperbolic orbits by changing the mass of the sun or the initial speed of Jupiter. Point out that they can do this quickly by (for example) changing `mS = 1.99E30` to `mS = 1.99E30*2`. No need to make a separate calculation on a calculator.

Teacher Resource 1

Using GlowScript to Solve Problems – Answer Sheet

Students may be surprised at first that changing the mass of Jupiter has no effect on its orbit, until they realize that increasing its mass leaves the acceleration unchanged, thus leaving the velocity change for each time increment the same. Point out that sending a satellite to another planet would be rather difficult if orbital speed depended on mass. (Note also that the sun's change in motion is neglected in this model.)

If your class has studied centripetal acceleration, then students should be able to calculate combinations of initial speed and position that would result in a circular orbit, for a given mass of the sun. They can then verify this in the GlowScript model.

17. It is recommended that you create printouts beforehand for students to use. You want to choose orbits that lead to visible ellipses. In addition, you should put some sort of marker at the other focus of the ellipse. See Lesson 5 for details about Kepler's first law. Important: If you choose to do this problem on paper, add the command `scene.background = color.white` to the first line of the program. This will set the background color to white, thus saving you a lot of toner.

18. Here is an example of a program that checks that the orbit of the dwarf planet Eris is an ellipse. Of course, any planet, real or fictional, will do.

```
GlowScript 2.4 VPython
scalefactor = 5E4
G = 6.67E-11                    # universal gravitational constant, SI units
mS = 1.99E30                    # mass of Sun, kg
mE = 1.66E22                    # mass of Eris, kg
RE = 1163E3                     # radius of Eris, m
RS = 6.96E8                     # radius of Sun, m
P = 5.723E12                    # perihelion of Eris from Sun, m
T = 558.04*365.25*24*60*60      # orbital period of Eris, s
speed = 2*pi* P/T*2.827         # initial speed to mimic orbit, m/s
A = P                           # initialize the aphelion to the perihelion
RealAphelion = 14.602E12

Sun = sphere(pos=vec(0,0,0), radius=RS*scalefactor/500, color=color.yellow)
Eris = sphere(pos=vec(P,0,0), radius=RE*scalefactor, color=color.red, make_
  trail=True)
otherfocus = sphere(pos=vec(P-RealAphelion,0,0), radius=RS*scalefactor/500,
  color=color.cyan)

v = vec(0,speed,0)
p = mE*v
dt = 0.5*24*60*60
t = 0
```

Teacher Resource 1

Using GlowScript to Solve Problems – Answer Sheet

```
checktime = T/10
checkcounter = 0

while t<T:
  rate(5E9/dt)
  r = Eris.pos-Sun.pos
  rhat = r/mag(r)
  F = -G*mS*mE/mag(r)**2 * rhat
  p = p + F*dt
  v = p/mE
  Eris.pos = Eris.pos + v*dt
  if mag(Eris.pos) > A:
    A = mag(Eris.pos)          # The aphelion is the maximum distance from the Sun.
if t/checktime > checkcounter:
    d1 = mag(Eris.pos-Sun.pos)
    d2 = mag(Eris.pos-otherfocus.pos)
    print("d1+d2 = ", d1+d2)
    checkcounter = checkcounter + 1
t = t + dt

print("elapsed time in years: ", t/(60*60*24*365.25))
print("calculated aphelion in meters: ", A)
```

19. The problem is intentionally under-specified, in the sense that students need to come up with the value of two parameters to hit the target. For example, with launch speed v and launch angle θ relative to the positive x axis (in the xy plane), the initial velocity is $\vec{v} = [-v\cos\theta, v\sin\theta, 0]$. By standard methods of solving kinematics equations, one obtains

$$v = \sqrt{\frac{-gx_f}{2\sin\theta\cos\theta}}$$

With (say) an angle of 55 degrees, the launch speed would need to be 20.42 m/s. Note that angles in GlowScript must be in radians. Students may use the `radians()` method to take care of this.

Teacher Resource 1

Using GlowScript to Solve Problems – Answer Sheet

20. Here is one example of a working program:

```
GlowScript 2.4 VPython
scalefactor = 1E4
G = 6.67E-11                    # universal gravitational constant, SI units
mE = 5.972E24                   # mass of Earth, kg
mc = 1400                       # mass of Freedom 7 capsule, kg
RE = 6.371E6                    # radius of Earth, m

# These two lines set up a "zoomed in" camera viewpoint. If you're curious
# about what they mean, read the GlowScript help menu under "Canvases".
scene.range=RE/10
scene.camera.pos = vec(0,RE,RE/10)

# The next several lines set up the target area of the ocean. The target is shown
# to scale - approximately 20 square miles. Note that the target appears to
# float above the surface of the Earth. This is because GlowScript spheres are
# not perfectly smooth.
targetrange = 487.3E3           # The Freedom 7 capsule traveled 487.3 km
                                # across the Earth's surface.
targetangle = targetrange/RE    # angular distance traveled, radians
targetpos = RE*vec(cos(pi/2-targetangle),cos(targetangle),0)
targetaxis = 4E3*vec(cos(pi/2-targetangle),cos(targetangle),0)
target = cylinder(pos=targetpos, axis=targetaxis, radius=4E3, color=color.yellow)

Earth = sphere(pos=vec(0,0,0), radius=RE, color=color.cyan)
Freedom7 = cone(pos=vec(0,RE,0), axis=vec(0,2,0)*scalefactor,radius=1*scalefactor,
  color=color.green, make_trail=True)

r = Freedom7.pos - Earth.pos
v = vec(1250,1868,0)
p = mc*v
```

Teacher Resource 1

Using GlowScript to Solve Problems – Answer Sheet

```
maxheight = 0
dt = 0.1

while mag(r) >= RE:
  rate(100/dt)
  F = -G*mc*mE/mag(r)**2 * norm(r)
  p = p + F*dt
  v = p/mc
  Freedom7.pos = Freedom7.pos + v*dt
  r = Freedom7.pos - Earth.pos
  if mag(r)-RE > maxheight:
    maxheight = mag(r)-RE

print(maxheight/1E3)          # print maximum height of rocket, km
```

Once the students have a working program, it is quite acceptable for them to find suitable initial conditions by trial and error. During the process of experimentation, it is likely that their initial guesses will either fall short or escape from the Earth altogether. By refining their guesses, they can get a feel for the gradual transition from parabolic flight in a constant gravitational field, to elliptical flight with a gravitational field that changes direction and magnitude.

Another key insight is that launching a spacecraft requires a large amount of sideways velocity, not just vertical velocity.

A hint for speeding up guesses: It helps to find an initial y component of velocity that gives the correct apogee of approximately 187.5 km. They can then refine the x component to make the capsule hit the target. The y component will need only small adjustments after that.

21. This is a challenging problem, and should be assigned only as a long-term project for highly motivated students. Students will need to change both the magnitude and direction of thrust over the course of the flight, in several distinct stages. Although the problem does not ask for it, another significant variable to model can be the changing mass of the spacecraft, as it sheds fuel, and also as the booster engines are dropped. If mass change is modeled, this problem can work well in conjunction with teaching the rocket equation (https://spaceflightsystems.grc.nasa.gov/education/rocket/rktpow.html).

22. Iterative methods are typically simple to describe, but require many calculations. They are suitable for machine computers, which are of course designed to quickly carry out many repetitive calculations. On the other hand, a closed form solution can be difficult or impossible (currently) for a machine to derive, but once obtained, may not require much calculation for any particular case. However, closed form solutions exist only for certain simple situations.

Teacher Resource 1

Using GlowScript to Solve Problems – Answer Sheet

23. This question is meant for classroom discussion after all the programming assignments have been done. Since the students have just completed an intensive "learn by doing" set of problems, they will likely have strong opinions on the matter.

24. This is a challenging problem for a motivated student. These papers emphasize that Katherine Johnson had a long (33-year) career with NASA, and that she did far more than solve orbital trajectory problems. Another possible way to go with this assignment is to have students write a broad survey on the topics Ms. Johnson studied and reported on while at NACA/NASA.

Journeys in Film
PO Box 65357
Albuquerque, NM 87193
www.journeysinfilm.org