SCALABLE GAME DESIGN

CS PRINCIPLES

Scalable Game Design and AP Computer Science Principles – Bringing Computational Thinking to all Students

Creativity

Algorithms

Abstraction

Data and Information

Programming

The Internet

Global Impact

Version 1.0

Created by:

Susan B. Miller
Susan.Miller@colorado.edu
University of Colorado
School of Education

This curriculum has been designed as part of the Scalable Games Design project.

This material is based upon work supported by the National Science Foundation under Grant No. DRL-1312129 and CNS-1138526. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

This material is funded through Google's Computer Science for High School (CS4HS) Grant (Repenning CS4HS Google K12 Education funding).

Contents

This page intentionally left blank.

Computer Science Principles – an Overview

AP Computer Science Principles introduces students to a broad view of the central ideas of computer science, instilling the ideas of computational thinking, and inviting students to understand the role of computers in today's society. Its focus is on fostering student creativity when developing computational artifacts. This course is designed to appeal to a diverse group of students, moving past the study of machines and systems to the impact of computing on our society.

The Computer Science Principles capture two core competencies that students should be able to demonstrate. First, these principles focus on **six key computational thinking practices** that are inherent to the work of computer scientists. Second, these principles encompass **seven big ideas of computing**, that students must understand as part of the 21st century skills. The six key computational thinking practices are interwoven throughout the seven big ideas – they are not separate from them. Each big idea incorporates at least two or three of the practices, though not all six practices are evident in every big idea. Let's take a look at both of these competencies.

Computational Thinking Practices

The **six key computational thinking practices** capture key aspects of the work of computer scientists. These practices are designed to help students make sense of problems in a way that enables them to see possible solutions.

Connecting Computing

- Identification of impacts of computing.
- Description of connections between people and computing.
- Explanation of connections between computing concepts.

Creating computational artifacts

- a. Creation of an artifact with a practical, personal, or societal intent.
- b. Selection of appropriate techniques to develop a computational artifact.
- c. Use of appropriate algorithmic and information-management principles.

Abstracting

- a. Explanation of how data, information, or knowledge are represented for
- b. Explanation of how abstractions are used in computation or modeling.
- c. Identification of abstractions.
- d. Description of modeling in a computational context.

Analyzing problems and artifacts

- a. Evaluation of a proposed solution to a problem.

- b. Location and correction of errors.
- c. Explanation of how an artifact functions.
- d. Justification of appropriateness and correctness.

Communicating

- a. Explanation of the meaning of a result in context.
- b. Description using accurate and precise language, notation, or visualizations
- c. Summary of purpose.

Collaborating

- a. Collaboration of participants in solving a computational problem.
- b. Collaboration of participants in producing an artifact.
- c. Collaboration at a large scale.

Seven Big Ideas of Computing

The **seven big ideas of computing**, ask students to consider essential questions for each idea. These seven big ideas of computing embrace ideas foundational to computer science. The seven big ideas are listed below:

Creativity:
Computing is a creative activity.

- How can a creative development process affect the creation of computational artifacts?
- How can computing and the use of computational tools foster creative expression?
- How can computing extend traditional forms of human expression and experience?

Abstraction:
Abstraction reduces information and detail to facilitate focus on relevant concepts.

- How are vastly different kinds of data, physical phenomena, and mathematical concepts represented on a computer?
- How does abstraction help us in writing programs, creating computational artifacts, and solving problems?
- How can computational models and simulations help generate new understanding and knowledge?

Data:
Data and information facilitate the creation of knowledge.

- How can computation be employed to help people process data and information to gain insight and knowledge?

- How can computation be employed to facilitate exploration and discovery when working with data?
- What considerations and trade-offs arise in the computational manipulation of data?
- What opportunities do large data sets provide for solving problems and creating knowledge?

Algorithms:
Algorithms are used to develop and express solutions to computational problems.

- How are algorithms implemented and executed on computers and computational devices?
- Why are some languages better than others when used to implement algorithms?
- What kinds of problems are easy, what kinds are difficult, and what kinds are impossible to solve algorithmically?
- How are algorithms evaluated?

Programming:
Programming enables problem solving, human expression, and creation of knowledge.

- How are programs developed to help people, organizations, or society solve problems?
- How are programs used for creative expression, to satisfy personal curiosity, or to create new knowledge?
- How do computer programs implement algorithms?
- How does abstraction make the development of computer programs possible?
- How do people develop and test computer programs?
- Which mathematical and logical concepts are fundamental to computer programming?

Internet:
The Internet pervades modern computing.

- What is the Internet? How is it built? How does it function?
- What aspects of the Internet's design and development have helped it scale and flourish?
- How is cybersecurity impacting the ever--increasing number of Internet users?

Global Impact:
Computing has global impacts.

- How does computing enhance human communication, interaction, and cognition?
- How does computing enable innovation?
- What are some potential beneficial and harmful effects of computing?
- How do economic, social, and cultural contexts influence innovation and the use of computing?

This page intentionally left blank.

Scalable Game Design and Computer Science Principles

Scalable Game Design offers a strong method of teaching many of the Computer Science Principles[1], even for teachers who may not feel ready to teach the AP Computer Science course. Computer Science Principles provides a great deal of flexibility in how programming is taught, as the emphasis is on creativity both in design and in problem solving techniques. As a result, students are motivated to use the curriculum to pursue their own interests relative to their own life experiences.

Proposed Curriculum

Scalable Game Design is a low threshold, high ceiling way to teach the programming aspects of CSP. Through learning to program games, and later program simulations, Scalable Game Design activities allow students with prior experience or no experience with programming to engage, enjoy, and learn from the activities. Students with prior experience with Scalable Game Design can skip Units 1 and 2 and explore more complex programming activities.

Note that all of the recommended lesson plans are the designated guided discovery lesson plans. These plans use an approach that invites students to see one another as experts, and moves the teacher to a support role in the classroom, rather than the usual expert role.

Suggested curriculum implementation:

Unit 1: Frogger

Complete Frogger Guided Discovery Lesson Plan

http://sgd.cs.colorado.edu/wiki/Featured_Lesson_Plans#Frogger

[1] SGD specializes in bringing computer science principles' programming module to classrooms through game and simulation design and creation. Though the activities simultaneously cover many of the big ideas and principles underlying the course in its entirety, it is not designed to fulfill all requirements for an AP CSP course. A summary of the standards matching appears on the following page, and a full list of standards matching appears in the appendix. Additional resources will be needed to address areas not covered by this curriculum. For example, understanding the design and use of the internet is a critical skill, but not one covered in our curriculum. Teachers are directed to the AP Computer Science Principles main webpage to access other support materials. http://apcsprinciples.org/

Unit 2: Journey or PacMan

Complete Journey Guided Discovery Lesson Plan

http://sgd.cs.colorado.edu/wiki/Featured_Lesson_Plans#Journey

OR

Complete Journey Guided Discovery Lesson Plan

http://sgd.cs.colorado.edu/wiki/Featured_Lesson_Plans#PacMan

Unit 3: Contagion

Basic Contagion Simulation Guided Discovery Lesson Plan

http://sgd.cs.colorado.edu/wiki/Featured_Lesson_Plans#Contagion_Simulation

Unit 4: Create your own simulation

Build Your Own Simulation Guided Discovery Lesson Plan

http://sgd.cs.colorado.edu/wiki/Featured_Lesson_Plans#Build_your_own_Simulation

Unit 5: Predator/Prey

While Scalable Game Design does not have prepared curricular materials for this unit at this time, we recommend that teachers use the Predator/Prey simulation that comes with AgentSheets to create a large data set. This dataset can then be the focus of discussions covering the topics that reference large data sets. (See the summary on page 12 for topics to be covered with this data set.) The Predator/Prey simulation can also be built by the students using the tutorial found here: http://sgd.cs.colorado.edu/wiki/Predators_and_Prey_Design.

Assessment

There are two different ways students are assessed on concepts covered in the AP CSP course. First, as part of the class itself, students must complete and submit two performance tasks (CREATE and EXPLORE) where all work is done during class time. Second, students also take a multiple choice test.

Performance Tasks

Two performance tasks are completed by students as part of the AP CSP Course. These tasks include both collaborative and individual components.

Create Performance Task

Using the Scalable Game Design curriculum will also enable students to fully complete the CREATE performance task. It requires the following actions:

> *For the collaborative submissions, you are required to work with your partner(s) to:*
> - *Design, create, and demonstrate the running of a program that solves a problem of interest to all partners and/or that represents an expression of shared personal interests among partners.*
> - *Solicit and provide feedback.*
>
> *For the individual submissions, you are required to work independently to:*
> - *Answer questions about your collaborative program and the process of collaboration.*
> - *Design and create a program on a topic that interests you and that solves a problem and/or provides an opportunity for self-expression.*
> - *Answer questions about your individual program.*
>
> (AP CSP, 2015)

As part of this task, students will work to collaboratively to

- program a simulation
- annotate the code
- create a video that records how the program works and they process used to create it

Explore Performance Task

The second performance task, the EXPLORE performance task, requires the following actions:

> *Select a computing innovation to investigate that has had a significant impact on society, economy, or culture. You must be able to convey an understanding of the innovation by discussing the relationship of the innovation to the principles of computer science, particularly the role data plays and the cybersecurity issues that exist. You will then*

> *address specific prompts that call for either a written response or a visual/audio response.*
>
> (AP CSP, 2015)

While this task is not directly related to the Scalable Game Design curriculum, teachers will find that bringing programming into the classroom will easily translate into broader discussions about computer innovations. Students will be able to draw on their knowledge of programming when completing this task.

AP CSP Test

Using the Scalable Game Design curriculum for the programming portions of AP CSP will enable students to easily answer all of the programming questions on the multiple choice test. More details on the AP CSP Test can be found here: https://advancesinap.collegeboard.org/stem/computer-science-principles.

Standards Matching – Scalable Game Design to AP Computer Science Principles

Scalable Game Design Units						Essential Knowledge - What students need to know
1 Frogger	2 Journey/ PacMan	3 Contagion	4 Simulation	5 Predator Prey	Outside Resources	
						CREATIVITY
♦	♦	♦	♦			1.1.1 Apply a creative development process when creating computational artifacts. [P2]
♦	♦	♦	♦			1.2.1 Create a computational artifact for creative expression. [P2]
		♦	♦			1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. [P2]
♦	♦	♦	♦			1.2.3 Create a new computational artifact by combining or modifying existing artifacts. [P2]
♦	♦	♦	♦			1.2.4 Collaborate in the creation of computational artifacts. [P6]
♦	♦	♦	♦			1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts. [P4]
♦	♦	♦	♦			1.3.1 Use computing tools and techniques for creative expression. [P2]
						ABSTRACTION
					♦	2.1.1 Describe the variety of abstractions used to represent data. [P3]
					♦	2.1.2 Explain how binary sequences are used to represent digital data. [P5]
		♦	♦			2.2.1 Develop an abstraction when writing a program or creating other computational artifacts. [P2]
		♦	♦			2.2.2 Use multiple levels of abstraction to write programs. [P3]
					♦	2.2.3 Identify multiple levels of abstractions that are used when writing programs. [P3]
		♦	♦			2.3.1 Use models and simulations to represent phenomena. [P3]
		♦	♦			2.3.2 Use models and simulations to formulate, refine, and test hypotheses. [P3]
						DATA AND INFORMATION
				♦		3.1.1 Use computers to process information, find patterns, and test hypotheses about digitally processed information to gain insight and knowledge. [P4]
				♦		3.1.2 Collaborate when processing information to gain insight and knowledge. [P6]
				♦		3.1.3 Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notations, and precise language. [P5]
				♦		3.2.1 Extract information from data to discover and explain connections, patterns, or trends. [P1]
				♦		3.2.2. Use large data sets to explore and discover information and knowledge. [P3]
					♦	3.3.1 Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information. [P4]
						ALGORITHMS
♦	♦	♦	♦			4.1.1 Develop an algorithm for implementation in a program. [P2]
♦	♦	♦	♦			4.1.2 Express an algorithm in a language. [P5]
			♦	♦		4.2.1 Explain the difference between algorithms that run in a reasonable time and those that do not run in a reasonable time. [P1]
			♦	♦		4.2.2 Explain the difference between solvable and unsolvable problems in computer science. [P1]
					♦	4.2.3 Explain the existence of undecidable problems in computer science. [P1]
			♦	♦		4.2.4 Evaluate algorithms analytically and empirically for efficiency, correctness, and clarity. [P4]
						PROGRAMMING
♦	♦	♦	♦			5.1.1 Develop a program for creative expression, to satisfy personal curiosity, or to create new knowledge. [P2]
♦	♦	♦	♦			5.1.2 Develop a correct program to solve problems. [P2]
♦	♦	♦	♦			5.1.3 Collaborate to develop a program. [P6]
♦	♦	♦	♦			5.2.1 Explain how programs implement algorithms. [P3]
			♦	♦		5.3.1 Use abstraction to manage complexity in programs. [P3]
♦	♦	♦	♦	♦		5.4.1 Evaluate the correctness of a program. [P4]
						THE INTERNET
					♦	6.1.1 Explain the abstractions in the Internet and how the Internet functions. [P3]
					♦	6.2.1 Explain characteristics of the Internet and the systems built on it. [P5]
					♦	6.2.2 Explain how the characteristics of the Internet influence the systems built on it. [P4]
					♦	6.3.1 Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it. [P1]
						GLOBAL IMPACT
					♦	7.1.1 Explain how computing innovations affect communication, interaction, and cognition. [P4]
					♦	7.1.2 Explain how people participate in a problem--- solving process that scales. [P4]
					♦	7.2.1 Explain how computing has impacted innovations in other fields. [P1]
					♦	7.3.1 Analyze the beneficial and harmful effects of computing. [P4]
					♦	7.4.1 Explain the connections between computing and economic, social, and cultural contexts. [P1]

This page intentionally left blank.

Scalable Game Design CSP Curriculum: Unit 1 Frogger

Students will be asked, as part of the AP CSP course, to design and create their own computer-based artifact. This is the first step of this process[2]. Your class will design and create a Frogger game together, working at first in a step-by-step fashion to learn the software and basic programming skills. While the initial activities are done together, as student skills increase, the students will be asked to do more and more on their own.

Materials:

- Frogger Guided Discovery Curricular Materials
- Code Comment PowerPoint

Computer Science Principles Big Ideas:

Creativity:
Focus on the need for creative thinking to solve problems. Creativity can start with agent and worksheet design, but the real goal is to find their own creativity in their programming.

Abstraction:
Students learn to develop an abstraction when writing a program. In doing so, they must remove detail and generalize functionality.

Frogger begins the process of seeing abstractions through programming. The movement of the cars is an example of this. In Frogger, a frog must cross a highway. This is abstracted as a scene where cars move across the screen along a highway background. In order to create this scene through programming, the students must recognize that they are creating the functionality of the cars moving through the tunnel as an abstraction. This occurs in two ways. First, the cars do not really move through the tunnel. Instead, the programming involves generating new cars at the left tunnel, and then absorbing them at the right tunnel. Second, the cars appear to come out of the tunnel at random intervals. This programming involves creating a sense of randomness of the cars by using probability.

Students also see this when there is a collision between the frog and the truck. In the design of the game, if the frog is hit by the truck, the truck honks his horn and the frog is squished. However, all of the code for the collision is given to the Frog

[2] If students have already programmed Frogger, proceed to Unit 2.

agent. The code checks to see if there is a truck next to the frog. If there is, there is a honking sound, and the frog changes to a 'dead frog' depiction. In this case, the Frog 'makes' the honking sound, but the player will assume that the truck makes the sound.

Programming

Students will program Frogger to learn the process of using abstractions to create a game. By allowing students to creatively adapt the game (such as by adding challenges provided as part of the curricular materials), students learn to develop for creative expression and add additional outcomes not originally planned.

Documentation is a key aspect of computing and students should be encouraged to document their game.

Students should be expected to annotate their code when building Frogger. A PowerPoint presentation gives a very brief overview on the need for commenting or annotating the code and shows students that they can quickly find comments as well as the symbols that precede the comments to separate them from the actual programming code.

Collaboration is a key part of programming. Students should be encouraged to work together to:

Ease the burden for all students

Facilitate multiple perspectives of possible solutions

Make the most of each student's individual talents

Debugging is an essential skill to find mistakes in the code. Knowledge of what the program is supposed to do should guide all testing and debugging activities. Students should be encouraged to justify and explain a program's correctness as well as the functionality of the game. The functionality is best described at a high level by what a program does, not by how the code works.

Computer Science Principles Computational Thinking Practices:

The following Computer Science Computational Thinking Practices should be highlighted as part of these lessons.

Creating Computational Artifacts

Students will create their own game and upload it to the arcade where others can play it. Sharing their game also allows others to download their game and modify it, either as an improvement or as an enhancement.

Analyzing Problems and Artifacts

Games should be analyzed on multiple levels, including thinking about the aesthetics as well as pragmatic components of the game. Evaluating their own and their classmate's work, and justifying their work orally and in writing are important aspects of analysis and artifact creation.

Communicating

Expectations for communications include oral and written explanation of the purpose of their game on a macro and code level. Precise language to reflect the computational thinking patterns learned is also needed.

Frogger will enable students to use and describe the following computational thinking patterns:

- *Absorb*
- *Generate*
- *Collision*
- *Transport*
- *Cursor Control*

Collaborating

Students working together often achieve more than students working alone. Students can collaborate by working together on solving a particular problem, or by designing the game together. The goal should be creating a high-quality artifact, that highlights the contributions of each student.

This page intentionally left blank.

Scalable Game Design CSP Curriculum: Unit 2 Journey or PacMan

Students will be asked, as part of the AP CSP course, to design and create their own computer-based artifact. This is the second step of this process[3]. Now that students are becoming more proficient with both the software and the process of programming, students will be able to do more one their own. Using the student pages from the curricular materials, have students build EITHER Journey or PacMan. Both games teach the same basic skills and will ready the students for the next unit, creating a simulation.

Materials:

- Journey OR PacMan Guided Discovery Curricular Materials
- Java Code PowerPoint

Computer Science Principles Big Ideas:

Creativity:
Focus on the need for creative thinking to solve problems. Creativity can start with agent and worksheet design, but the real goal is to find their own creativity in their programming.

Abstraction:
Students learn to develop and abstraction when writing a program. In doing so, they must remove detail and generalize functionality.

In both Journey and PacMan, students will use abstraction to think about how to determine when the game is over. How can we simulate or model someone checking to see if all the goals or pellets are gone?

Algorithms

Students will use basic algorithms to track values. As the students program the game controller to determine if the game is over, the program will use an iterative process.

In both Journey and PacMan, students will use variables (either for the number of goals, or the number of pellets remaining).

Knowledge of standard algorithms (Computational Thinking Patterns) can help in constructing new algorithms. Different algorithms can be developed to solve the same problem. Sometimes, finding new ways to solve a problem with a different algorithm

[3] If students have programmed both Frogger and Journey or PacMan, proceed to Unit 3.

can lead to new insight into the problem itself. Encourage students (especially when collaborating) to find different ways to solve the same problem.

Programming

Procedures (called METHODS) are used by various agents. These methods create a reusable programming abstraction, and may incorporate parameters and return values.

An example of a procedure in a class room is the response to a fire alarm. Students learn the steps required of them when the alarm is activated – for example, they likely line up silently, proceed to the nearest door, and wait for a teacher to take attendance. This procedure is consistently applied, for all students at the school.

In the same way, code can be written as a procedure to take certain steps when the method is activated (or called). For example, one might write a method that enables the PacMan game to increase the score by one with each pellet the PacMan eats. It is important to point out that methods generalize a solution by allowing a procedure to be used rather than duplicated code.

Furthermore, algorithms are implemented within the program. The algorithms will use variables and are executed sequentially.

Analyzing the program becomes more and more important. . Encourage students to use method and variable names with meaning, such as 'CHASE' or 'SCENT' rather than 'METHOD1' or 'V' for variable. As the programming calls for methods, meaningful names, both for these methods as well as for variables, will help people better understand the program

Documentation is a key aspect of computing and students should be encouraged to document their game both by annotating the code, as well as writing written documentation.

Collaboration is a key part of programming. Encourage stuednts to work together in order to:

Ease the burden for all students

Facilitate multiple perspectives of possible solutions

Make the most of each student's individual talents

Debugging is an essential skill to find mistakes in the code. Knowledge of what the program is supposed to do should guide all testing and debugging activities. Students should be encouraged to justify and explain a program's correctness. Students should

also be able to justify the functionality of the game. The functionality is best described at a high level by what a program does, not by how the code works.

Students should be thinking about the code being written, and should be exposed to the Java code behind the visual language. Use the JAVA CODE PowerPoint to provide examples to the students of the Java code.

Computer Science Principles Computational Thinking Practices:

Creating Computational Artifacts

Students will create their own game and upload it to the arcade where others can play it. Sharing their game also allows others to download their game and modify it, either as an improvement or as an enhancement.

Particularly for students working on Journey, creativity should be stressed. How can students change the game (even slightly) to make it more exciting or more relevant? Because PacMan is so universally known, creativity is more difficult. However, students can still build in added features on their own.

Analyzing Problems and Artifacts

Games should be analyzed on multiple levels, including thinking about the aesthetics as well as pragmatic components of the game. Students should evaluate their own work, as well as their colleagues' work. Students should be expected to justify their work both orally as well as in writing.

Communicating

Students should be expected to communicate about their game, both orally and in writing. Communications should be both at the code level, as well as on a macro level, explaining to others the purpose of their game. Language used should be precise and should reflect the computational thinking patterns being learned.

Journey or PacMan will enable students to use and describe the following computational thinking patterns:

Reviewed from Frogger:

- *Absorb*
- *Generate*
- *Collision*
- *Cursor Control*

New to Journey/PacMan

- *Diffusion*
- *Polling*
- *Hill Climbing*

Collaborating

Students working together often achieve more than students working alone. Students can collaborate by working together on solving a particular problem, or by designing the game together. The goal should be creating a high-quality artifact, in a way that highlights the contributions of each student.

Scalable Game Design CSP Curriculum: Unit 3 Contagion

Students will be asked, as part of the AP CSP course, to design and create their own computer-based artifact. This is the third step of this process. Now that students are becoming more proficient with both the software and the process of programming, students will be able to build a simulation. Using the student pages from the curricular materials, have students build Contagion.

Materials:

- Contagion Guided Discovery Curricular Materials

Computer Science Principles Big Ideas:

Creativity:
Focus on the need for creative thinking to solve problems. Creativity can start with agent and worksheet design, but the real goal is to find their own creativity in their programming.

> *In Contagion, there will be two areas for creativity beyond the basic agent and worksheet creation. First, students must think creatively about how viruses are transmitted, and how they will simulation that transmission. Second, students must think creatively about how people will recover (and who will not recover).*

Abstraction:
Students learn to develop and abstraction when writing a program. In doing so, they must remove detail and generalize functionality.

> *Creating a simulation is the ultimate abstraction in this course. Students cannot create an exact replication of a contagious disease, and therefore must decide which details are critical, and which are not.*

Algorithms

Again, knowledge of standard algorithms (Computational Thinking Patterns) can help in constructing new algorithms. Different algorithms can be developed to solve the same problem. Sometimes, finding new ways to solve a problem with a different algorithm can lead to new insight into the problem itself. Encourage students (especially when collaborating) to find different ways to solve the same problem.

In Contagion, students will use variables and will be able to see the effects of changing those variables. Students will also begin to pass variables from one agent to another, allowing for both global and local variables.

Programming

Procedures (called METHODS) are used by various agents. Students will continue to use methods to repeat various blocks of code.

Analyzing the program becomes more and more important. As the programming calls for methods, meaningful names, both for these methods as well as for variables, will help people understand the program. Therefore, encourage students to use method and variable names with meaning, such as 'CHASE' or 'SCENT' rather than METHOD1 or V for variable.

Documentation is a key aspect of computing and students should continue to be encouraged to document their game.

Collaboration is a key part of programming. Students should continue to be encouraged to work together.

Debugging is an essential skill to find mistakes in the code. Knowledge of what the program is supposed to do should guide all testing and debugging activities. Students should be encouraged to justify and explain a program's correctness. Students should also be able to justify the functionality of the game. The functionality is best described at a high level by what a program does, not by how the code works.

Computer Science Principles Computational Thinking Practices:

Creating Computational Artifacts

This simulation will again be uploaded to the arcade. This enables others to share and modify each design.

Analyzing Problems and Artifacts

Games should be analyzed on multiple levels, including thinking about the aesthetics as well as pragmatic components of the game. Students should evaluate their own work, as well as their colleagues' work. Students should be expected to justify their work both orally as well as in writing.

For Contagion, this includes justifying the assumptions used as part of the design.

Communicating

Students should be expected to communicate about their game, both orally and in writing. Communications should be both at the code level, as well as on a macro level, explaining to others the purpose of their game. Language used should be precise and should reflect the computational thinking patterns being learned.

Contagion will enable students to use and describe the following computational thinking patterns:

Reviewed from Frogger/Journey/PacMan

- *Absorb*
- *Generate*
- *Collision*
- *Cursor Control*
- *Diffusion*
- *Polling*
- *Hill Climbing*

New to Contagion

- *Perceive/Act*
- *Seeking*

Collaborating

Students working together often achieve more than students working alone. Students can collaborate by working together on solving a particular problem, or by designing the game together. The goal should be creating a high-quality artifact, in a way that highlights the contributions of each student.

This page intentionally left blank.

Scalable Game Design CSP Curriculum: Unit 4 Design a Simulation

Students will be asked, as part of the AP CSP course, to design and create their own computer-based artifact. This is the final step of this process. Now that students are becoming more proficient with both the software and the process of programming, students will be able to create their own simulation.

Materials:

- None

Computer Science Principles Big Ideas:

Creativity:
Focus on the need for creative thinking to solve problems. Creativity can start with agent and worksheet design, but the real goal is to find their own creativity in their programming in solving a problem or achieving a goal.

Students should be encouraged to design their own simulation. Some students will find that type of open ended assignment to be stressful, not knowing what types of simulations can be built. For students who need added support, you might want to share with them the types of simulations we have seen including:

- *A washing machine – how it heats the water, mixes the soap, and washes the clothes*
- *A fall day, and how wind speed affects the number of leaves falling from a tree*
- *The life-cycle of salmon in the river*
- *The effects of a dockworker who checks boats for Zebra Mussels on overall growth of Zebra Mussels in a pond.*
- *The ways in which gossip travels throughout a school, and the effects of students who refuse to pass on the gossip*
- *A model of the airport lines through security – in what ways might the lines be changed so that people progress through faster*[4].

[4] http://www.latimes.com/business/la-fi-tsa-ideas-to-speed-up-screening-lines-20140725-story.html

Abstraction:
Students learn to develop and abstraction when writing a program. In doing so, they must remove detail and generalize functionality.

> *Creating a simulation is the ultimate abstraction in this course. Students cannot create an exact replication of their situation, and therefore must decide which details are critical, and which are not.*

Algorithms:
Again, knowledge of standard algorithms (Computational Thinking Patterns) can help in constructing new algorithms. Different algorithms can be developed to solve the same problem. Sometimes, finding new ways to solve a problem with a different algorithm can lead to new insight into the problem itself. Encourage students (especially when collaborating) to find different ways to solve the same problem.

> *In the simulation, students should use variables and will be able to see the effects of changing those variables. Students will also begin to pass variables from one agent to another, allowing for both global and local variables.*

Programming:
Procedures (called METHODS) are used by various agents. Students will continue to use methods to repeat various blocks of code.

Analyzing the program becomes more and more important. As the programming calls for methods, meaningful names, both for these methods as well as for variables, will help people better understand the program. Therefore, encourage students to use method and variable names with meaning, such as 'CHASE' or 'SCENT' rather than METHOD1 or V for variable.

Documentation is a key aspect of computing and students should continue to be encouraged to document their game.

Collaboration is a key part of programming. Students should continue to be encouraged to work together.

Debugging is an essential skill to find mistakes in the code. Knowledge of what the program is supposed to do should guide all testing and debugging activities. Students should be encouraged to justify and explain a program's correctness. Students should also be able to justify the functionality of the game. The functionality is best described at a high level by what a program does, not by how the code works.

Computer Science Principles Computational Thinking Practices:

Creating Computational Artifacts
This simulation will again be uploaded to the arcade. This enables others to share and modify each design.

Analyzing Problems and Artifacts
Games should be analyzed on multiple levels, including thinking about the aesthetics as well as pragmatic components of the game. Students should evaluate their own work, as well as their colleagues' work. Students should be expected to justify their work both orally as well as in writing.

For a new simulation, this includes justifying the assumptions used as part of the design.

Communicating
Students should be expected to communicate about their game, both orally and in writing. Communications should be both at the code level, as well as on a macro level, explaining to others the purpose of their game. Language used should be precise and should reflect the computational thinking patterns being learned.

Students will use and describe the following computational thinking patterns:

Reviewed from Frogger/Journey/PacMan

- *Absorb*
- *Generate*
- *Collision*
- *Cursor Control*
- *Diffusion*
- *Polling*
- *Hill Climbing*
- *Perceive/Act*
- *Seeking*

Collaborating
Students working together often achieve more than students working alone. Students can collaborate by working together on solving a particular problem, or by designing the game together. The goal should be creating a high-quality artifact, in a way that highlights the contributions of each student.

This page intentionally left blank.

Appendix A:
Scalable Game Design - AP Computer Science Principles
Complete Standards Matching[5]

[5] The standards used are part of the AP CSP Pilot Project. If standards change, this document will be updated and posted to http://sgd.cs.colorado.edu/wiki/Scalable_Game_Design_wiki

Big Idea 1: Creativity

Computing is a creative activity. Creativity and computing are prominent forces in innovation; the innovations enabled by computing have had and will continue to have far--reaching impact. At the same time, computing facilitates exploration and the creation of computational artifacts and new knowledge that help people solve personal, societal, and global problems. This course emphasizes the creative aspects of computing. Students in this course use the tools and techniques of computer science to create interesting and relevant artifacts with characteristics that are enhanced by computation.

Essential Questions:

- How can a creative development process affect the creation of computational artifacts?
- How can computing and the use of computational tools foster creative expression?
- How can computing extend traditional forms of human expression and experience?

Computational Thinking Practices used:

P2: Creating computational artifacts

P4: Analyzing problems and artifacts

P6: Collaborating

Big Idea 1: Creativity

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this content is met in Scalable Game Design
1.1 Creative development can be an essential process for creating computational artifacts.	**1.1.1** Apply a creative development process when creating computational artifacts. [P2]	**1.1.1A** A creative process in the development of a computational artifact can include, but is not limited to, employing nontraditional, nonprescribed techniques; the use of novel combinations of artifacts, tools, and techniques; and the exploration of personal curiosities. **1.1.1B** Creating computational artifacts employs an iterative and often exploratory process to translate ideas into tangible form.	Students create their own games and later, their own simulations. A creative process is employed throughout the experience. Students will be encouraged to use an iterative process of design, programming and testing.
1.2 Computing enables people to use creative development processes to create computational artifacts for creative expression or to solve a problem.	**1.2.1** Create a computational artifact for creative expression. [P2]	**1.2.1A** A computational artifact is anything created by a human using a computer and can be, but is not limited to, a program, an image, audio, video, a presentation, or a web page file. **1.2.1B** Creating computational artifacts requires understanding and using software tools and services. **1.2.1C** Computing tools and techniques are used to create computational artifacts and can include, but are not limited to, programming IDEs, spreadsheets, 3D printers, or text editors. **1.2.1D** A creatively developed computational artifact can be created by using nontraditional, nonprescribed computing techniques. **1.2.1E** Creative expressions in a computational artifact can reflect personal expressions of ideas or interests.	Creation of a game using the Scalable Game Design Curriculum requires understanding and using the AgentSheets/AgentCubes software. Artifacts in the form of games and/or simulations are created. Students are encouraged to make their games unique, reflecting their own life experiences

SGD and CSP (Continued)

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this content is met in Scalable Game Design
1.2 Computing enables people to use creative development processes to create computational artifacts for creative expression or to solve a problem. ***(Continued)***	**1.2.2** Create a computational artifact using computing tools and techniques to solve a problem. [P2]	**1.2.2A** Computing tools and techniques can enhance the process of finding a solution to a problem. **1.2.2B** A creative development process for creating computational artifacts can be used to solve problems when traditional or prescribed computing techniques are not effective.	Creating science simulations using the tools in SGD enables students to find a solution to a problem. Using the Contagion curricular materials, students learn to explore and model an outbreak of a virus.
	1.2.3 Create a new computational artifact by combining or modifying existing artifacts. [P2]	**1.2.3A** Creating computational artifacts can be done by combining and modifying existing artifacts or by creating new artifacts. **1.2.3B** Computation facilitates the creation and modification of computational artifacts with enhanced detail and precision. **1.2.3C** Combining or modifying existing artifacts can show personal expression of ideas.	Students can learn more about programming by exploring games and simulations created by others. Games and simulations are uploaded to the arcade, enabling students to later download and modify these designs.
	1.2.4 Collaborate in the creation of computational artifacts. [P6]	**1.2.4A** A collaboratively created computational artifact reflects effort by more than one person. **1.2.4B** Effective collaborative teams consider the use of online collaborative tools. **1.2.4C** Effective collaborative teams practice interpersonal communication, consensus building, conflict resolution, and negotiation. **1.2.4D** Effective collaboration strategies enhance performance. **1.2.4E** Collaboration facilitates the application of multiple perspectives (including sociocultural perspectives) and diverse talents and skills in developing computational artifacts. **1.2.4F** A collaboratively created computational artifact can reflect personal expressions of ideas.	Students are encouraged to work together to design games and simulations. Students can form teams to find solutions through modeling real world situations. By asking students to justify their solutions, students learn to build consensus, and negotiate their designs.

SGD and CSP (Continued)

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this content is met in Scalable Game Design
1.2 Computing enables people to use creative development processes to create computational artifacts for creative expression or to solve a problem. ***(Continued)***	**1.2.5** Analyze the correctness, usability, functionality, and suitability of computational artifacts. [P4]	**1.2.5A** The context in which an artifact is used determines the correctness, usability, functionality, and suitability of the artifact. **1.2.5B** A computational artifact may have weaknesses, mistakes, or errors depending on the type of artifact. **1.2.5C** The functionality of a computational artifact may be related to how it is used or perceived. **1.2.5D** The suitability (or appropriateness) of a computational artifact may be related to how it is used or perceived.	Students are encouraged to look at and assess other students' work. These assessments assist the owners of the design, helping them to find issues and use an iterative process to refine their designs.
1.3 Computing can extend traditional forms of human expression and experience.	**1.3.1** Use computing tools and techniques for creative expression. [P2]	**1.3.1A** Creating digital effects, images, audio, video, and animations has transformed industries. **1.3.1B** Digital audio and music can be created by synthesizing sounds, sampling existing audio and music, and recording and manipulating sounds, including layering and looping. **1.3.1C** Digital images can be created by generating pixel patterns, manipulating existing digital images, or combining images. **1.3.1D** Digital effects and animations can be created by using existing software or modified software that includes functionality to implement the effects and animations. **1.3.1E** Computing enables creative exploration of both real and virtual phenomena.	Through the creation of a game, students create agents by generating pixel patterns, manipulating digital images or combining techniques. Students can also create animations through the use of the software, which enables exploration of both real and virtual phenomena.

Big Idea 2: Abstraction

Abstraction reduces information and detail to facilitate focus on relevant concepts. Everyone uses abstraction on a daily basis to effectively manage complexity. In computer science, abstraction is a central problem--solving technique. It is a process, a strategy, and the result of reducing detail to focus on concepts relevant to understanding and solving problems. This course includes examples of abstractions used in modeling the world, managing complexity, and communicating with people as well as with machines. Students in this course learn to work with multiple levels of abstraction while engaging with computational problems and systems; use models and simulations that simplify complex topics in graphical, textual, and tabular formats; and use snapshots of models and simulation outputs to understand how data is changing, identify patterns, and recognize abstractions.

Essential Questions:

- How are vastly different kinds of data, physical phenomena, and mathematical concepts represented on a computer?
- How does abstraction help us in writing programs, creating computational artifacts, and solving problems?
- How can computational models and simulations help generate new understanding and knowledge?

Computational Thinking Practices used:

P2: Creating computational artifacts

P3: Abstracting

P5: Communicating

Big Idea 2: Abstraction

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this objective is met through Scalable Game Design
2.1 A variety of abstractions built upon binary sequences can be used to represent all digital data.	**2.1.1** Describe the variety of abstractions used to represent data. [P3] **Exclusion Statement (2.1.1E):** Two's complement conversions are beyond the scope of this course and the AP Exam.	**2.1.1A** Digital data is represented by abstractions at different levels. **2.1.1B** At the lowest level, all digital data are represented by bits. **2.1.1C** At a higher level, bits are grouped to represent abstractions, including but not limited to numbers, characters, and color. **2.1.1D** Number bases, including binary, decimal, and hexadecimal, are used to represent and investigate digital data. **2.1.1E** At one of the lowest levels of abstraction, digital data is represented in binary (base 2) using only combinations of the digits zero and one. **2.1.1F** Hexadecimal (base 16) is used to represent digital data because hexadecimal representation uses fewer digits than binary. **2.1.1G** Numbers can be converted from any base to any other base.	This information is not explicitly covered by the SGD curriculum. However, the Incrementing Numbers simulation would support the investigation of various base number systems.
	2.1.2 Explain how binary sequences are used to represent digital data. [P5] **Exclusion Statement (2.1.2A):** Binary representations of scientific notation are beyond the scope of this course and the AP Exam. **Exclusion Statement (2.1.2B):** Range limitations of any one language, compiler, or architecture are beyond the scope of this course and the AP Exam.	**2.1.2A** A finite representation is used to model the infinite mathematical concept of a number. **2.1.2B** In many programming languages, the fixed number of bits used to represent characters or integers limits the range of integer values and mathematical operations; this limitation can result in overflow or other errors. **2.1.2C** In many programming languages, the fixed number of bits used to represent real numbers (as floating--point numbers) limits the range of floating-- point values and mathematical operations; this limitation can result in round--off and other errors. **2.1.2D** The interpretation of a binary sequence depends on how it is used. **2.1.2E** A sequence of bits may represent instructions or data. **2.1.2F** A sequence of bits may represent different types of data in different contexts.	This information is not covered by the Scalable Game Design curricular materials.

2.2 Multiple levels of abstraction are used to write programs or create other computational artifacts	**2.2.1** Develop an abstraction when writing a program or creating other computational artifacts. [P2] **Exclusion Statement (2.2.1C): An understanding of the difference between value and reference parameters is beyond the scope of this course and the AP Exam.**	**2.2.1A** The process of developing an abstraction involves removing detail and generalizing functionality. **2.2.1B** An abstraction extracts common features from specific examples in order to generalize concepts. **2.2.1C** An abstraction generalizes functionality with input parameters that allow software reuse.	The Contagion Simulation curricular materials have students generalize from specific examples of the spreading of viruses to a more generalized view. Input parameters are used to enable students to see the effect of varying parameters.
	2.2.2 Use multiple levels of abstraction to write programs. [P3]	**2.2.2A** Software is developed using multiple levels of abstractions, such as constants, expressions, statements, procedures, and libraries. **2.2.2B** Being aware of and using multiple levels of abstraction in developing programs helps to more effectively apply available resources and tools to solve problems.	The Contagion Simulation curricular materials have students use input parameters, constants and expressions to enable students to see how these values can increase the complexity of the simulation design.

2.2 Multiple levels of abstraction are used to write programs or create other computational artifacts (Continued)	**2.2.3** Identify multiple levels of abstractions that are used when writing programs. [P3] **Exclusion Statement (2.2.3A):** Knowledge of the abstraction capabilities of all programming languages is beyond the scope of this course and the AP Exam. **Exclusion Statement (2.2.3F):** Memorization of specific gate visual representations is beyond the scope of this course and the AP Exam.	**2.2.3A** Different programming languages offer different levels of abstraction. **2.2.3B** High--level programming languages provide more abstractions for the programmer and make it easier for people to read and write a program. **2.2.3C** Code in a programming language is often translated into code in another (lower--level) language to be executed on a computer. **2.2.3D** In an abstraction hierarchy, higher levels of abstraction (the most general concepts) would be placed toward the top and lower--level abstractions (the more specific concepts) toward the bottom. **2.2.3E** Binary data is processed by physical layers of computing hardware, including gates, chips, and components. **2.2.3F** A logic gate is a hardware abstraction that is modeled by a Boolean function. **2.2.3G** A chip is an abstraction composed of low-- level components and circuits that perform a specific function. **2.2.3H** A hardware component can be low level like a transistor or high level like a video card. **2.2.3I** Hardware is built using multiple levels of abstractions, such as transistors, logic gates, chips, memory, motherboards, special purposes cards, and storage devices. **2.2.3J** Applications and systems are designed, developed, and analyzed using levels of hardware, software, and conceptual abstractions. **2.2.3K** Lower--level abstractions can be combined to make higher--level abstractions, such as short message services (SMS) or email messages, images, audio files, and videos.	This information is not covered by the Scalable Game Design curricular materials.

2.3 Models and simulations use abstraction to generate new understanding and knowledge.	**2.3.1** Use models and simulations to represent phenomena. [P3]	**2.3.1A** Models and simulations are simplified representations of more complex objects or phenomena. **2.3.1B** Models may use different abstractions or levels of abstraction depending on the objects or phenomena being posed. **2.3.1C** Models often omit unnecessary features of the objects or phenomena that are being modeled. **2.3.1D** Simulations mimic real--world events without the cost or danger of building and testing the phenomena in the real world.	The Contagion Simulation curricular materials have students use input parameters, constants and expressions to enable students to see how these values can increase the complexity of the simulation design.
	2.3.2 Use models and simulations to formulate, refine, and test hypotheses. [P3]	**2.3.2A** Models and simulations facilitate the formulation and refinement of hypotheses related to the objects or phenomena under consideration. **2.3.2B** Hypotheses are formulated to explain the objects or phenomena being modeled. **2.3.2C** Hypotheses are refined by examining the insights that models and simulations provide into the objects or phenomena. **2.3.2D** The results of simulations may generate new knowledge and new hypotheses related to the phenomena being modeled. **2.3.2E** Simulations allow hypotheses to be tested without the constraints of the real world. **2.3.2F** Simulations can facilitate extensive and rapid testing of models. **2.3.2G** The time required for simulations is impacted by the level of detail and quality of the models, and the software and hardware used for the simulation. **2.3.2H** Rapid and extensive testing allows models to be changed to accurately reflect the objects or phenomena being modeled.	The Contagion Simulation curricular materials have students use input parameters, constants and expressions to enable students to see how these values can increase the complexity of the simulation design. Students are encouraged to hypothesize about the effects of changing parameters. The simulation enables students to then test those hypotheses.

Big Idea 3: Data and Information

Data and information facilitate the creation of knowledge. Computing enables and empowers new methods of information processing, driving monumental change across many disciplines - from art to business to science. Managing and interpreting an overwhelming amount of raw data is part of the foundation of our information society and economy. People use computers and computation to translate, process, and visualize raw data, and to create information. Computation and computer science facilitate and enable a new understanding of data and information that contributes knowledge to the world. Students in this course work with data using a variety of computational tools and techniques to better understand the many ways in which data is transformed into information and knowledge.

Essential Questions:

- How can computation be employed to help people process data and information to gain insight and knowledge?
- How can computation be employed to facilitate exploration and discovery when working with data?
- What considerations and trade--offs arise in the computational manipulation of data?
- What opportunities do large data sets provide for solving problems and creating knowledge?

Computational Thinking Practices used:

P1: Connecting Computing

P3: Abstracting

P4: Analyzing problems and artifacts

P5: Communicating

P6: Collaborating

Big Idea 3: Data and Information

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this criteria is met
3.1 People use computer programs to process information to gain insight and knowledge.	**3.1.1** Use computers to process information, find patterns, and test hypotheses about digitally processed information to gain insight and knowledge. [P4]	**3.1.1A** Computers are used in an iterative and interactive way when processing digital information to gain insight and knowledge. **3.1.1B** Digital information can be filtered and cleaned by using computers to process information. **3.1.1C** Combining data sources, clustering data, and data classification are part of the process of using computers to process information. **3.1.1D** Insight and knowledge can be obtained from translating and transforming digitally represented information. **3.1.1E** Patterns can emerge when data is transformed using computational tools.	Teachers could use the available Predators and Prey simulation already available with AgentSheets as a backdrop to all of these activities.
	3.1.2 Collaborate when processing information to gain insight and knowledge. [P6]	**3.1.2A** Collaboration is an important part of solving data--driven problems. **3.1.2B** Collaboration facilitates solving computational problems by applying multiple perspectives, experiences, and skill sets. **3.1.2C** Communication between participants working on data--driven problems gives rise to enhanced insights and knowledge. **3.1.2D** Collaboration in developing hypotheses and questions, and in testing hypotheses and answering questions, about data helps participants gain insight and knowledge. **3.1.2E** Collaborating face--to--face and using online collaborative tools can facilitate processing information to gain insight and knowledge. **3.1.2F** Investigating large data sets collaboratively can lead to insight and knowledge not obtained when working alone.	Teachers could use the available Predators and Prey simulation already available with AgentSheets as a backdrop to all of these activities.

SGD and CSP (Continued)

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this criteria is met
3.1 People use computer programs to process information to gain insight and knowledge. (Continued)	**3.1.3** Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notations, and precise language. [P5]	**3.1.3A** Visualization tools and software can communicate information about data. **3.1.3B** Tables, diagrams, and textual displays can be used in communicating insight and knowledge gained from data. **3.1.3C** Summaries of data analyzed computationally can be effective in communicating insight and knowledge gained from digitally represented information. **3.1.3D** Transforming information can be effective in communicating knowledge gained from data. **3.1.3E** Interactivity with data is an aspect of communicating.	Teachers could use the available Predators and Prey simulation already available with AgentSheets as a backdrop to all of these activities.
3.2 Computing facilitates exploration and the discovery of connections in information.	**3.2.1** Extract information from data to discover and explain connections, patterns, or trends. [P1] **Exclusion Statement (3.2.1F):** Students are not expected to know specific formulas or options available in spreadsheet or database software packages..	**3.2.1A** Large data sets provide opportunities and challenges for extracting information and knowledge. **3.2.1B** Large data sets provide opportunities for identifying trends, making connections in data, and solving problems. **3.2.1C** Computing tools facilitate the discovery of connections in information within large data sets. **3.2.1D** Search tools are essential for efficiently finding information. **3.2.1E** Information filtering systems are important tools for finding information and recognizing patterns in the information. **3.2.1F** Software tools, including spreadsheets and databases, help to efficiently organize and find trends in information. **3.2.1G** Metadata is data about data. **3.2.1H** Metadata can be descriptive data about an image, a Web page, or other complex objects. **3.2.1I** Metadata can increase the effective use of data or data sets by providing additional information about various aspects of that data.	Teachers could use the available Predators and Prey simulation already available with AgentSheets as a backdrop to all of these activities.

SGD and CSP (Continued)

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this criteria is met
3.2 Computing facilitates exploration and the discovery of connections in information. (Continued)	**3.2.2.** Use large data sets to explore and discover information and knowledge. [P3]	**3.2.2A** Large data sets include data such as transactions, measurements, text, sound, images, and video. **3.2.2B** The storing, processing, and curating of large data sets is challenging. **3.2.2C** Structuring large data sets for analysis can be challenging. **3.2.2D** Maintaining privacy of large data sets containing personal information can be challenging. **3.2.2E** Scalability of systems is an important consideration when data sets are large. **3.2.2F** The size or scale of a system that stores data affects how that data set is used. **3.2.2G** The effective use of large data sets requires computational solutions. **3.2.2H** Analytical techniques to store, manage, transmit, and process data sets change as the size of data sets scale.	Teachers could use the available Predators and Prey simulation already available with AgentSheets as a backdrop for all of these activities.

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this criteria is met
3.3 There are trade--offs when representing information as digital data.	**3.3.1** Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information. [P4]	**3.3.1A** Digital data representations involve trade-- offs related to storage, security, and privacy concerns. **3.3.1B** Security concerns engender trade--offs in storing and transmitting information. **3.3.1C** There are trade-offs in using lossy and lossless compression techniques for storing and transmitting data. **3.3.1D** Lossless data compression reduces the number of bits stored or transmitted but allows complete reconstruction of the original data. **3.3.1E** Loss of data compression can significantly reduce the number of bits stored or transmitted at the cost of being able to reconstruct only an approximation of the original data. **3.3.1F** Security and privacy concerns arise with data containing personal information. **3.3.1G** Data is stored in many formats depending on its characteristics (e.g., size and intended use). **3.3.1H** The choice of storage media affects both the methods and costs of manipulating the data it contains. **3.3.1I** Reading data and updating data have different storage requirements.	This information is not covered by the Scalable Game Design curricular materials.

Big Idea 4: Algorithms

Algorithms are used to develop and express solutions to computational problems. Algorithms are fundamental to even the most basic everyday task. Algorithms realized in software have affected the world in profound and lasting ways. Secure data transmission and quick access to large amounts of relevant information are made possible through the implementation of algorithms. The development, use, and analysis of algorithms are some of the most fundamental aspects of computing. Students in this course work with algorithms in many ways: they develop and express original algorithms, they implement algorithms in a language, and they analyze algorithms analytically and empirically.

Essential Questions:

- How are algorithms implemented and executed on computers and computational devices?
- Why are some languages better than others when used to implement algorithms?
- What kinds of problems are easy, what kinds are difficult, and what kinds are impossible to solve algorithmically?
- How are algorithms evaluated?

Computational Thinking Practices used:

P1: Connecting Computing

P2: Creating computational artifacts

P4: Analyzing problems and artifacts

P5: Communicating

Big Idea 4: Algorithms

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this content is met
4.1 Algorithms are precise sequences of instructions for processes that can be executed by a computer and are implemented using programming languages.	**4.1.1** Develop an algorithm for implementation in a program. [P2]	**4.1.1A** Sequencing, selection, and iteration are building blocks of algorithms. **4.1.1B** Sequencing is the application of each step of an algorithm in the order in which the statements are given. **4.1.1C** Selection uses a Boolean condition to determine which of two parts of an algorithm is used. **4.1.1D** Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times. **4.1.1E** Algorithms can be combined to make new algorithms. **4.1.1F** Using existing correct algorithms as building blocks for constructing a new algorithm helps ensure the new algorithm is correct. **4.1.1G** Knowledge of standard algorithms can help in constructing new algorithms. **4.1.1H** Different algorithms can be developed to solve the same problem. **4.1.1I** Developing a new algorithm to solve a problem can yield insight into the problem.	As part of designing a game in SGD, students will learn how to sequence steps, use Boolean conditions, create iterations within their program, and create basic and more complex algorithms. Students will also use their existing knowledge of algorithms to support them in creating new algorithms. An emphasis on the multiplicity of solutions is always encouraged, and conversations between students to understand why different solutions may lead to new insights is a critical aspect of collaboration.

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this content is met
4.1 Algorithms are precise sequences of instructions for processes that can be executed by a computer and are implemented using programming languages. (Continued)	**4.1.2** Express an algorithm in a language. [P5]	**4.1.2A** Languages for algorithms include natural language, pseudocode, and visual and textual programming languages. **4.1.2B** Natural language and pseudocode describe algorithms so that humans can understand them. **4.1.2C** Algorithms described in programming languages can be executed on a computer. **4.1.2D** Different languages are better suited for expressing different algorithms. **4.1.2E** Some programming languages are designed for specific domains and are better for expressing algorithms in those domains. **4.1.2F** The language used to express an algorithm can affect characteristics such as clarity or readability but not whether an algorithmic solution exists. **4.1.2G** Every algorithm can be constructed using only sequencing, selection, and iteration. **4.1.2H** Nearly all programming languages are equivalent in terms of being able to express any algorithm. **4.1.2I** Clarity and readability are important considerations when expressing an algorithm in a language.	The software used in Scalable Game Design enables students to see the code expressed in natural language (through sentences), as a visual language, and then as the java code behind the scenes. The software also facilitates students ability to comment their code, so that others can follow their thought process.

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this content is met
4.2 Algorithms can solve many but not all computational problems.	**4.2.1** Explain the difference between algorithms that run in a reasonable time and those that do not run in a reasonable time. [P1] **Exclusion Statement (LO 4.2.1):** Any discussion of nondeterministic polynomial (NP) is beyond the scope of this course and the AP Exam.	**4.2.1A** Many problems can be solved in a reasonable time. **4.2.1B** Reasonable time means that as the input size grows, the number of steps the algorithm takes is proportional to the square (or cube, fourth power, fifth power, etc.) of the size of the input. **4.2.1C** Some problems cannot be solved in a reasonable time, even for small input sizes. **4.2.1D** Some problems can be solved but not in a reasonable time. In these cases, heuristic approaches may be helpful to find solutions in reasonable time.	Simulations (either existing ones provided as part of the software, or simulations created by students) could serve as a backdrop for these discussions.
	4.2.2 Explain the difference between solvable and unsolvable problems in computer science. [P1] Exclusion Statement (4.2.2B): Specific heuristic solutions are beyond the scope of this course and the AP Exam. **Exclusion Statement (LO 4.2.2):** Determining whether a given problem is solvable or unsolvable is beyond the scope of this course and the AP Exam.	**4.2.2A** A heuristic is a technique that may allow us to find an approximate solution when typical methods fail to find an exact solution. **4.2.2B** Heuristics may be helpful for finding an approximate solution more quickly when exact methods are too slow. **4.2.2C** Some optimization problems such as "find the best" or "find the smallest" cannot be solved in a reasonable time, but approximations to the optimal solution can. **4.2.2D** Some problems cannot be solved using any algorithm.	Simulations (either existing ones provided as part of the software, or simulations created by students) could serve as a backdrop for these discussions.

SGD and CSP (Continued)

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this content is met
4.2 Algorithms can solve many but not all computational problems. (Continued)	**4.2.3** Explain the existence of undecidable problems in computer science. [P1] **Exclusion Statement (4.2.3C):** Determining whether a given problem is undecidable is beyond the scope of this course and the AP Exam.	**4.2.3A** An undecidable problem may have instances that have an algorithmic solution, but there is no algorithmic solution that solves all instances of the problem. **4.2.3B** A decidable problem is one in which an algorithm can be constructed to answer "yes" or "no" for all inputs (e.g., "is the number even?"). **4.2.3C** An undecidable problem is one in which no algorithm can be constructed that always leads to a correct yes--or--no answer.	This information is not specifically covered by the Scalable Game Design curricular materials.
	4.2.4 Evaluate algorithms analytically and empirically for efficiency, correctness, and clarity. [P4] **Exclusion statement (4.2.4C):** Formally proving program correctness is beyond the scope of this course and the AP Exam. **Exclusion Statement (4.2.4G):** Formal analysis of algorithms (Big--O) and formal reasoning using mathematical formulas are beyond the scope of this course and the AP Exam.	**4.2.4A** Determining an algorithm's efficiency is done by reasoning formally or mathematically about the algorithm. **4.2.4B** Empirical analysis of an algorithm is done by implementing the algorithm and running it on different inputs. **4.2.4C** The correctness of an algorithm is determined by reasoning formally or mathematically about the algorithm, not by testing an implementation of the algorithm. **4.2.4D** Different correct algorithms for the same problem can have different efficiencies. **4.2.4E** Sometimes more efficient algorithms are more complex. **4.2.4F** Finding an efficient algorithm for a problem can help solve larger instances of the problem. **4.2.4G** Efficiency includes both execution time and memory usage. **4.2.4H** Linear search can be used when searching for an item in any list; binary search can be used only when the list is sorted.	By comparing different methods of game design through the Scalable Game Design curriculum, students can work toward understanding the correctness of an algorithm as well as its efficiency.

Big Idea 5: Programming

Programming enables problem solving, human expression, and creation of knowledge.

Programming and the creation of software has changed our lives. Programming results in the creation of software, and it facilitates the creation of computational artifacts, including music, images, and visualizations. In this course, programming enables exploration and is the object of study. This course introduces students to the concepts and techniques related to writing programs, developing software, and using software effectively; the focus of the course is not on programming per se but on all aspects of computation. Students in this course create programs, translating human intention into computational artifacts.

Essential Questions:

- How are programs developed to help people, organizations, or society solve problems?
- How are programs used for creative expression, to satisfy personal curiosity, or to create new knowledge?
- How do computer programs implement algorithms?
- How does abstraction make the development of computer programs possible?
- How do people develop and test computer programs?
- Which mathematical and logical concepts are fundamental to computer programming?

Computational Thinking Practices used:

P1: Connecting Computing

P3: Abstracting

P4: Analyzing problems and artifacts

P5: Collaborating

Big Idea 5: Programming

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this content is met in Scalable Game Design
5.1 Programs can be developed for creative expression, to satisfy personal curiosity, to create new knowledge, or to solve problems (to help people, organizations, or society).	**5.1.1** Develop a program for creative expression, to satisfy personal curiosity, or to create new knowledge. [P2]	**5.1.1A** Programs are developed and used in a variety of ways by a wide range of people depending on the goals of the programmer. **5.1.1B** Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may have visual, audible, or tactile inputs and outputs. **5.1.1C** Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may be developed with different standards or methods than programs developed for widespread distribution. **5.1.1D** Additional desired outcomes may be realized independently of the original purpose of the program. **5.1.1E** A computer program or the results of running a program may be rapidly shared with a large number of users and can have widespread impact on individuals, organizations, and society. **5.1.1F** Advances in computing have generated and increased creativity in other fields.	These are general concepts that would make excellent discussions at the start of each day using the Scalable Game Design curriculum.

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this content is met in Scalable Game Design
5.1 Programs can be developed for creative expression, to satisfy personal curiosity, to create new knowledge, or to solve problems (to help people, organizations, or society). (Continued)	**5.1.2** Develop a correct program to solve problems. [P2]	**5.1.2A** An iterative process of program development helps in developing a correct program to solve problems. **5.1.2B** Developing correct program components and then combining them helps in creating correct programs. **5.1.2C** Incrementally adding tested program segments to correct, working programs helps create large correct programs. **5.1.2D** Program documentation helps programmers develop and maintain correct programs to efficiently solve problems. **5.1.2E** Documentation about program components, such as blocks and procedures, helps in developing and maintaining programs. **5.1.2F** Documentation helps in developing and maintaining programs when working individually or in collaborative programming environments. **5.1.2G** Program development includes identifying programmer and user concerns that affect the solution to problems. **5.1.2H** Consultation and communication with program users is an important aspect of program development to solve problems. **5.1.2I** A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem. **5.1.2J** A programmer designs, implements, tests, debugs, and maintains programs when solving problems.	All of these components are already part of the Scalable Game Design guided discovery curricular materials. Students will learn to use an iterative process that includes design and testing. The software enables students to comment throughout their code, providing documentation.

SGD and CSP (Continued)

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this content is met in Scalable Game Design
5.1 Programs can be developed for creative expression, to satisfy personal curiosity, to create new knowledge, or to solve problems (to help people, organizations, or society). (Continued)	**5.1.3** Collaborate to develop a program. [P6]	**5.1.3A** Collaboration can decrease the size and complexity of tasks required of individual programmers. **5.1.3B** Collaboration facilitates multiple perspectives in developing ideas for solving problems by programming. **5.1.3C** Collaboration in the iterative development of a program requires different skills than developing a program alone. **5.1.3D** Collaboration can make it easier to find and correct errors when developing programs. **5.1.3E** Collaboration facilitates developing program components independently. **5.1.3F** Effective communication between participants is required for successful collaboration when developing programs.	Using collaborative teams is an easy way to build this skill, particularly in the design of simulations where different experiences and skills are helpful to the design process.
5.2 People write programs to execute algorithms.	**5.2.1** Explain how programs implement algorithms. [P3]	**5.2.1A** Algorithms are implemented using program instructions that are processed during program execution. **5.2.1B** Program instructions are executed sequentially. **5.2.1C** Program instructions may involve variables that are initialized and updated, read, and written. **5.2.1D** An understanding of instruction processing and program execution is useful for programming. **5.2.1E** Program execution automates processes. **5.2.1F** Processes use memory, a central processing unit (CPU), and input and output. **5.2.1G** A process may execute by itself or with other processes. **5.2.1H** A process may execute on one or several CPUs. **5.2.1I** Executable programs increase the scale of problems that can be addressed. **5.2.1J** Simple algorithms can solve a large set of problems when automated. **5.2.1K** Improvements in algorithms, hardware, and software increase the kinds of problems and the size of problems solvable by programming.	Discussing how other programs are designed, as well as possible underlying algorithms would make for an excellent extension of game design activities.

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this content is met in Scalable Game Design
5.3 Programming is facilitated by appropriate abstractions.	**5.3.1** Use abstraction to manage complexity in programs. [P3]	**5.3.1A** Procedures are reusable programming abstractions. **5.3.1B** A function is a named grouping of programming instructions. **5.3.1C** Procedures reduce the complexity of writing and maintaining programs. **5.3.1D** Procedures have names and may have parameters and return values. **5.3.1E** Parameterization can generalize a specific solution. **5.3.1F** Parameters generalize a solution by allowing a function to be used instead of duplicated code. **5.3.1G** Parameters provide different values as input to procedures when they are called in a program. **5.3.1H** Data abstraction provides a means of separating behavior from implementation. **5.3.1I** Strings and string operations, including concatenation and some form of substring, are common in many programs. **5.3.1J** Integers and floating--point numbers are used in programs without requiring understanding of how they are implemented. **5.3.1K** Lists and list operations, such as add, remove, and search, are common in many programs. **5.3.1L** Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain. **5.3.1M** Application program interfaces (APIs) and libraries simplify complex programming tasks. **5.3.1N** Documentation for an API/library is an important aspect of programming. **5.3.1O** APIs connect software components, allowing them to communicate	All games and simulations past Frogger make sure of called procedures (methods) to facilitate programming. They also use different input values, including strings and integers. APIs are not covered as part of the Scalable Game Design curriculum, but could be incorporated through an extension using simulations where data is exported into a spreadsheet.

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this content is met in Scalable Game Design
5.4 Programs are developed, maintained, and used by people for different purposes.	**5.4.1** Evaluate the correctness of a program. [P4]	**5.4.1A** Program style can affect the determination of program correctness. **5.4.1B** Duplicated code can make it harder to reason about a program. **5.4.1C** Meaningful names for variables and procedures help people better understand programs. **5.4.1D** Longer code blocks are harder to reason about than shorter code blocks in a program. **5.4.1E** Locating and correcting errors in a program is called debugging the program. **5.4.1F** Knowledge of what a program is supposed to do is required in order to find most program errors. **5.4.1G** Examples of intended behavior on specific inputs help people understand what a program is supposed to do. **5.4.1H** Visual displays (or different modalities) of program state can help in finding errors. **5.4.1I** Programmers justify and explain a program's correctness. **5.4.1J** Justification can include a written explanation about how a program meets its specifications. **5.4.1K** Correctness of a program depends on correctness of program components, including code blocks and procedures. **5.4.1L** An explanation of a program helps people understand the functionality and purpose of it. **5.4.1M** The functionality of a program is often described by how a user interacts with it. **5.4.1N** The functionality of a program is best described at a high level by what the program does, not at the lower level of how the program statements work to accomplish this.	Multiple methods of testing and debugging programs are built into the Scalable Game Design guided discovery curricular materials. Students are encouraged to work together to test each others' designs. Many teachers incorporate a final write-up of the program to explain its' functions and its' design components.

Big Idea 6: The Internet

The Internet pervades modern computing. The Internet and the systems built on it have had a profound impact on society. Computer networks support communication and collaboration. The principles of systems and networks that helped enable the Internet are also critical in the implementation of computational solutions. Students in this course gain insight into how the Internet operates, study characteristics of the Internet and systems built upon it, and analyze important concerns such as cybersecurity.

Essential Questions:

- What is the Internet? How is it built? How does it function?
- What aspects of the Internet's design and development have helped it scale and flourish?
- How is cybersecurity impacting the ever--increasing number of Internet users?

Computational Thinking Practices used:

P1: Connecting Computing

P3: Abstracting

P4: Analyzing problems and artifacts

P5: Communicating

Big Idea 6: The Internet

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this content is covered
6.1 The Internet is a network of autonomous systems.	**6.1.1** Explain the abstractions in the Internet and how the Internet functions. [P3] **Exclusion Statement (LO 6.1.1):** Specific devices used to implement the abstractions in the Internet are beyond the scope of this course and the AP Exam. **Exclusion Statement (6.1.1F):** Specific details of any particular standard for addresses are beyond the scope of this course and the AP Exam.	**6.1.1A** The Internet connects devices and networks all over the world. **6.1.1B** An end--to--end architecture facilitates connecting new devices and networks on the Internet. **6.1.1C** Devices and networks that make up the Internet are connected and communicate using addresses and protocols. **6.1.1D** The Internet and the systems built on it facilitate collaboration. **6.1.1E** Connecting new devices to the Internet is enabled by assignment of an Internet protocol (IP) address. **6.1.1F** The Internet is built on evolving standards, including those for addresses and names. **6.1.1G** The domain name system (DNS) translates names to IP addresses. **6.1.1H** The number of devices that could use an IP address has grown so fast that a new protocol (IPv6) has been established to handle routing of many more devices. **6.1.1I** Standards such as hypertext transfer protocol (HTTP), IP, and simple mail transfer protocol (SMTP) are developed and overseen by the Internet Engineering Task Force (IETF).	This information is not covered by the Scalable Game Design curricular materials.

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this content is covered
6.2 Characteristics of the Internet influence the systems built on it.	**6.2.1** Explain characteristics of the Internet and the systems built on it. [P5]	**6.2.1A** The Internet and the systems built on it are hierarchical and redundant. **6.2.1B** The domain name syntax is hierarchical. **6.2.1C** IP addresses are hierarchical. **6.2.1D** Routing on the Internet is fault tolerant and redundant.	This information is not covered by the Scalable Game Design curricular materials.
	6.2.2 Explain how the characteristics of the Internet influence the systems built on it. [P4] **Exclusion Statements (6.2.2F):** Specific details of any particular packet-- switching system are beyond the scope of this course and the AP Exam. **Exclusion Statement (6.2.2G):** Specific technical details of how TCP/IP works are beyond the scope of this course and the AP Exam. **Exclusion Statement (6.2.2H):** Understanding the technical aspects of how SSL/TLS works is beyond the scope of this course and the AP Exam.	**6.2.2A** Hierarchy and redundancy help systems scale. **6.2.2B** The redundancy of routing (i.e., more than one way to route data) between two points on the Internet increases the reliability of the Internet and helps it scale to more devices and more people. **6.2.2C** Hierarchy in the DNS helps that system scale. **6.2.2D** Interfaces and protocols enable widespread use of the Internet. **6.2.2E** Open standards fuel the growth of the Internet. **6.2.2F** The Internet is a packet--switched system through which digital data is sent by breaking the data into blocks of bits called packets, which contain both the data being transmitted and control information for routing the data. **6.2.2G** Standards for packets and routing include transmission control protocol/Internet protocol (TCP/IP). **6.2.2H** Standards for sharing information and communicating between browsers and servers on the Web include HTTP and secure sockets layer/transport layer security (SSL/TLS). **6.2.2I** The size and speed of systems affect their use. **6.2.2J** The bandwidth of a system is a measure of bit rate - the amount of data (measured in bits) that can be sent in a fixed amount of time. **6.2.2K** The latency of a system is the time elapsed between the transmission and the receipt of a request.	This information is not covered by the Scalable Game Design curricular materials.

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this content is covered
6.3 Cybersecurity is an important concern for the Internet and the systems built on it.	**6.3.1** Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it. [P1] **Exclusion statement (6.3.1I):** Specific mathematical functions used in cryptography are beyond the scope of this course and the AP Exam. **Exclusion Statement (6.3.1K):** The methods used in encryption are beyond the scope of this course and the AP Exam. **Exclusion statement (6.3.1L):** The mathematical methods used in public key encryption are beyond the scope of this course and the AP Exam. **Exclusion statement (6.3.1M):** The technical details of the process certificate authorities follow are beyond the scope of this course and the AP Exam.	**6.3.1A** The trust model of the Internet involves trade--offs. **6.3.1B** The domain name system (DNS) was not designed to be completely secure. **6.3.1C** Implementing cybersecurity has software, hardware, and human components. **6.3.1D** Cyber warfare and cyber crime have widespread and potentially devastating effects. **6.3.1E** Distributed denial--of--service attacks (DDoS) compromise a target by flooding it with requests from multiple systems. **6.3.1F** Phishing, viruses, and other attacks have human and software components. **6.3.1G** Antivirus software and firewalls can help prevent unauthorized access to private data. **6.3.1H** Cryptography is essential to many models of cybersecurity. **6.3.1I** Cryptography has a mathematical foundation. **6.3.1J** Open standards help ensure cryptography is secure. **6.3.1K** Symmetric encryption is a method of encryption involving one key for encryption and decryption. **6.3.1L** Public key encryption, which is not symmetric, is an encryption method that is widely used because of the enhanced security associated with its use. **6.3.1M** Certificate authorities (CAs) issue digital certificates that validate the ownership of encrypted keys used in secured communication and are based on a trust model.	This information is not covered by the Scalable Game Design curricular materials.

Big Idea 7: Global Impact

Computing has global impact. Computation has changed the way people think, work, live, and play. Our methods for communicating, collaborating, problem solving, and doing business have changed and are changing due to innovations enabled by computing. Many innovations in other fields are fostered by advances in computing. Computational approaches lead to new understandings, new discoveries, and new disciplines. Students in this course become familiar with many ways in which computing enables innovation, and they analyze the potential benefits and harmful effects of computing in a number of contexts.

Essential Questions:

- How does computing enhance human communication, interaction, and cognition?
- How does computing enable innovation?
- What are some potential beneficial and harmful effects of computing?
- How do economic, social, and cultural contexts influence innovation and the use of computing?

Big Idea 7: Global Impact

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this content is covered
7.1 Computing enhances communication, interaction, and cognition.	**7.1.1** Explain how computing innovations affect communication, interaction, and cognition. [P4] **Exclusion Statement (7.1.1C):** Detailed knowledge of any social media site is beyond the scope of this course and the AP Exam.	**7.1.1A** Email, short message service (SMS), and chat have fostered new ways to communicate and collaborate. **7.1.1B** Video conferencing and video chat have fostered new ways to communicate and collaborate. **7.1.1C** Social media continues to evolve and foster new ways to communicate. **7.1.1D** Cloud computing fosters new ways to communicate and collaborate. **7.1.1E** Widespread access to information facilitates the identification of problems, development of solutions, and dissemination of results. **7.1.1F** Public data provides widespread access and enables solutions to identified problems. **7.1.1G** Search trends are predictors. **7.1.1H** Social media, such as blogs and Twitter, have enhanced dissemination. **7.1.1I** Global Positioning System (GPS) and related technologies have changed how humans travel, navigate, and find information related to geolocation. **7.1.1J** Sensor networks facilitate new ways of interacting with the environment and with physical systems. **7.1.1K** Smart grids, smart buildings, and smart transportation are changing and facilitating human capabilities. **7.1.1L** Computing contributes to many assistive technologies that enhance human capabilities. **7.1.1M** The Internet and the Web have enhanced methods of and opportunities for communication and collaboration. **7.1.1N** The Internet and the Web have changed many areas, including e--commerce, health care, access to information and entertainment, and online learning. **7.1.1O** The Internet and the Web have impacted productivity, positively and negatively, in many areas.	This information is not covered by the Scalable Game Design curricular materials.

SGD and CSP (Continued)

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this content is covered
	7.1.2 Explain how people participate in a problem--solving process that scales. [P4]	**7.1.2A** Distributed solutions must scale to solve some problems. **7.1.2B** Science has been impacted by using scale and "citizen science" to solve scientific problems using home computers in scientific research. **7.1.2C** Human computation harnesses contributions from many humans to solve problems related to digital data and the Web. **7.1.2D** Human capabilities are enhanced by digitally enabled collaboration. **7.1.2E** Some online services use the contributions of many people to benefit both individuals and society. **7.1.2F** Crowdsourcing offers new models for collaboration, such as connecting people with jobs and businesses with funding. **7.1.2G** The move from desktop computers to a proliferation of always--on mobile computers is leading to new applications.	This information is not covered by the Scalable Game Design curricular materials.
7.2 Computing enables innovation in nearly every field.	**7.2.1** Explain how computing has impacted innovations in other fields. [P1]	**7.2.1A** Machine learning and data mining have enabled innovation in medicine, business, and science. **7.2.1B** Scientific computing has enabled innovation in science and business. **7.2.1C** Computing enables innovation by providing access to and sharing of information. **7.2.1D** Open access and Creative Commons have enabled broad access to digital information. **7.2.1E** Open and curated scientific databases have benefited scientific researchers. **7.2.1F** Moore's law has encouraged industries that use computers to effectively plan future research and development based on anticipated increases in computing power. **7.2.1G** Advances in computing as an enabling technology have generated and increased the creativity in other fields.	This information is not covered by the Scalable Game Design curricular materials.

SGD and CSP (Continued)

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)	How this content is covered
7.3 Computing has a global affect - both beneficial and harmful - on people and society.	**7.3.1** Analyze the beneficial and harmful effects of computing. [P4]	**7.3.1A** Innovations enabled by computing raise legal and ethical concerns. **7.3.1B** Commercial access to music and movie downloads and streaming raises legal and ethical concerns. **7.3.1C** Access to digital content via peer--to--peer networks raises legal and ethical concerns. **7.3.1D** Both authenticated and anonymous access to digital information raise legal and ethical concerns. 7.3.1E Commercial and governmental censorship of digital information raise legal and ethical concerns. 7.3.1F Open source and licensing of software and content raise legal and ethical concerns.	This information is not covered by the Scalable Game Design curricular materials.
7.4 Computing innovations influence and are influenced by the economic, social, and cultural contexts in which they are designed and used	**7.4.1** Explain the connections between computing and economic, social, and cultural contexts. [P1]	**7.4.1A** The innovation and impact of social media and online access is different in different countries and in different socioeconomic groups. **7.4.1B** Mobile, wireless, and networked computing have an impact on innovation throughout the world. **7.4.1C** The global distribution of computing resources raises issues of equity, access, and power. **7.4.1D** Groups and individuals are affected by the "digital divide" - differing access to computing and the Internet based on socioeconomic or geographic characteristics. **7.4.1E** Networks and infrastructure are supported by both commercial and governmental initiatives.	This information is not covered by the Scalable Game Design curricular materials.