Program Your Own Game

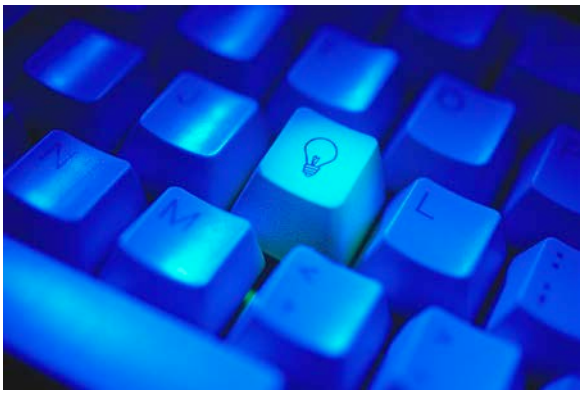
Provided by TryEngineering - www.tryengineering.org

Lesson Focus

Lesson focuses on how software engineers design computer games and other software. Student teams work together to develop a simple computer program using free software that is available in multiple languages. Teams evaluate the games developed by other teams and present findings to the class.

Lesson Synopsis

The Program Your Own Game activity explores the work of software engineers and allows student teams to develop their own computer game using free and simple software. Teams present their game to their class, evaluate other games, and reflect on the engineering experience.

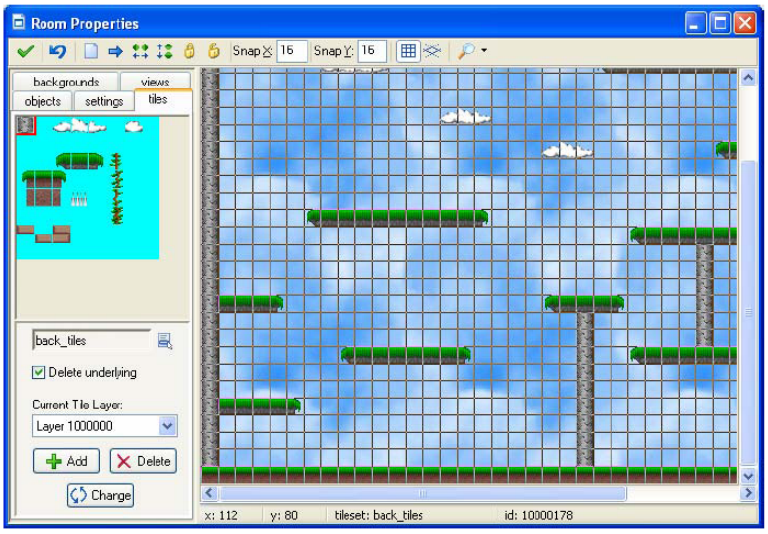


Age Levels

11-18 (Note: this lesson can range from very simple programming or program editing for younger students to more advanced programming for older or advanced students.)

Objectives

- Learn how software engineers develop computer games.
- Learn about the process of product re-engineering.
- Learn how engineering teams address problem solving.
- Learn about teamwork and working in groups.



Anticipated Learner Outcomes

As a result of this activity, students should develop an understanding of:

- software engineering and programming
- product design and engineering
- problem solving
- teamwork

Lesson Activities

Students learn about basic computer programming and the work of software engineers. Student teams work together to develop a simple computer program using free software that is available in multiple languages. Students execute their own games, and evaluate the games developed by other student teams.

Resources / Materials

- Teacher Resource Documents (attached)
- Game Maker Tutorials (www.yoyogames.com/make/tutorials)
- Student Worksheets (attached)
- Student Resource Sheets (attached)

Alignment to Curriculum Frameworks

See attached curriculum alignment sheet.

Internet Connections

- TryEngineering (www.tryengineering.org)
- YoYo Game - Game Maker Software (http://www.yoyogames.com/gamemaker/studio/)
- ITEA Standards for Technological Literacy: Content for the Study of Technology (www.iteaconnect.org/TAA)
- National Science Education Standards (www.nsta.org/publications/nses.aspx)
- NCTM Principles and Standards for School Mathematics (http://standards.nctm.org)

Recommended Reading

- Game Creation For Teens (ISBN: 159863500X)
- Getting Started with Game Maker (ISBN: 1598638823)

Optional Writing Activity

- Write an essay or a paragraph describing the ethical implications of adapting someone else's software programming. The concept of "intellectual property" is an umbrella term for legal entitlements which attach to certain names, written and recorded media, and inventions. Take a stand for or against whether you should provide financial or other credit to the original developer of software that you adapt/change to make new software. Consider that the original software did not sell well, but that your edited version sold very well. As an extension idea, this writing activity could turn into a lively debate of the pros and cons and the concept of intellectual property rights.

Program Your Own Game

For Teachers: Teacher Resources

◆ Lesson Goal

Explore engineering problem solving by working in teams to program a new computer game. Students learn about basic computer programming and the work of software engineers. Student teams work together to develop a simple computer program using free software that is available in multiple languages. Students execute their own games, and evaluate the games developed by other student teams.

◆ Lesson Objectives

- Learn how software engineers develop computer games.
- Learn about the process of product re-engineering.
- Learn how engineering teams address problem solving.
- Learn about teamwork and working in groups.

◆ Materials

- Student Resource Sheets and Worksheet
- Internet or computer access (free software may be downloaded and installed on windows-based computer without internet access; software is available in multiple languages)

◆ Procedure

1. Download and install the free YoYo Games GameMaker software (http://www.yoyogames.com/game_showcases/273/legacy_download) on multiple computers or in a lab so students can work in teams to develop their games. Note that there is a free limited version and a higher level one for a low fee. The free one will work well in the classroom environment for beginners.
2. You may also wish to view the tutorials at www.yoyogames.com/make/tutorials.
3. Show students the various Student Reference Sheets. These may be read in class, or provided as reading material for the prior night's homework. The pages on "beginning programming" should be reviewed by students in advance of using the software on a computer.
4. Divide students into teams of 2-3 students (you may need to adjust this depending upon how many computer stations you have); provide a set of materials per group.
5. Explain that they are a team of software engineers and they need to develop a new computer game that will be used by students who are aged 6-10.
6. Student teams develop a simple game and share it with the class.
7. Each student group evaluates the games developed by other teams, and completes an evaluation/reflection worksheet.

◆ Tips

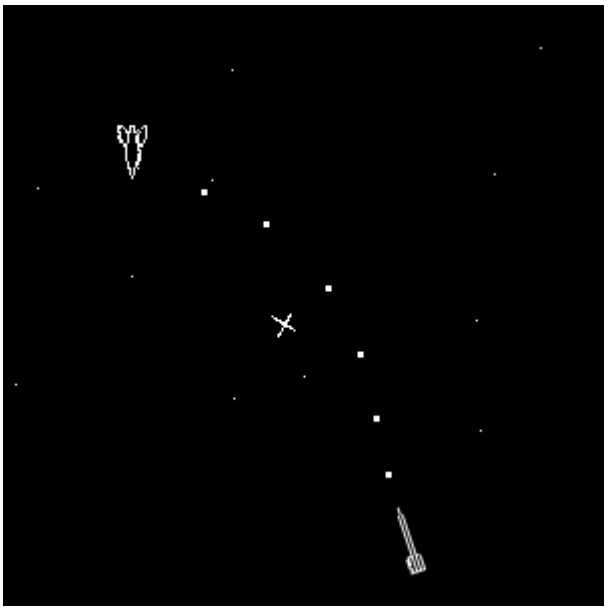
This lesson can be extended to a semester long project, or simplified by instructing students to create or enhance any of the demo games that are provided by the software developer. The "space cleaner" game, for example can be modified within a single class period. If you choose to have students modify a game, have them first explore the existing game, meet as a team to determine what changes they would like to engineer into the game, then have them attempt to execute their plan.

◆ Time Needed

One to two 45 minute sessions

Student Resource
Software Engineering: Game History

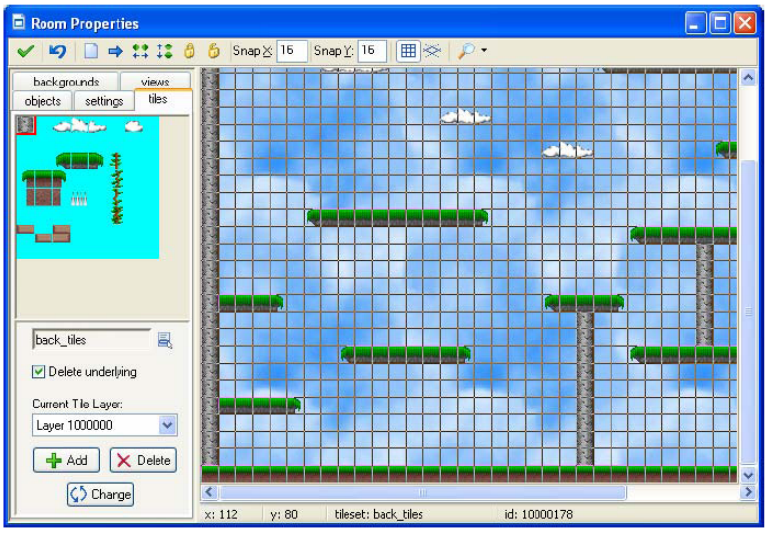
◆ The Start of Computer Gaming



Although personal computers only became popular with the development of the microprocessor, mainframe and minicomputers have been used for computer gaming since at least the 1960s. One of the first computer games was developed in 1961, when MIT students Martin Graetz and Alan Kotok, with MIT employee Stephen Russell, developed Spacewar! on a computer used for statistical calculations. As seen on the right, the game consisted of two player-controlled spaceships maneuvering around a central star, each attempting to destroy the other.

The first generation of PC games consisted of text adventures or interactive fiction, in which the player communicated with the computer by entering commands through a keyboard. By the mid-1970s, games were developed and distributed through hobbyist groups and gaming magazines, such as Creative Computing and later Computer Gaming World. These publications provided game code that could be typed into a computer and played, encouraging readers to submit their own software to competitions.

◆What Software Engineers Do



Software engineers working in applications or systems development analyze users' needs and design, construct, test, and maintain computer applications software or systems. Software engineers can be involved in the design and development of many types of software, including software for operating systems and network distribution, and compilers, which convert programs for execution on a computer. In programming, or coding, software engineers instruct a computer, line by line, how to perform a function. They also solve technical problems that arise. Software engineers must possess strong programming skills, but are more concerned with developing algorithms and analyzing and solving programming problems than with actually writing code.

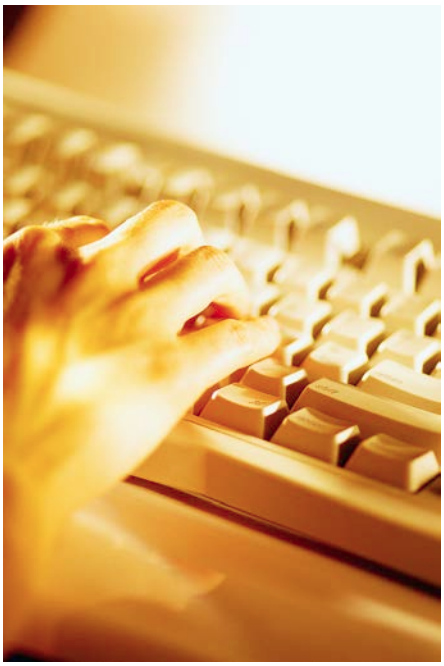
Student Resource
Software Engineering: Algorithms

◆ What is an Algorithm?

In mathematics, computing, linguistics, and related disciplines, an algorithm is a finite list of well-defined instructions for accomplishing some task which, given an initial state, will terminate in a defined end-state. The concept of an algorithm originated as a means of recording procedures for solving mathematical problems such as finding the common divisor of two numbers or multiplying two numbers.

The concept was formalized in 1936 through Alan Turing's Turing machines and Alonzo Church's lambda calculus, which in turn formed the foundation of computer science. A simple example is a flow chart which is basically a logical sequence of steps for solving a problem.

◆ Computer Applications



Algorithms are essential to the way computers process information, because a computer program is essentially an algorithm that tells the computer what specific steps to perform (in what specific order) in order to carry out a specified task, such as calculating an invoice, printing report cards, or completing a budget analysis.

Because an algorithm is a precise list of precise steps, the order of computation will almost always be critical to the functioning of the algorithm. Instructions are usually assumed to be listed explicitly, and are described as starting 'from the top' and going 'to the bottom,' -- sometimes also called "flow of control."

Every field of science has its own problems and needs efficient algorithms. Related problems in one field are often studied together.

Some example classes are search algorithms, sorting algorithms, merge algorithms, numerical algorithms, graph algorithms, string algorithms, computational geometric algorithms, combinatorial algorithms, machine learning, cryptography, data compression algorithms and parsing techniques.

Program Your Own Game

Student Worksheet: You are the Engineer!

◆ You are a team of engineers which has to tackle the challenge of developing a new computer game for use by 6 - 10 year old children.

◆ Preparation

1. Review the various Student Reference Sheets.
2. Read the Basic Programming guide that has been provided to you.

◆ Activity Steps

1. As a team, come up with a plan or idea for your game -- also come up with a name for your game. In the box below, write a two sentence description of your new game that might be used in an advertisement:

Game Name: Description:

2. Work together using the software (http://www.yoyogames.com/gamemaker/studio/) and build your game.

3. Present your game to other teams in your class so they can see how your game works (you'll test out their games as well).

4. Complete the following questions about your game and other games developed in your classroom.

5. As a team, present your findings and reflections to the class.

Student Worksheet: You are the Engineer!

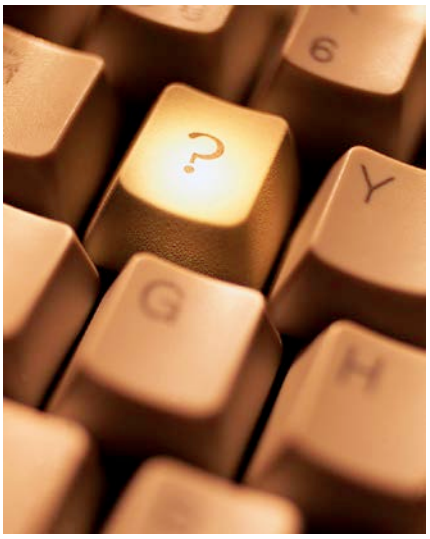
◆ Evaluation Questions

1. How did your plan for your game change once you tried to build it using the software provided?

2. How long do you think it would take to develop a new piece of word processing or graphic software? How many people do you think it might take to work on the engineering team to build this type of software? Why?

3. Did you find that it was easier or harder than you thought to program a computer game? Why?

4. What challenges did you face in building your game?



5. What other games developed in your classroom did you like the best? Why? What features appealed to you?

6. Do you think it was easier or harder to have developed this game as part of a team? Do you think you would have been able to create your new design if you had not been working in a team? What are the advantages of teamwork vs. working alone?

7. What did you learn about how engineers solve problems through this lesson?

For Teachers:
Alignment to Curriculum Frameworks

Note: Lesson plans in this series are aligned to one or more of the following sets of standards:

- U.S. Science Education Standards (http://www.nap.edu/catalog.php?record_id=4962)
- U.S. Next Generation Science Standards (http://www.nextgenscience.org/)
- International Technology Education Association's Standards for Technological Literacy (http://www.iteea.org/TAA/PDFs/xstnd.pdf)
- U.S. National Council of Teachers of Mathematics' Principles and Standards for School Mathematics (http://www.nctm.org/standards/content.aspx?id=16909)
- U.S. Common Core State Standards for Mathematics (http://www.corestandards.org/Math)
- Computer Science Teachers Association K-12 Computer Science Standards (http://csta.acm.org/Curriculum/sub/K12Standards.html)

◆National Science Education Standards Grades 5-8 (ages 10 - 14)

CONTENT STANDARD A: Science as Inquiry

As a result of activities, all students should develop

- Abilities necessary to do scientific inquiry

CONTENT STANDARD E: Science and Technology

As a result of activities in grades 5-8, all students should develop

- Abilities of technological design
- Understandings about science and technology

CONTENT STANDARD G: History and Nature of Science

As a result of activities, all students should develop understanding of

- History of science

◆National Science Education Standards Grades 9-12 (ages 14-18)

CONTENT STANDARD A: Science as Inquiry

As a result of activities, all students should develop

- Abilities necessary to do scientific inquiry

CONTENT STANDARD E: Science and Technology

As a result of activities, all students should develop

- Abilities of technological design
- Understandings about science and technology

CONTENT STANDARD G: History and Nature of Science

As a result of activities, all students should develop understanding of

- Historical perspectives

◆Next Generation Science Standards Grades 3-5 (Ages 8-11)

Engineering Design

Students who demonstrate understanding can:

- 3-5-ETS1-1.Define a simple design problem reflecting a need or a want that includes specified criteria for success and constraints on materials, time, or cost.
- 3-5-ETS1-2.Generate and compare multiple possible solutions to a problem based on how well each is likely to meet the criteria and constraints of the problem.
- 3-5-ETS1-3.Plan and carry out fair tests in which variables are controlled and failure points are considered to identify aspects of a model or prototype that can be improved.

For Teachers:
Alignment to Curriculum Frameworks (continued)

◆Next Generation Science Standards Grades 6-8 (Ages 11-14)

Engineering Design

Students who demonstrate understanding can:

- MS-ETS1-2 Evaluate competing design solutions using a systematic process to determine how well they meet the criteria and constraints of the problem.

◆Principles and Standards for School Mathematics

Number and Operations Standard

As a result of activities, all students should develop

- Understand numbers, ways of representing numbers, relationships among numbers, and number systems.
- Compute fluently and make reasonable estimates.

Connections Standard

As a result of activities, all students should develop

- Understand how mathematical ideas interconnect and build on one another to produce a coherent whole.
- Recognize and apply mathematics in contexts outside of mathematics.

◆Standards for Technological Literacy - All Ages

The Nature of Technology

- Standard 2: Students will develop an understanding of the core concepts of technology.
- Standard 3: Students will develop an understanding of the relationships among technologies and the connections between technology and other fields of study.

Technology and Society

- Standard 7: Students will develop an understanding of the influence of technology on history.

Design

- Standard 9: Students will develop an understanding of engineering design.
- Standard 10: Students will develop an understanding of the role of troubleshooting, research and development, invention and innovation, and experimentation in problem solving.

Abilities for a Technological World

- Standard 12: Students will develop abilities to use and maintain technological products and systems.

The Designed World

- Standard 17: Students will develop an understanding of and be able to select and use information and communication technologies.

For Teachers:
Alignment to Curriculum Frameworks

◆CSTA K-12 Computer Science Standards Grades 6-9 (ages 11-14)

5. 2 Level 2: Computer Science and Community (L2)

- Computational Thinking: (CT)

  6. Describe and analyze a sequence of instructions being followed (e.g., describe a character's behavior in a video game as driven by rules and algorithms).

  7. Represent data in a variety of ways including text, sounds, pictures, and numbers.

  14. Examine connections between elements of mathematics and computer science including binary numbers, logic, sets and functions.

- Collaboration (CL)

  2. Collaboratively design, develop, publish, and present products (e.g., videos, podcasts, websites) using technology resources that demonstrate and communicate curriculum concepts.

  3. Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities.

  4. Exhibit dispositions necessary for collaboration: providing useful feedback, integrating feedback, understanding and accepting multiple perspectives, socialization.

◆CSTA K-12 Computer Science Standards Grades 6-9 (ages 11-14)

5. 2 Level 2: Computer Science and Community (L2)

- Computing Practice & Programming (CPP)

  1. Select appropriate tools and technology resources to accomplish a variety of tasks and solve problems.

  3. Design, develop, publish, and present products (e.g., webpages, mobile applications, animations) using technology resources that demonstrate and communicate curriculum concepts.

  5. Implement problem solutions using a programming language, including: looping behavior, conditional statements, logic, expressions, variables, and functions.

- Computers & Communications Devices (CD)

  1. Recognize that computers are devices that execute programs.

  3. Demonstrate an understanding of the relationship between hardware and software.

◆CSTA K-12 Computer Science Standards Grades 9-10 (ages 14-15)

5.3 Level 3: Applying Concepts and Creating Real-World Solutions (L3)

- Computational Thinking: (CT)
  2. Describe a software development process used to solve software problems (e.g., design, coding, testing, verification).
  6. Analyze the representation and trade-offs among various forms of digital information.
- Collaboration (CL)
  1. Work in a team to design and develop a software artifact.
  4. Identify how collaboration influences the design and development of software products.
- Computing Practice & Programming (CPP)
  4. Apply analysis, design, and implementation techniques to solve problems (e.g., use one or more software lifecycle models).
  6. Select appropriate file formats for various types and uses of data.