Creative Coding Through Games And Apps

About the Course

Copyright Notice

Creative Coding through Games and Apps

> *Over the past few decades, computers have transformed both the world and the workforce in many profound ways. As a result, computer science and the technologies it enables now lie at the heart of our economy and the way we live our lives. To be well-educated citizens in a computing-intensive world and to be prepared for careers in the 21st century, our students must have a clear understanding of the principles and practices of computer science. No other subject will open as many doors in the 21st century as computer science, regardless of a student's ultimate field of study or occupation.*
>
> -CSTA K-12 Computer Science Standards

This document describes *Creative Coding through Games and Apps*, an introductory computer programming course that lays the foundation for understanding the principles and practices of computer science.

Contributors

Sponsored and published by Microsoft, *Creative Coding through Games and Apps* was developed by a group of current and former computer science teachers, instructional designers, and artists.

- **Brian Swan**, Senior Content Developer, Microsoft Learning Experiences Group
- **Pat Phillips**, CS Education Consultant
- **Marisa Vitiello**, Instructional Designer, Sublime Media
- **David Linder**, Creative Director, Sublime Media
- **David Burkhart**, CS and Technology Teacher, Sheridan High School
- **Stephanie Hoeppner**, CS Teacher, Clermont Northeastern High School
- **Andy Kuemmel**, CS Teacher, Madison West High School
- **Tim McMichael**, CIS Faculty at Estrella Mountain Community College

Description

Creative Coding through Games and Apps is a first-semester course for introduction to programming for the early secondary grades. The course is designed to attract and reach a broad and diverse range of students, including those who may have never before considered programming. Students learn how to code by working in a real software development environment to design, program and publish mobile apps and games. Learning to code by creating real products, students discover how to make amazing things and have an impact on their world.

Creative Coding through Games and Apps is designed to be delivered with success by any teacher, regardless of computer science background. Featuring online and in-class lessons that emphasize hands-on coding, the course can be taught via any modern web browser on phones, tablets, laptops or desktop computers. The combination of online plus in-class resources provides flexibility in teaching style, allowing educators to choose the right balance for their students of in-class instruction and out-of-class study. The course length is flexible, deliverable in 6, 9, 12, 18 weeks or more, depending on the time available in your school.

Vision

By developing problem-solving and computational-thinking skills in the context of learning fundamental computer science concepts, *Creative Coding through Games and Apps* sets students of early secondary grades on a journey to become well prepared for life and careers in the contemporary world. At the core of the learning experience in *Creative Coding through Games and Apps* are the beliefs that students invest when they can be creative, that students learn best by doing, and that social engagement can be a powerful accelerant to learning. These beliefs are brought to life by using leading-edge technology to create an interactive and immersive learning experience focused on solving real-world problems that are relevant to students' lives.

Audience

In addition to youngsters who are already motivated and curious about coding, *Creative Coding through Games and Apps* is intended for students in early secondary grades who have never expressed interest in programming, have no previous experience in programming, and/or who may lack confidence in their technical skills. Typically, students who study computer programming are self-motivated students with a strong affinity for mathematics and complex logical reasoning. While *Creative Coding through Games and Apps* will appeal to these students, it is specifically designed to attract a broad student audience through emphasis on creativity, relevance, and real-world results.

For student success in this course, prerequisite knowledge and skill in reading and mathematics equivalent to 7th grade level is advised.

Creative Coding through Games and Apps is intended for teachers who come from any discipline, have an affinity for working with computers, and have basic math skills (9th grade algebra). While most introductory programming courses are taught by teachers with some background in computer science, *Creative Coding through Games and Apps* teachers are not expected to have any such background. Teacher materials for the course allow a new-to-programming teacher to learn ahead of his/her students the first time through the course.

Goals

Creative Coding through Games and Apps is designed to:

- Attract interest in computer science from a broad and diverse range of students
- Teach the fundamentals of programming and computational thinking
- Instill confidence in working with technology
- Foster creativity, curiosity, and collaboration
- Teach how to program computing devices to make things of real-world interest and use

Students who successfully complete this course will be able to:

- Describe what a computer program is
- Use and implement common program control structures
- Read code in the TouchDevelop environment and describe what it does
- Create and publish TouchDevelop apps and games

Please see the Course Outline in this document for comprehensive accounting of student learning objectives.

Learning Experience and Technology Platform

The TouchDevelop instructional programming environment from Microsoft Research serves as the foundation for the *Creative Coding through Games and Apps* student learning experience.

Requiring nothing more than a modern web browser running on any computing platform across phones, tablets, laptops or desktop computers, TouchDevelop includes four features that are essential to the learning experience:

- **Interactive tutorials**. TouchDevelop interactive tutorials allow instructional video to be embedded within the programming environment and for varying levels of guided practice when writing code.
- **Portability**. TouchDevelop runs on any device that has a modern browser. TouchDevelop is designed to enable students to write code on a device that does not have a keyboard (e.g. a mobile phone or a tablet), but also works on a computer with a keyboard and mouse.
- **Ease of use**. TouchDevelop makes it easy to build mobile apps and games. TouchDevelop abstracts much of the complicated engineering that makes games and apps fun and compelling to use, but difficult to create. For example, TouchDevelop has a built-in physics engine that makes it easy to simulate gravity and detect collisions. This feature allows students to learn fundamental programming concepts without having to understand details they may find tedious at this phase of their development.
- **Sociability**. TouchDevelop enables students to easily share and build upon each other's work. A TouchDevelop program can be published with the click of a button. Once a program is published, others can use it and/or build upon its functionality.

Creative Coding through Games and Apps uses TouchDevelop to introduce fundamental programming concepts through a series of interactive lessons that build upon ideas in previous lessons. In many lessons, students are introduced to concepts through video instruction embedded within the programming environment. After being introduced to a concept, students

are guided through writing code that practices the concept. As such lessons progress, guidance is reduced, requiring students write code independently. At the conclusion of many lessons, students are directed to complete unguided projects that reinforce concepts and encourage them to be creative and to explore further.

Creative Coding through Games and Apps introduces concepts within the context of a game, app, or puzzle. For example, when learning about conditionals, students may be asked to complete a partially functioning game by adding the necessary conditional logic to complete the game. As the course progresses, students bring concepts together by creating simple apps and games of their own.

Finally, students build upon each other's work by publishing and sharing their games and apps.

Classroom Resource and Technical Requirements

Students need:

- Student learning materials provided in digital file folders associated with this document
- A computing device (PC, laptop, tablet, or phone) with a modern browser.
- Internet access
- Journal
- Headphones (recommended)

Teachers need:

- Lesson preparation and presentation materials provided in digital file folders associated with this document
- A computer with a modern browser
- Internet access
- An audiovisual projection system

Description of Course Materials

The following materials are included with this course:

- Course overview document
  - Includes day-by-day plans for implementing the curriculum as a 6, 9, 12, or 18-week instructor-led course.
- For each of 12 units:
  - Teacher and student unit-level guides
  - Lesson plans and presentation materials
  - Student learning activities, quizzes, and supplemental resources

The digital files for this course apply a naming convention to organize the curricular unit and lesson structure. Digital file names for this course begin with **U<x>.<y>_<file type>**, where

<x> = unit number

<y> = lesson number, and

<file type> = the type of curriculum resource provided by the given file.

For example, you will find the following documents in the Unit 1 folder (among others):

- **U1.02_Lesson**: The *lesson plan* for Lesson 2 of Unit 1.
- **U1.02_Activity1**: A *student activity worksheet* for Lesson 2 of Unit 1.
- **U1.02_Activity2**: A *second student activity worksheet* for Lesson 2 of Unit 1.
- **U1.02_Resource**: A *student-facing resource* for Lesson 2 of Unit 1, meant to be kept as a reference. **_U1_Teacher_Guide**: A *teacher-facing unit-level overview* of Unit 1.
- **_U1_Student_Guide**: A *student-facing unit-level overview* of Unit 1.

For more details about these documents and the other types of documents found for each unit, see the table below.

Bill of Materials

Document Type	Description
_Teacher_Guide	This document is the "unit in a nutshell" for the teacher. It includes the following information: • A high-level overview of the unit • A list of "big" questions that should be answered in the unit • The unit schema, which includes a lesson outline, student learning objectives, and required resources for each lesson • Required preparation for the unit • Anticipated challenges for the unit This document should be used when preparing to teach the unit and used as a reference throughout the unit.
_Student_Guide	This document is the "unit in a nutshell" for the student. It includes the following information: • A high-level overview of the unit, with context for many of the concepts covered. • Links to all online resources a student will need in the unit. This document should be provided to students at the start of each unit and should be kept as a reference throughout the unit.
_Slides	This document is a set of slides for use throughout the unit. You will generally find a few slides per lesson. Information contained on slides is not detailed or comprehensive. The slides are simply intended to help structure each lesson.
_Lesson	This document is the daily lesson plan. There will be multiple such documents in each unit folder, one for each lesson of the unit. Each lesson plan includes the following information: • Topic description • Student learning objectives • Materials and preparation required, including links to all online activities • Outline of the lesson • Teaching/Learning strategies (i.e. lesson details) Many lessons will include the following additional information: • Teaching tips • Additional resources • Extensions • Keys to activities and/or quizzes (when a lesson contains activities and/or quizzes)

Document Type	Description
	This document should be used to prepare for each lesson and as a reference during each lesson.
_Activity	These documents (there may be multiple activity documents associated with one lesson) is a worksheet for students. It should be copied and distributed (or electronically distributed) to students at the appropriate point in the lesson. Students are expected to submit or otherwise validate their work.
_Tutorial_Ex	These documents (there may be multiple tutorial exercise documents associated with one lesson) are worksheets associated with a TouchDevelop tutorial. There are meant to reinforce concepts introduced in a tutorial. These documents should be distributed to students prior to students' beginning a tutorial and completed while (or immediately after) students work through the tutorial. Students are expected to submit or otherwise validate their work.
_Quiz	These documents are quizzes. There will be, at most, one quiz per unit. Students are expected to submit or otherwise validate their work.
_Resource	These documents are included in some units and provide students with timely supplemental reference and guidance material related to the unit or lesson in which they are introduced.

Course Delivery

Course Outline

Creative Coding through Games and Apps is comprised of 12 units broken into 50-minute lessons to deliver up to 90 hours of academic instruction. The course is designed for flexibility in duration, with configurations that may be taught in 6, 9, 12, or 18 weeks. Suggested pacing for units and lessons is provided later in this document, with recognition the suggested pacing may be expanded or compressed to suit the needs and interests of individual classrooms, schools, and/or school systems.

The following table outlines the units, lessons and learning objectives for each lesson of the full 18-week configuration of the course. Further details about the 6, 9, and 12-week course configurations follow later in this document.

Unit 1- Surveying the Landscape: Examining Games and How they are Made			
Students analyze the structure, elements, and logic of computer games. Students define *program* and *algorithm*, examine the concept of "computationally hard problems," and create a personal collage in TouchDevelop.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
1.1	• Brainstorm definitions for "computer program" • Tutorial: Coding Jetpack Jumper • Analyze Jetpack Jumper Complete game flow • Journal entry: Revisit definitions for "computer program"	• Follow a TouchDevelop tutorial • Explain the rules, goals, and structure of the *Jetpack Jumper* game	• U1.01_Lesson • *Coding Jetpack Jumper* tutorial: http://aka.ms/StartCoding • *Jetpack Jumper Complete* game: http://aka.ms/JetpackComplete • Student journals • Headphones for students (recommended)
1.2	• Review rules, goals, and structure of *Jetpack Jumper Complete* game • Sign in to TouchDevelop • Examine the code for the *Jetpack Jumper Complete* game • Find scripts in TouchDevelop • Define "computer program" • Discuss course structure and goals	• Explain the concept of a "computer program" • Describe the structure and goals of this course • Sign in to their TouchDevelop accounts • Find scripts in their TouchDevelop accounts	• U1.02_Lesson • *Jetpack Jumper Complete* game: http://aka.ms/JetpackComplete • Jetpack Jumper Code: U1.02_Activity1 • Computer Program Definition: U1.02_Activity2 • Large space for writing ideas/definitions • Course Overview - Creative Coding with Games and Apps: U1.02_Resource • Students will need a Microsoft, Facebook, Google, or Yahoo! account
1.3	• Think-Pair-Share: Identify favorite game and characteristics • Categorize games • Define "genre" and identify computer game genres • Define "fun" • Journal entry	• Describe characteristics that make games fun • Define "genre" and describe genres for games • Categorize games into genres • Describe characteristics of successful games	• U1.03_Lesson • Note cards (or similarly sized pieces of paper), four for each student • Student journals
1.4	• Review the definition of "fun" • Video: Welcome to	• Identify and name basic features and tools in the	• U1.04_Lesson • *Chase and Gather Pro*

Unit 1- Surveying the Landscape: Examining Games and How they are Made			
Students analyze the structure, elements, and logic of computer games. Students define *program* and *algorithm*, examine the concept of "computationally hard problems," and create a personal collage in TouchDevelop.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
	TouchDevelop • Analyze rules, goals, and structure of *Chase and Gather Pro* game • Examine the code for *Chase and Gather Pro*	TouchDevelop programming environment • Describe rules, goals, and structure of the *Chase and Gather Pro* game • Describe the elements of *Chase and Gather Pro* that make it successful • Identify code for *Chase and Gather Pro* that relates to the game's rules, goals, and structure	game: http://aka.ms/ChasePro • Analyzing Game Play and Processes: U1.04_Activity • Video: *Welcome to TouchDevelop*: http://aka.ms/WelcomeToTouchDevelop
1.5	• Think-Pair-Share: Identify hard problems • Decompose a problem: How to make toast? • Define "algorithm" • Categorize problems as hard or easy	• Decompose a familiar problem into simple steps • Illustrate a process with precision and clarity • Define "algorithm" • Identify problems as hard or easy for a computer to solve	• U1.05_Lesson • Notecards • Large paper for arranging/taping notecards • *Decomposing Problems*: U1.05_Activity • Video: What's an algorithm? http://ed.ted.com/lessons/your-brain-can-solve-algorithms-david-j-malan
1.6	• Review hard problems • Unplugged activity: Exploring a hard problem • Quiz	• Collaborate to solve The 4-Color Map Problem • Identify characteristics of problems that are hard for computers to solve • Demonstrate problem-solving skills (decomposition and algorithms) in a quiz	• U1.06_Lesson • The 4-Color Map Problem: U1.06_Activity • Colored pencils • U1.06_Quiz
1.7	• Complete the Collage Tutorial • Mark the Spot! game • Journal entry	• Plot points on a gaming coordinate system • Identify code used to create and position sprites	• U1.07_Lesson • Collage Tutorial: http://aka.ms/CollageTutorial • Collage Tutorial Exercise: U1.07_Tutorial_Ex • Mark the Spot!: http://aka.ms/MarkTheSpot • Student journals • Headphones for students (recommended)
1.8	• Review the coordinate system • Introduce the Personal Collage Project • Work on the Personal Collage Project • Assess progress and plan for the	• Create a personal collage in TouchDevelop • Apply coordinate system knowledge to a Personal Collage Project	• U1.08_Lesson • Mark the Spot!: http://aka.ms/MarkTheSpot • Personal Collage Project: U1.08_Activity

Unit 1- Surveying the Landscape: Examining Games and How they are Made			
Students analyze the structure, elements, and logic of computer games. Students define *program* and *algorithm*, examine the concept of "computationally hard problems," and create a personal collage in TouchDevelop.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
	next class period		
1.9	• Review Personal Collage Project status • Finish Personal Collage Project • Self-assessment and reflection • Demonstration of a gallery walk	• Create a personalized collage in TouchDevelop • Apply coordinate system knowledge to a Personal Collage Project • Describe the process for giving feedback in a gallery walk	• U1.09_Lesson • *Personal Collage Project*: U1.08_Activity • Sticky notes of at least three different colors
1.10	• Define gallery walk rules • Gallery walk activity • Review feedback • Revise Personal Collage Project • Journal entry	• Give and receive constructive feedback in a gallery walk • Incorporate feedback into Personal Collage Projects	• U1.10_Lesson • *Personal Collage Project*: U1.08_Activity • Sticky notes of at least three different colors (enough for each student to have five of each color) • Student journals

Unit 2 -Spinning the Globe: Use of Randomization in Games			
Students explore concepts around random numbers as used in coding. Students analyze the purposes/uses of games and present their findings in a Pecha Kucha presentation.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
2.1	• Review the TouchDevelop programming environment • *Random Tutorial* • Brainstorm uses of random numbers in programs	• Identify the basic coding elements of the TouchDevelop programming environment • Use the random range function in TouchDevelop to generate random numbers	• U2.01_Lesson • *TouchDevelop Tutorials* video: http://aka.ms/ExploringTouchDevelop • *Welcome to TouchDevelop* video: http://aka.ms/WelcomeToTouchDevelop • *Random Tutorial*: http://aka.ms/RandomTutorial • *Random Tutorial Exercise*: U2.01_Tutorial_Ex
2.2	• Play *Roll the Dice!* and analyze rules • Analyze code for *Roll the Dice!* • Unplugged activity: Apply random numbers in designing a new game • Peer feedback	• Analyze a game to discover how random numbers are used in a game • Create an original game that uses random numbers	• U2.02_Lesson • http://aka.ms/RollTheDice • Enough dice for every pair of students to have one die
2.3	• Short discussion of game purposes • Independent research: The purposes of games • Create a Pecha Kucha • Ticket out the door	• Describe multiple ways that games are used to educate and inform	• U2.03_Lesson • Pecha Kucha—The wide world of Games: U2.03_Activity • How I found my passion in a dumpster: http://www.pechakucha.org/presentations/how-i-found-my-passion-in-a-dumpster • A presentation program such as Microsoft PowerPoint for students to use in creating their Pecha Kucha presentations
2.4	• Check in on research project • Group research and preparation of Pecha Kucha presentations • Ticket out the door	• Use online resources to collect information about game genres • Contribute to a collaboratively created presentation	• U2.04_Lesson • http://www.pechakucha.org/presentations/how-i-found-my-passion-in-a-dumpster • http://www.pechakucha.org/
2.5	• Set expectations regarding presentations • Presentations with feedback • Self-evaluations	• Communicate learning with a Pecha Kucha presentation • Provide constructive feedback to their peers • Self-assess their research project with a rubric	• U2.05_Lesson • Peer Evaluation Form: U2.05_Activity
2.6	• Re-examining the *Chase and Gather Basic* game • Round table activity: Create and	• Think of creative modifications to the *Chase and Gather Basic* game	• U2.06_Lesson • *Unit 2 Quiz*: U2.06_Quiz • Poster-size (16'' X 20'') or larger

Unit 2 -Spinning the Globe: Use of Randomization in Games			
Students explore concepts around random numbers as used in coding. Students analyze the purposes/uses of games and present their findings in a Pecha Kucha presentation.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
	share game designs • Quiz	• Demonstrate their knowledge of random numbers	pieces of paper and markers • *Chase and Gather Basic* game: http://aka.ms/ChaseBasic

Unit 3 -Shifting into Gear: Controlling Game Objects			
Students are introduced to objects, functions, parameters, and variables. Students write code to solve a maze problem and create a currency converter application.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
3.1	• Identify objects and functions • *Objects and Functions Tutorial* • Analyze functions in TouchDevelop games • Journal entry	• Explain what an object is and identify objects in programs • Explain what a function is and identify functions in programs • Explain what a parameter is and identify parameters in functions	• U3.01_Lesson • *Objects and Functions Tutorial*: http://aka.ms/ObjectsTutorial • *Objects and Functions Tutorial Exercise*: U3.01_Tutorial_Ex • *Exploring Functions*: U3.01_Activity • Student journals • Headphones for students (recommended)
3.2	• Review objects, functions, and parameters • *Find the Objects and Functions* activity • Unguided turtle maze problem • Journal entry	• Recall and describe objects, functions, and parameters • Analyze ways to solve a TouchDevelop turtle maze problem • Program a solution for the turtle maze problem • Reflect on the process of finding a solution	• U3.02_Lesson • *Find the Way* maze problem: http://aka.ms/FindTheWay • *Find the Objects and Functions*: U3.02_Activity • Student journals
3.3	• Brainstorm types of user input • Variable container activity • *Variables and User Input Tutorial 1* • View the *Coding in TouchDevelop* video • Introduce the currency converter project	• Give examples of user input • Describe variables • Declare and use variables in a TouchDevelop program	• U3.03_Lesson • *Variables and User Input Tutorial 1*: http://aka.ms/VariablesTutorial1 • *Variables and User Input Tutorial 1 Exercise*: U3.03_Tutorial_Ex • *Coding in TouchDevelop* video: http://aka.ms/CodingInTouchDevelop • Containers (8-10 paper cups work well), sticky notes, and small note cards or pieces of paper
3.4	• Review variables and input concepts • *Variables and User Input Tutorial 2* • Demonstrate *Currency Converter Example* • Create a currency conversion program • Self-assessment	• Utilize input and variables to calculate new information • Create a currency conversion program	• U3.04_Lesson • *Variables and User Input Tutorial 2*: http://aka.ms/VariablesTutorial2 • *Currency Converter Example*: http://aka.ms/ConverterExample • Variables On Your Own: http://aka.ms/VariablesOnYourOwn • *My Currency Converter:*

Unit 3 -Shifting into Gear: Controlling Game Objects			
Students are introduced to objects, functions, parameters, and variables. Students write code to solve a maze problem and create a currency converter application.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
			U3.04_Activity
3.5	• Review user input • *Chase and Gather Tutorial* • Understand storyboarding • Create a storyboard	• Understand objects, functions, and variables in games • Analyze *Chase and Gather* game sequence • Create a storyboard	• U3.05_Lesson • *Chase and Gather Tutorial*: http://aka.ms/ChaseTutorial • *Chase and Gather Tutorial Exercise*: U3.05_Tutorial_Ex • *Creating a Storyboard*: U3.05_Activity

Unit 4 - Gaining Ground: Controlling Game Flow with Events and Conditionals			
Students learn about events (as used in games) and conditional statements.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
4.1	• Define and identify an event • *Events Tutorial* • Events within games • Journal entry	• Identify an event • Write the code for an event in TouchDevelop • Describe what is meant by a random number within a range	• U4.01_Lesson • *Events Tutorial 1*: http://aka.ms/EventsTutorial1 • *Events Tutorial Exercise*: U4.01_Tutorial_Ex • The best Rube Goldberg Machine videos: http://www.oobject.com/best-rube-goldberg-machines-videos/ • Student journals • Headphones for students (recommended)
4.2	• Review Events • *Chase and Gather- Events Tutorial* • *Learn to Code with CODExist: the Birth of Bot - Module 4* video • Journal entry	• Create an event in TouchDevelop • Write the code for an event in TouchDevelop	• U4.02_Lesson • *Chase and Gather – Events Tutorial*: http://aka.ms/EventsTutorial2 • *Learn to Code with CODExist: the Birth of Bot - Module 4* video: http://www.microsoftvirtualacademy.com/training-courses/codexist-the-birth-of-bot?m=14252&ct=39145 • Student journals • Headphones for students (recommended)
4.3	• Unplugged activity to introduce conditionals • *Conditionals Tutorial 1 (If)* • *Conditionals Tutorial 2 (If-else)* • Journal entry	• Identify and define a conditional • Write an if-then statement • Write an if-else statement • Write the code for a conditional in TouchDevelop	• U4.03_Lesson • *Conditionals Tutorial 1*: http://aka.ms/ConditionalsTutorial1 • *Conditionals Tutorial Exercise*: U4.03_Tutorial_Ex • *Conditionals Tutorial 2*: http://aka.ms/ConditionalsTutorial2 • Student Journals • Headphones for students (recommended)
4.4	• Review decision structures • Applying logic with *and*, *or*, and *not* • Code conditional statements in TouchDevelop • Journal entry	• Create Boolean logic in an unplugged activity • Write complex conditional statements using *and*, *or*, and *not* • Identify the use of conditionals in TouchDevelop game play	• U4.04_Lesson • Objects for *and*, *or*, and *not* unplugged activity • *Roll the Dice!* game: http://aka.ms/RollTheDice • Student journals • Headphones for students

Unit 4 - Gaining Ground: Controlling Game Flow with Events and Conditionals			
Students learn about events (as used in games) and conditional statements.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
			(recommended)
4.5	• Review the partner activity from Lesson 4 • Play *Roll the Dice!* game • *Roll the Dice Tutorial* • *Conditionals Quiz*	• Apply random and conditionals in TouchDevelop • Demonstrate knowledge of random numbers and conditionals in a quiz	• U4.05_Lesson • *Roll the Dice!* game: http://aka.ms/RollTheDice • *Roll the Dice Tutorial*: http://aka.ms/DiceTutorial • *Roll the Dice Tutorial Exercise*: U4.05_Tutorial_Ex • *Conditionals Quiz*: U4.05_Quiz • Student journals • Headphones for students (recommended)

Unit 5 - Shifting into Overdrive: Using Loops and Custom Functions			
Students learn about loops and custom functions and explore the collection of built-in TouchDevelop functions that add extra interest and excitement to programs.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
5.1	• Introduction of repetitive events • *On Every Frame Tutorial* • Apply the **on every frame** statement to the *Chase and Gather Basic* game • Journal entry	• Explain the **on every frame** code • Write TouchDevelop code using the **on every frame** statement • Identify instances when conditionals and the **on every frame** code are needed	• U5.01_Lesson • *On Every Frame Tutorial*: http://aka.ms/EveryFrameTutorial1 • *On Every Frame Tutorial Exercise*: U5.01_Tutorial_Ex • *Chase and Gather Basic* game: http://aka.ms/ChaseBasic • Student journals • Headphones for students (recommended)
5.2	• *Loops Tutorial 1* • Unplugged activity • *Loops Tutorial 2* • Exploring loops on your own • Journal entry	• Identify the use of a loop in TouchDevelop • Apply the idea of a loop to a real-life situation • Write a loop using TouchDevelop	• U5.02_Lesson • *Loops Tutorial 1*: http://aka.ms/LoopsTutorial1 • *Loops Tutorial 2*: http://aka.ms/LoopsTutorial2 • *Loops Tutorial Exercise*: U5.02_Tutorial_Ex • Headphones for students (recommended)
5.3	• *Custom Functions Tutorial 1* • Review the concepts of functions, parameters, and turtle heading • Project: Design your dream house	• Identify the use of a custom function in TouchDevelop • Write a custom function using TouchDevelop • Use a custom turtle function (set heading) • Create their own project idea using TouchDevelop concepts	• U5.03_Lesson • *Custom Functions Tutorial 1*: http://aka.ms/FunctionsTutorial1 • *Custom Functions Tutorial 1* Exercise: U5.03_Tutorial_Ex • *Dream House Activity*: U5.03_Activity • Student journals • Headphones for students (recommended)
5.4	• Review project requirements • Finish the dream house project • Label the code • Self-evaluate and reflect on the project	• Apply concepts of custom functions to a creative project using TouchDevelop	• U5.04_Lesson • *Dream House Activity*: U5.03_Activity •
5.5	• Square pegs in round holes • *Custom Functions Tutorial 2* • Think-Pair-Share • Journal entry	• Understand the difference between variable types • Use correct variable types in TouchDevelop projects	• U5.05_Lesson • *Custom Functions Tutorial 2*: http://aka.ms/FunctionsTutorial2 • *Custom Functions Tutorial 2 Exercise*: U5.05_Tutorial_Ex • Student journals • Headphones for students (recommended)

Unit 5 - Shifting into Overdrive: Using Loops and Custom Functions			
Students learn about loops and custom functions and explore the collection of built-in TouchDevelop functions that add extra interest and excitement to programs.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
5.6	• *Adding Bells and Whistles Tutorial* • Ideas for adding bells and whistles to the *Chase and Gather Basic* • *Unit 5 Quiz*	• Describe enhancements available in TouchDevelop • Add enhancements to their projects	• U5.06_Lesson • *Adding Bells and Whistles Tutorial*: http://aka.ms/BellsTutorial • *Adding Bells and Whistles Tutorial Exercise*: U5.06_Tutorial_Ex • Headphones for students (recommended) • *Unit 5 Quiz*: U5.06_Quiz

Unit 6 - Charting New Territory: Making the Game Your Own			
Students explore how effective teams work together through collaboratively planning and creating a TouchDevelop game.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
6.1	• Half-Baked Pitch • Project overview and rubric • *Game Developer Resume* activity	• Demonstrate collaboration skills • Practice pitching a project through a fun exercise • Understand how their projects will be evaluated • Assess skills related to projects	• U6.01_Lesson • *Game Developer Resume*: U6.01_Activity • *Fix Chase and Gather*: U6.01_Activity2a • *Build Chase and Gather*: U6.01_Activity2b • *Build Your Own Game*: U6.01_Activity2c • Two containers
6.2	• Form student teams • Set ground rules/guidelines for teamwork • Teams plan project pitches and create storyboards • Journal entry: Excited and scared	• Understand the expectations of working in a team • Develop their arguments for choosing a particular project • Decompose the project into a storyboard	• U6.02_Lesson • *Storyboard Your Game*: U6.02_Activity1
6.3	• Pitch projects • Give and receive feedback • Finish *Project Pitch Evaluation*	• Communicate reasons for selecting a topic • Practice public speaking • Give and receive constructive feedback • Assess feedback and incorporate ideas into their plans	• U6.03_Lesson • *Project Pitch Evaluation*: U6.03_Activity
6.4	• *TouchDevelop Beyond the Basics* video • *Ready Set Go*: Evaluate knowledge and research needs • Create a game company and project name • Decompose the tasks, delegate, and create Scrum boards	• Improve project and time management skills • Identify their current knowledge and skills • Analyze the skills need to produce the project compared to the skills they possess • Plan project tasks and delegate responsibilities	• U6.04_Lesson • *Ready, Set, Go*: U6.04_Activity • *TouchDevelop Beyond the Basics* video: http://aka.ms/BeyondBasicsInTouchDevelop
6.5 – 6.10	• Program games (lessons 5-7) • Periodically review progress and needs • Draft review (lesson 8) • Reassess, revise, finalize (lesson 9) • Conduct user observations (lesson 10) • Conduct user interviews (lesson 10) • Reassess, revise, finalize (lesson 10)	• Collaborate in a team project • Meet responsibilities for project production • Apply programming concepts learned to a new project • Give and receive feedback • Evaluate feedback and revise the project accordingly • Gather user input through observations • Gather user input through interviews	• U6.05-10_Lesson • *Draft Version Review*: U6.09_Activity • *User Observation*: U6.10_Activity1 • *User Interview*: U6.10_Activity2

Unit 6 - Charting New Territory: Making the Game Your Own			
Students explore how effective teams work together through collaboratively planning and creating a TouchDevelop game.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
6.11	• Define presentation requirements • Prepare presentation • Create a users' manual or other documentation	• Articulate instructions for playing the game • Plan a creative project presentation	• U6.12_Lesson •
6.12 – 6.13	• Group Presentations and feedback • Assess the project as a team • Complete the *Project Reflection* activity	• Communicate about the project in a presentation format • Share feedback • Evaluate learning	• U6.12-13_Lesson • *Project Reflection*: U6.12_Activity • *Fix Chase and Gather*: U6.01_Activity2a • *Build Chase and Gather*: U6.01_Activity2b • *Build Your Own Game*: U6.01_Activity2c

Unit 7 - Refueling: Adding Animation with Sprite Sheets			
Students learn how to use sprite sheets to create animations in TouchDevelop games.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
7.1	• Creating animations • Create a flip book • Flip books to sprite sheets • Design a sprite sheet	• Demonstrate how a series of individual frames can create an animated image • Create a paper prototype of a sprite sheet	• U7.01_Lesson • Sticky notes (at least 10 per student) • *How to make a Flip Book Animation* https://www.youtube.com/watch?v=Njl-uqnmBGA • *What is a sprite sheet?* https://www.codeandweb.com/what-is-a-sprite-sheet
7.2	• Using sprite sheets in games • *Animation with Sprite Sheets Tutorial* • Sprite Sheet limitations • Journal entry	• Use a sprite sheet in a TouchDevelop game • Change the parameters of **set frame grid** to work with different sprite sheets	• U7.02_Lesson • http://aka.ms/SpritesheetsTutorial • *Animation with Sprite Sheets Exercise*: U7.02_Tutorial_Ex • Sprite sheet used in the tutorial: http://aka.ms/spritesheet1 • Student journals • Headphones for students (recommended)
7.3	• Planning a sprite animation • Create an animated sprite sheet • Share and reflect	• Create a sprite sheet using an online image editor	• U7.03_Lesson • *Jetpack Jumper Complete* game: http://aka.ms/JetpackComplete • *Animated Sprites*: U7.03_Activity • Piskel online sprite creator: http://www.piskelapp.com
7.4	• Tell a story with animation • Create a TouchDevelop story • Discuss animation challenges	• Explain how to add a sprite sheet to a TouchDevelop game	• U7.04_Lesson • *Sprite Sheet Animation*: U7.04_Activity

Unit 8 - Riding the Momentum: Loops and Collections			
Students reinforce and expand their understanding of iteration and apply it to collections and other data structures.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
8.1	• Review loops • Counting Birthday Money • *Loop Counter Tutorial* • Challenge activity	• Recall and use **for loops** • Understand the use of the counting variable • Use a counting variable in the body of the loop •	• U8.01_Lesson • *Loop Counter Tutorial:* http://aka.ms/LoopCounterTutorial • *Loop Counter Tutorial Exercise*: U8.01_Tutorial_Ex • *Birthday Money*: U8.01_Activity1 • *More Obstacles*: U8.01_Activity2
8.2	• Introduction to Collections • Collections Tutorial • Journal entry	• Understand collections • Understand and apply knowledge to add, remove, and insert at elements • Use a for loop and a for each loop to iterate through the collection	• U8.02_Lesson • *Collections Tutorial*: http://aka.ms/CollectionsTutorial • *Collections Tutorial Exercise*: U8.02_Tutorial_Ex • *Variables vs. Arrays Vs. Objects* video https://www.youtube.com/watch?v=FjSCdXd04Cs&list=PLS9eKaCePM68xlSJC3a9KGQEDwmoKWbxC • Student journals • Headphones for students (recommended)
8.3	• Review of Loops and Random Generator • Write pseudocode for the Band Name Generator • Brainstorm collection/loop/random uses in games	• Explain what a loop is and how to use the control variable • Explain how random numbers are generated • Analyze for loop, random number and control variable constructs •	• U8.03_Lesson • Notecards with words (prepared before class) and dice (1 die per group) • Brand Name Generator: http://www.bandnamemaker.com • Random Words: http://aka.ms/RandomWords
8.4	• Whacky Family Vacation • Analyze an online word game • Create a wacky tale game	• Plan a program that uses collections, loops, and random numbers • Explain how the constructs will be used in the game •	• U8.04_Lesson • *A Wacky Family Vacation*: U8.04_Activity1 • Mad Libs: http://www.madglibs.com/ • *Wacky Tales*: U8.04_Activity2
8.5	• Review plans • Code a wacky tale game	• Apply their knowledge of collections, random numbers, loops, and variables in coding a TouchDevelop wacky tales game	• U8.05_Lesson • *Wacky Tales*: U8.04_Activity2

Unit 8 - Riding the Momentum: Loops and Collections			
Students reinforce and expand their understanding of iteration and apply it to collections and other data structures.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
		• Use problem-solving strategies to resolve challenges or errors	
8.6	• Finish the wacky tale • Play other games and provide feedback to classmates • Assess the game and teamwork skills	• Complete the Wacky Tales project • Provide and accept feedback from their peers • Assess their project, as well as their problem-solving and collaborative skills	• U8.06_Lesson • *Wacky Tales*: U8.04_Activity2

Unit 9 - Into the Great Beyond: Cloud Computing			
Students learn about the cloud and how it can be used to create games and apps that share data and connect people.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
9.1	• How the Internet works video • Messages in a Bottle • What is cloud computing? • How do we use the cloud?	• Explain how information travels across the Internet • Define cloud computing • Describe the role of cloud computing and cloud storage	• U9.01_Lesson • *How The Internet Works* video: http://www.wimp.com/internetworks/ • *The Internet Explained*: U9.01_Activity1 • *Command Cards*: U9.01_Resource • *Cloud Computing* video: http://www.schooltube.com/video/84a8c36a7b534643939b/Cloud%20Computeing • *Cloud Computing*: U9.01_Activity2 • Student Journals
9.2	• What is a Cloud Variable? • *Cloud Tutorial* • Chat app • Uses for cloud variables	• Explain the role of cloud variables • Create an app that uses cloud data • Explain the difference between a cloud variable and a temporary variable	• U9.02_Lesson • *Running Cloud Apps*: U9.02_Resource • *Cloud Tutorial*: http://aka.ms/CloudTutorial • *Cloud Tutorial Exercise:* U9.02_Tutorial_Ex • *Instant Messenger*: http://aka.ms/InstantMessenger • *Chat using The Cloud:* U9.02_Activity • Headphones for students (recommended)
9.3	• Understanding messaging apps • *Instant Messenger Tutorial* • Summarizing cloud variables	• Use a cloud variable in an app • Publish and share a cloud-data app	• U9.03_Lesson • *How IM Works* video: https://www.youtube.com/watch?v=ybcHjUP3TtY • *Instant Messenger Tutorial*: http://aka.ms/IMTutorial • *Instant Messenger Tutorial Exercise*: U9.03_Tutorial_Ex • Headphones for students (recommended)
9.4	• What is a Database? • Create a table with a tutorial • *Tables – On Your Own* activity	• Explain the differences between cloud variables and tables • Access a table in a TouchDevelop app • Create a table with their own	• U9.04_Lesson • *What is a Database* video: https://www.youtube.com/watch?v=t8jgX1f8kc4 • *Table Tutorial*: http://aka.ms/TableTutorial

Unit 9 - Into the Great Beyond: Cloud Computing			
Students learn about the cloud and how it can be used to create games and apps that share data and connect people.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
		data	• *Table Tutorial Exercise*: U9.04_Tutorial_Ex • *Tables – On Your Own*: U9.04_Activity • Headphones for students (recommended)
9.5	• Review assignment details • Creating a table and app • Reflection	• Design, create and populate a table • Create an app to use a table to generate new information	• U9.05_Lesson • *Creating a Table*: U9.05_Activity

Unit 10 - Discovering New Horizons: TouchDevelop App Design			
Students are introduced to apps (as opposed to games), app design and app purpose.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
10.1	• Brainstorm elements of useful apps • Comparing games and apps • Basic user interface (UI) design • Group story creation	• Distinguish between apps and games in TouchDevelop • Identify the role of pages in an app • Understand basic design concepts	• U10.01_Lesson • *App Design Rules*: U10.01_Activity1 • *Group Story*: U10.01_Activity2 • "3 Rules of App Design" http://www.fastcodesign.com/3041374/3-rules-of-app-design-according-to-yahoos-marissa-mayer • *App Example*: http://aka.ms/AppExample
10.2	• The differences between apps and games • *App Tutorial 1 Exercise* • Storytelling App • Journal entry: identifying boxes and layout	• Create and load pages in TouchDevelop • Use boxes to arrange elements on a page	• U10.02_Lesson • *App Tutorial 1*: http://aka.ms/AppTutorial1 • *App Tutorial 1 Exercise*: U10.02_Tutorial_Ex • Student journals • Headphones for students (recommended)
10.3	• Parameters Review • *App Tutorial 2* • Journal entry: Improving the multiplication practice app	• Identify the roles of the data, initialize, and display sections of a page • Pass data to a page using a parameter	• U10.03_Lesson • *App Tutorial 2*: http://aka.ms/AppTutorial2 • *App Tutorial 2 Exercise*: U10.03_Tutorial_Ex • Student journals • Headphones for students (recommended)
10.4	• Flashcard Ideas • Examine the App Example • Create a Flashcard App • Journal entry: Flashcard Challenges	• Understand how tables are used in the context of pages • Create their own flash card app	• U10.04_Lesson • *App Example*: http://aka.ms/AppExample • *Flashcard App Activity*: U10.04_Activity • Student journals
10.5	• Review assignment requirements • Work on flashcard apps • Reflection	• Create their own flash card app	• U10.05_Lesson • *App Example*: http://aka.ms/AppExample • *Flashcard App Activity*: U10.04_Activity

Unit 11 - Traveling Safely: Privacy and Encryption			
Students are introduced to the fundamentals of transmitting data safely across the Internet.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
11.1	• What is Digital Privacy? • Why Use Encryption? • Converting Text to Unicode • Journal entry	• Describe the Internet of things • Describe multiple uses for data • Describe ways that personal information could be used in inappropriate ways • Encode/decode messages using Unicode	• U11.01_Lesson • *Internet of Things*: https://www.youtube.com/watch?v=GApYwlnO0G4 • 8.5 x 11" pieces of scrap paper • Tape • *Unicode Activity*: U11.01_Activity • Student journals
11.2	• The Caesar Cipher • Coding with the *Encryption Tutorial* • Adding decryption • Exit Ticket: Partner feedback	• Describe the Caesar Cipher • Encrypt and decrypt messages in a TouchDevelop app	• U11.02_Lesson • *Encryption Tutorial*: http://aka.ms/EncryptionTutorial • *Encryption Tutorial Exercise*: U11.02_Tutorial_Ex • *Decryption Example*: http://aka.ms/DecryptionExample • Headphones for students (recommended)
11.3	• Journal Entry • Class Discussion about sign-in security • *Creating Strong Passwords Activity* • *Cryptographic Hash Functions Activity*	• Describe the steps they can take to keep someone from gaining access to their online accounts • List important considerations when creating passwords • Describe steps websites use to keep passwords secure	• U11.03_Lesson • *Creating Strong Passwords*: U11.03_Activity1 • *Cryptographic Hash Functions*: U11.03_Activity2 • Online Hash Calculator: http://www.xorbin.com/tools/sha1-hash-calculator • *How NOT to Store Passwords!*: https://www.youtube.com/watch?v=8ZtInClXe1Q • Student journals
11.4	• Review of password security • Add a password feature • Test the apps	• Add a password feature to the encryption app • Store the hashed password in the cloud	• U11.04_Lesson • *Password Example*: http://aka.ms/PasswordExample (for teacher use) • *Creating a Password Hash*: U11.04_Activity

Unit 12 - You Have Arrived: Independent Project Development			
Students work in teams to design, build, present, and market a game or app.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
12.1	• Review project guidelines and procedures • Brainstorm session • Share ideas, discuss, and narrow the list of project ideas • Students apply to work on their favorite projects	• Understand the requirements and schedule of the final project • Brainstorm project ideas to fit criteria • Analyze a problem for a specific audience in terms of how an app or game can contribute to the solution	• U12.01_Lesson • *Build Your App or Game*: U12.01_Activity1 • *Project Application*: U12.01_Activity2 • *Audience List*: U12.01_Resource.
12.2	• Initial team organization and planning • Know your audience • Sketch initial ideas • Outside of class: Conduct interviews	• Form a cooperative team • Gather information about the target group • Create sketches of initial solution design ideas • Plan interviews with audience members	• U12.02_Lesson • Scrum task board • *User Interviews and Research*: U12.02_Activity1 • *Thumbnail Sketches*: U12.02_Activity2
12.3	• Progress update • The user is not you! • Advice from the experts • Journal entry	• Track project progress • Describe user-experience principles that will impact their projects	• U12.03_Lesson • Scrum boards • *UX Principles*: U12.03_Activity1 • What the heck is UX Design?: https://www.youtube.com/watch?v=Ovj4hFxko7c • How giant websites design for you: https://www.ted.com/talks/margaret_gould_stewart_how_giant_websites_design_for_you_and_a_billion_others_too?language=en
12.4	• Update the Scrum board • Summarize the interviews with the *Empathy Map* • Write the algorithm and create storyboards • Plan class presentation of storyboard and plans • Outside of class: present storyboards and designs with potential users	• Track project progress • Empathize with members of the audience group • Create an algorithm and story board as a guide for the development process • Recognize the value of storyboarding and algorithms in project development • Gather feedback on their initial design with a potential user	• U12.04_Lesson • Scrum boards • *Empathy Map*: U12.04_Activity1 • *Storyboard Your Game or App*: U12.04_Activity2 • *Predevelopment User Testing*: U12.04_Activity3
12.5	• Update the Scrum board • Present projects plans, give and receive feedback • Assess the feedback and revise plans • Define tasks for the project production stage	• Track project progress • Communicate project plan to the class • Give and receive constructive feedback • Assess feedback and revise project plans	• U12.05_Lesson • Scrum boards • *Project Pitch Feedback and Revisions*: U12.05_Activity

Unit 12 - You Have Arrived: Independent Project Development			
Students work in teams to design, build, present, and market a game or app.			
Lesson	Outline of Lesson	Students will be able to…	Required Resources
12.6 - 17	• Update Scrum boards • Program projects • Conduct user observations and interviews • Draft review on day 15 • Reassess, revisit, finalize • Reflect on learning	• Track project progress • Follow a storyboard and algorithm to create a game or app • Apply computer science concepts and TouchDevelop constructs to a project • Collaborate within a team • Demonstrate time management skills • Give and receive feedback • Follow a cycle of create, evaluate, revise • Conduct user observation • Conduct user interviews	• U12.06-17_Lesson • Scrum boards • *User Observation*: U12.14_Activity1 • *User Interview*: U12.14_Activity2 • *Draft Review*: U12.15_Activity
12.18 - 19	• Update the Scrum board • Define presentation requirements • Prepare presentations • Create a users' manual	• Track project progress • Articulate instructions for their game or app in a written manual • Plan a project presentation	• U12.18-19_Lesson
12.20 - 21	• Finalize the Scrum board • Group presentations and feedback • As a team, assess project with a rubric • Complete individual project reflection	• Present projects • Provide feedback • Evaluate and reflect on their learning	• U12.20-21_Lesson • *Project Reflection*: U12.21_Activity • *Build Your App or Game*: U12.01_Activity1

Standards Map

A quality computer science curriculum not only engages students in interesting, collaborative learning activities using sound pedagogical strategies, but it also ensures that students are learning concepts vital to the deep and broad field of computer science. *Creative Coding through Games and Apps* has been designed to achieve this goal by aligning with the *CSTA K–12 Computer Science Standards*. While this is course is primarily an introductory programming course, units are grounded in an array of standards from the strands of Computational Thinking, Collaboration, Computing Practice and Programming, Computer and Communication Devices, and Community, Global, and Ethical Impacts.

Unit 1: Surveying the Landscape: Examining Games and How they are Made
Students analyze the structure, elements, and logic of computer games. Students define *program* and *algorithm*, examine the concept of "computationally hard problems," and create a personal collage in TouchDevelop.
Collaboration CL 2-2. Collaboratively design, develop, publish, and present products (e.g., videos, podcasts, websites) using technology resources that demonstrate and communicate curriculum concepts. CL 2-3. Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities. **Computational Thinking** CT 2-1. Use the basic steps in algorithmic problem-solving to design solutions. CT 2-3. Define an algorithm as a sequence of instructions that can be processed by a computer. CT 2-6. Describe and analyze a sequence of instructions being followed (e.g., describe a character's behavior in a video game as driven by rules and algorithms). CT 2-10. Evaluate what kinds of problems can be solved using modeling and simulation. CT 2-12. Use abstraction to decompose a problem into sub problems. CT 2-14. Examine connections between elements of mathematics and computer science including binary numbers, logic, sets and functions. CT 3A-3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. **Computing Practice and Programming** CPP 2-3. Design, develop, publish, and present products using technology resources that demonstrate and communicate curriculum concepts. CPP 2-4. Demonstrate an understanding of algorithms and their practical application.
Unit 2: Spinning the Globe: Use of Randomization in Games
Students explore concepts around random numbers as used in coding. Students analyze the purposes/uses of games and present their findings in a Pecha Kucha presentation.
Collaboration CL 2-2. Collaboratively design, develop, publish, and present products (e.g., videos, podcasts, websites) using technology resources that demonstrate and communicate curriculum concepts. CL 2-3. Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities. **Computational Thinking** CT 2-1. Use the basic steps in algorithmic problem-solving to design solutions. CT 2-3. Define an algorithm as a sequence of instructions that can be processed by a computer. CT 2-6. Describe and analyze a sequence of instructions being followed (e.g., describe a character's behavior in a video game as driven by rules and algorithms). CT 2-10. Evaluate what kinds of problems can be solved using modeling and simulation. CT 2-12. Use abstraction to decompose a problem into sub problems.

Unit 1: Surveying the Landscape: Examining Games and How they are Made
Students analyze the structure, elements, and logic of computer games. Students define *program* and *algorithm*, examine the concept of "computationally hard problems," and create a personal collage in TouchDevelop.
CT 2-14. Examine connections between elements of mathematics and computer science including binary numbers, logic, sets and functions. CT 3A-3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. **Computing Practice and Programming** CPP 2-3. Design, develop, publish, and present products using technology resources that demonstrate and communicate curriculum concepts. CPP 2-4. Demonstrate an understanding of algorithms and their practical application. **Community, Global, and Ethical Impacts** CI 2-3. Analyze the positive and negative impacts of computing on human culture.
Unit 3: Shifting into Gear: Controlling Game Objects
Students are introduced to objects, functions, and parameters. Students write code to solve a maze problem and create a currency converter application.
Collaboration CL 2-2. Collaboratively design, develop, publish, and present products (e.g., videos, podcasts, websites) using technology resources that demonstrate and communicate curriculum concepts. CL 2-3. Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities. **Computational Thinking** CT 2-1. Use the basic steps in algorithmic problem-solving to design solutions. CT 2-6. Describe and analyze a sequence of instructions being followed (e.g., describe a character's behavior in a video game as driven by rules and algorithms). CT 2-12. Use abstraction to decompose a problem into sub problems. CT 2-14. Examine connections between elements of mathematics and computer science including binary numbers, logic, sets and functions. CT 3A-1. Use predefined functions and parameters, classes and methods to divide a complex problem into simpler parts. CT 3A-3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms CT 3A-8. Use modeling and simulation to represent and understand natural phenomena. **Computing Practice and Programming** CPP 2-3. Design, develop, publish, and present products using technology resources that demonstrate and communicate curriculum concepts. CPP 2-4. Demonstrate an understanding of algorithms and their practical application. CPP 2-5. Implement problem solutions using a programming language including: looping behavior, conditional statements, logic, expressions, variables, and functions.
Unit 4: Gaining Ground: Controlling Game Flow with Events and Conditionals
Students learn about events (as used in games) and conditional statements.
Collaboration CL 2-2. Collaboratively design, develop, publish, and present products (e.g., videos, podcasts, websites) using technology resources that demonstrate and communicate curriculum concepts. CL 2-3. Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities. **Computational Thinking** CT 2-1. Use the basic steps in algorithmic problem-solving to design solutions. CT 2-6. Describe and analyze a sequence of instructions being followed (e.g., describe a character's

Unit 1: Surveying the Landscape: Examining Games and How they are Made
Students analyze the structure, elements, and logic of computer games. Students define *program* and *algorithm*, examine the concept of "computationally hard problems," and create a personal collage in TouchDevelop.
behavior in a video game as driven by rules and algorithms). CT 2-12. Use abstraction to decompose a problem into sub problems. CT 2-14. Examine connections between elements of mathematics and computer science including binary numbers, logic, sets and functions. CT 3A-1. Use predefined functions and parameters, classes and methods to divide a complex problem into simpler parts. CT 3A-3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. CT 3A-8. Use modeling and simulation to represent and understand natural phenomena, **Computing Practice and Programming** CPP 2-3. Design, develop, publish, and present products using technology resources that demonstrate and communicate curriculum concepts. CPP 2-4. Demonstrate an understanding of algorithms and their practical application. CPP 2-5. Implement problem solutions using a programming language including: looping behavior, conditional statements, logic, expressions, variables, and functions
Unit 5: Shifting into Overdrive: Using Loops and Custom Functions
Students learn about loops and custom functions and explore the collection of built-in TouchDevelop functions that add extra interest and excitement to programs.
Collaboration CL 2-3. Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities. **Computational Thinking** CT 2-1. Use the basic steps in algorithmic problem-solving to design solutions. CT 2-6. Describe and analyze a sequence of instructions being followed (e.g., describe a character's behavior in a video game as driven by rules and algorithms). CT 2-12. Use abstraction to decompose a problem into sub problems. CT 3A-1. Use predefined functions and parameters, classes and methods to divide a complex problem into simpler parts. CT 3A-3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. CT 3A-8. Use modeling and simulation to represent and understand natural phenomena. **Computing Practice and Programming** CPP 2-3. Design, develop, publish, and present products using technology resources that demonstrate and communicate curriculum concepts. CPP 2-4. Demonstrate an understanding of algorithms and their practical application. CPP 2-5. Implement problem solutions using a programming language including: looping behavior, conditional statements, logic, expressions, variables, and functions.
Unit 6: Charting New Territory: Making the Game Your Own
Students explore how effective teams work together through collaboratively planning and creating a TouchDevelop game.
Collaboration CL 2-2. Collaboratively design, develop, publish, and present products (e.g., videos, podcasts, websites) using technology resources that demonstrate and communicate curriculum concepts. CL 2-3. Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities. **Computational Thinking**

Unit 1: Surveying the Landscape: Examining Games and How they are Made
Students analyze the structure, elements, and logic of computer games. Students define *program* and *algorithm*, examine the concept of "computationally hard problems," and create a personal collage in TouchDevelop.
CT 2-1. Use the basic steps in algorithmic problem-solving to design solutions. CT 2-6. Describe and analyze a sequence of instructions being followed (e.g., describe a character's behavior in a video game as driven by rules and algorithms). CT 2-12. Use abstraction to decompose a problem into sub problems. CT 3A-1. Use predefined functions and parameters, classes and methods to divide a complex problem into simpler parts. CT 3A-3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. CT 3A-8. Use modeling and simulation to represent and understand natural phenomena. **Computing Practice and Programming** CPP 2-3. Design, develop, publish, and present products using technology resources that demonstrate and communicate curriculum concepts. CPP 2-4. Demonstrate an understanding of algorithms and their practical application. CPP 2-5. Implement problem solutions using a programming language, including: looping behavior, conditional statements, logic, expressions, variables, and functions. Community, Global, and Ethical Impacts CI 2-3. Analyze the positive and negative impacts of computing on human culture.
Unit 7: Refueling: Adding Animation with Sprite Sheets
Students design sprite sheets and implement in a TouchDevelop game.
Collaboration CL 2-3. Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities. **Computational Thinking** CT 2-1. Use the basic steps in algorithmic problem-solving to design solutions. CT 2-6. Describe and analyze a sequence of instructions being followed (e.g., describe a character's behavior in a video game as driven by rules and algorithms). CT 2-12. Use abstraction to decompose a problem into sub problems. CT 2-14. Examine connections between elements of mathematics and computer science including binary numbers, logic, sets and functions. CT 3A-1. Use predefined functions and parameters, classes and methods to divide a complex problem into simpler parts. CT 3A-3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. CT 3A-8. Use modeling and simulation to represent and understand natural phenomena. **Computing Practice and Programming** CPP 2-3. Design, develop, publish, and present products using technology resources that demonstrate and communicate curriculum concepts. CPP 2-4. Demonstrate an understanding of algorithms and their practical application. CPP 2-5. Implement problem solutions using a programming language including: looping behavior, conditional statements, logic, expressions, variables, and functions.
Unit 8: Riding the Momentum: Loops and Collections
Students learn about loops and counters and apply those concepts to a "random phrase" generator app.
Collaboration CL 2-3. Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities.

Unit 1: Surveying the Landscape: Examining Games and How they are Made
Students analyze the structure, elements, and logic of computer games. Students define *program* and *algorithm*, examine the concept of "computationally hard problems," and create a personal collage in TouchDevelop.
Computational Thinking CT 2-1. Use the basic steps in algorithmic problem-solving to design solutions. CT 2-6. Describe and analyze a sequence of instructions being followed (e.g., describe a character's behavior in a video game as driven by rules and algorithms). CT 2-12. Use abstraction to decompose a problem into sub problems. CT 2-14. Examine connections between elements of mathematics and computer science including binary numbers, logic, sets and functions. CT 3A-1. Use predefined functions and parameters, classes and methods to divide a complex problem into simpler parts. CT 3A-3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. CT 3A-8. Use modeling and simulation to represent and understand natural phenomena. Computing Practice and Programming CPP 2-3. Design, develop, publish, and present products using technology resources that demonstrate and communicate curriculum concepts. CPP 2-4. Demonstrate an understanding of algorithms and their practical application. CPP 2-5. Implement problem solutions using a programming language including: looping behavior, conditional statements, logic, expressions, variables, and functions.
Unit 9: Into the Great Beyond: Cloud Computing
Students explore cloud computing and implement cloud variables and structured data storage in creating a chat app.
Collaboration CL 2-3. Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities. **Computational Thinking** CT 2-1. Use the basic steps in algorithmic problem-solving to design solutions. CT 2-6. Describe and analyze a sequence of instructions being followed (e.g., describe a character's behavior in a video game as driven by rules and algorithms). CT 2-12. Use abstraction to decompose a problem into sub problems. CT 3A-1. Use predefined functions and parameters, classes and methods to divide a complex problem into simpler parts. CT 3A-3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. CT 3B-6. Compare and contrast simple data structures and their uses (e.g., arrays and lists). Computing Practice and Programming CPP 2-4. Demonstrate an understanding of algorithms and their practical application. CPP 2-5. Implement problem solutions using a programming language including: looping behavior, conditional statements, logic, expressions, variables, and functions. **Computers and Communications Devices** CD 3A-9. Describe how the Internet facilitates global communication.
Unit 10: Discovering New Horizons: TouchDevelop App Design
Students study app design and create a multi-page flash card app.
Collaboration CL 2-3. Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities. **Computational Thinking**

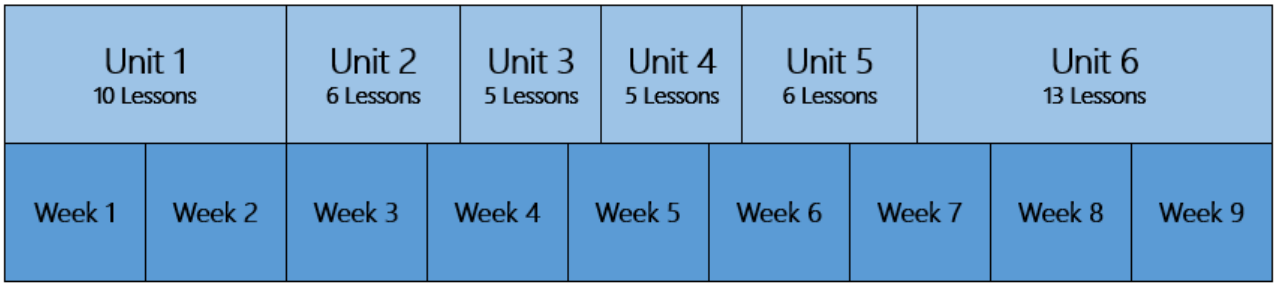
Unit 1: Surveying the Landscape: Examining Games and How they are Made
Students analyze the structure, elements, and logic of computer games. Students define *program* and *algorithm*, examine the concept of "computationally hard problems," and create a personal collage in TouchDevelop.
CT 2-1. Use the basic steps in algorithmic problem-solving to design solutions. CT 2-6. Describe and analyze a sequence of instructions being followed (e.g., describe a character's behavior in a video game as driven by rules and algorithms). CT 2-12. Use abstraction to decompose a problem into sub problems. CT 3A-1. Use predefined functions and parameters, classes and methods to divide a complex problem into simpler parts. CT 3A-3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. CT 3B-6. Compare and contrast simple data structures and their uses (e.g., arrays and lists). Computing Practice and Programming CPP 2-4. Demonstrate an understanding of algorithms and their practical application. CPP 2-5. Implement problem solutions using a programming language including: looping behavior, conditional statements, logic, expressions, variables, and functions. CPP 3A-1. Create and organize Web pages through the use of a variety of web programming design tools. CPP 3A-2. Use mobile devices/emulators to design, develop, and implement mobile computing applications. **Computers and Communications Devices** CD 3A-9. Describe how the Internet facilitates global communication.
Unit 11: Traveling Safely: Privacy and Encryption
Students explore Internet privacy issues and encryption, and apply the concepts to creating a password app.
Collaboration CL 2-3. Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities. **Computational Thinking** CT 2-1. Use the basic steps in algorithmic problem-solving to design solutions. CT 2-6. Describe and analyze a sequence of instructions being followed (e.g., describe a character's behavior in a video game as driven by rules and algorithms). CT 2-12. Use abstraction to decompose a problem into sub problems. CT 3A-1. Use predefined functions and parameters, classes and methods to divide a complex problem into simpler parts. CT 3A-3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. CT 3A-7. Describe how various types of data are stored in a computer system. CT 3B-6. Compare and contrast simple data structures and their uses (e.g., arrays and lists). **Computing Practice and Programming** CPP 2-4. Demonstrate an understanding of algorithms and their practical application. CPP 2-5. Implement problem solutions using a programming language including: looping behavior, conditional statements, logic, expressions, variables, and functions. CPP 3A-2. Use mobile devices/emulators to design, develop, and implement mobile computing applications. CPP 3A-9. Explain the principles of security by examining encryption, cryptography, and authentication techniques. CPP 3B-5. Deploy principles of security by implementing encryption and authentication strategies. **Computers and Communications Devices** CD 3A-9. Describe how the Internet facilitates global communication.

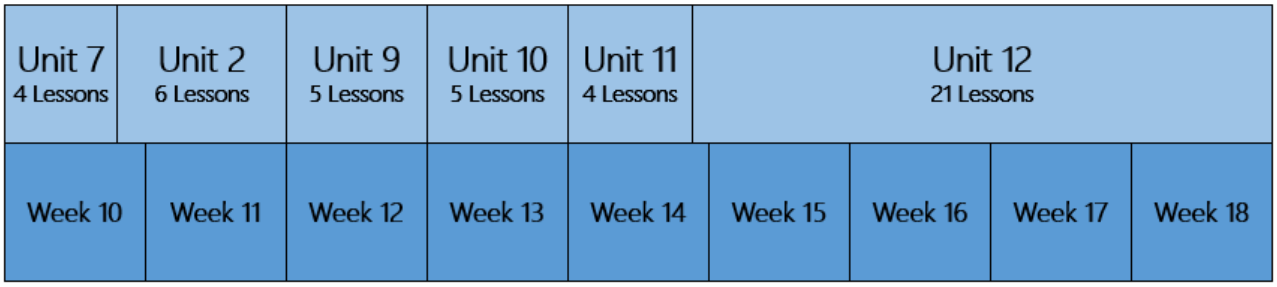
Unit 1: Surveying the Landscape: Examining Games and How they are Made
Students analyze the structure, elements, and logic of computer games. Students define *program* and *algorithm*, examine the concept of "computationally hard problems," and create a personal collage in TouchDevelop.
Unit 12: You Have Arrived: Independent Project Development
Students explore how effective teams work together through collaboratively planning and creating a TouchDevelop game or app that incorporates concepts from the second nine weeks of the course.
Collaboration CL 2-2. Collaboratively design, develop, publish, and present products (e.g., videos, podcasts, websites) using technology resources that demonstrate and communicate curriculum concepts. CL 2-3. Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities. **Computational Thinking** CT 2-1. Use the basic steps in algorithmic problem-solving to design solutions. CT 2-6. Describe and analyze a sequence of instructions being followed (e.g., describe a character's behavior in a video game as driven by rules and algorithms). CT 2-12. Use abstraction to decompose a problem into sub problems. CT 3A-1. Use predefined functions and parameters, classes and methods to divide a complex problem into simpler parts. CT 3A-3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. CT 3A-8. Use modeling and simulation to represent and understand natural phenomena. Computing Practice and Programming CPP 2-3. Design, develop, publish, and present products using technology resources that demonstrate and communicate curriculum concepts. CPP 2-4. Demonstrate an understanding of algorithms and their practical application. CPP 2-5. Implement problem solutions using a programming language including: looping behavior, conditional statements, logic, expressions, variables, and functions. CPP 3A-1. Create and organize Web pages through the use of a variety of web programming design tools. CPP 3A-2. Use mobile devices/emulators to design, develop, and implement mobile computing applications. **Computers and Communications Devices** 3A-9. Describe how the Internet facilitates global communication. **Community, Global, and Ethical Impacts** CI 2-3. Analyze the positive and negative impacts of computing on human culture.

Pacing Guide

The diagram below provides suggested pacing guidance for *Creative Coding through Games and Apps*. For a day-by-day breakdown of lessons, see the *Course Outline* section above.

Note that each lesson is designed to be 50 minutes in duration. It is recognized that the suggested pacing for individual lessons and/or units of the course may be expanded or compressed to suit the needs and interests of individual classrooms, schools, and/or school systems.


Unit 1
10 Lessons
Unit 2
6 Lessons
Unit 3
5 Lessons
Unit 4
5 Lessons
Unit 5
6 Lessons
Unit 6
13 Lessons
Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Week 7
Week 8
Week 9


Unit 7
4 Lessons
Unit 2
6 Lessons
Unit 9
5 Lessons
Unit 10
5 Lessons
Unit 11
4 Lessons
Unit 12
21 Lessons
Week 10
Week 11
Week 12
Week 13
Week 14
Week 15
Week 16
Week 17
Week 18

Course Duration Options

Creative Coding through Games and *Apps* is designed to be flexible, allowing for different configurations of the course to be taught in 6, 9, 12, or 18 weeks. The 18-week configuration is detailed in the table earlier in this document. The following table shows which units are incorporated into the 6-, 9-, and 12-week configurations. Unit sequence varies across configurations. The table does not show the sequence of units for each configuration. For further details about each configuration, see the document sections immediately following this table.

Overview of Options				
	6-Week Course (30 lessons)	9-Week Course (45 lessons)	12-Week Course (60 lessons)	18-Week Course (90 lessons)
Unit 1 Surveying the Landscape: Examining Games and How they are Made	Selected lessons from Units 1-6.	✓	✓	✓
Unit 2 Spinning the Globe: Spinning the Globe: Use of Randomization in Games		✓	✓	✓
Unit 3 Shifting into Gear: Controlling Game Objects		✓	✓	✓
Unit 4 Gaining Ground: Controlling Game Flow with Events and Conditionals		✓	✓	✓
Unit 5 Shifting into Overdrive: Using Loops and Custom Functions		✓	✓	✓
Unit 6 Charting New Territory: Making the Game Your Own		✓	✓	✓
Unit 7 Refueling: Adding Animation with Sprite Sheets	X	X	✓	✓
Unit 8 Riding the Momentum: Loops and Collections	X	X	✓	✓
Unit 9 Into the Great Beyond: Cloud Computing	X	X	✓	✓
Unit 10 Discovering New Horizons: TouchDevelop App Design	X	X	X	✓
Unit 11 Traveling Safely: Privacy and Encryption	X	X	X	✓
Unit 12 You Have Arrived: Independent Project Development	X	X	X	✓

Teaching the 6-Week Course

The course materials for *Creative Coding through Games and Apps* have been developed for flexible implementation as a 9, 12, or 18-week course. In addition, the materials can be easily adapted for implementation as a 6-week course. The table below provides guidance for adapting selected lessons from units 1 through 6 for use in a 6-week setting. Note that per-unit Teacher Guides and Students Guides developed for the longer courses do not map directly to the 6-week topics. The Teacher Guides and Student Guides for units 1-6 contain a superset of information required for the 6-week course and may require adaptation for use in this setting.

6-Week Creative Coding through Games and Apps		
Week 1		
Day	Lesson	Notes
1	U1.01_Lesson	Use lesson as-is.
2	U1.02_Lesson	Use lesson as-is.
3	U1.03_Lesson	Use lesson as-is.
4	U1.04_Lesson	Use lesson as-is.
5	U1.05_Lesson	Use lesson as-is.
Week 2		
Day	Lesson	Notes
1	U1.07_Lesson	Use lesson as-is.
2	U2.01_Lesson	Omit the brainstorm activity. Instead, introduce the *Personal Collage Project* described in U1.08_Activity.
3	U1.09_Lesson	Use lesson as-is.
4	U1.10_Lesson	Use lesson as-is.
5	U2.02_Lesson	Use lesson as-is.
Week 3		
Day	Lesson	Notes
1	U3.01_Lesson	Use lesson as-is.
2	U3.02_Lesson	Use lesson as-is.
3	U3.03_Lesson	Use lesson as-is.
4	U3.04_Lesson	Use lesson as-is.
5	U3.05_Lesson	Use lesson as-is.
Week 4		
Day	Lesson	Notes
1	U4.01_Lesson	Use lesson as-is.
2	U4.03_Lesson	Use lesson as-is.
3	U4.04_Lesson	Use lesson as-is.
4	U5.01_Lesson	Use lesson as-is.
5	U6.01_Lesson	In this lesson, students are introduced to the requirements for the final project. There are three project options to choose from: U6.01_Activity2a, U6.01_Activity2b, and U6.01_Activity2c. For the 6-week course, it is strongly recommended that students select either the 2a or 2b option. Option 2c will most likely require more class time than is allocated for the project in the 6-week course.
Week 5		
Day	Lesson	Notes
1	U6.02_Lesson	Use lesson as-is.
2	U6.03_Lesson	Use lesson as-is.
3	U6.05-10_Lesson	Use lesson as-is. Note that the U6.05-10_Lesson is designed for 6 days; use for

6-Week Creative Coding through Games and Apps		
Week 1		
Day	Lesson	Notes
		only 5 work days during weeks 5 and 6.
4	U6.05-10_Lesson	Use lesson as-is.
5	U6.05-10_Lesson	Use lesson as-is.
Week 6		
Day	Lesson	Notes
1	U6.05-10_Lesson	Use lesson as-is.
2	U6.05-10_Lesson	Use lesson as-is.
3	U6.11_Lesson	Use lesson as-is.
4	U6.12-13_Lesson	Use lesson as-is.
5	U6.12-13_Lesson	Use lesson as-is.

Teaching the 9-Week Course

Units 1 through 6, in sequence, comprise the 9-week course.

Teaching the 12-Week Course

Units 1 through 9 comprise the 12-week course. However, the sequence of units is modified. The table below describes the modified sequencing.

12-Week Creative Coding through Games and Apps	
Unit Sequence	Notes
Unit 1	Use unit as-is.
Unit 2	Use unit as-is.
Unit 3	Use unit as-is.
Unit 4	Use unit as-is.
Unit 5	Use unit as-is.
Unit 7	Use unit as-is.
Unit 8	Use unit as-is.
Unit 9	Use unit as-is.
Unit 6	Note that this unit is designed to follow units 1 through 5. In this 12-week course, students will have more programming knowledge going into this unit, and so may be held to a higher-bar for the final project.

Preparing to Teach

General Guidance

Creative Coding through Games and Apps is designed to be taught by any teacher with an interest in teaching the course, regardless of prior knowledge about coding. If you have experience in this area, you will be able to enrich the course with your expertise. But, if you do not, you will find that you can learn along with your students.

Regardless of your prior knowledge you will want to focus on two areas prior to teaching *Creative Coding through Games and Apps*:

1. **Become familiar with TouchDevelop.** If you have experience with coding, you will find that all of the concepts covered in this course are familiar, but applying them in TouchDevelop may be new to you. If you do not have experience in coding, you will want to learn basic coding concepts *and* become familiar with TouchDevelop. The following free resources will help you do both:
    - *Hour of Code with TouchDevelop* (Video-on-demand, 1 hour): https://www.microsoftvirtualacademy.com/en-US/training-courses/hour-of-code-with-touchdevelop-8401
    - *The Birth of Bot* (Video-on-demand, 8 hours): http://www.microsoftvirtualacademy.com/training-courses/codexist-the-birth-of-bot
    - *Bot Levels Up* (Video-on-demand, 5 hours): http://www.microsoftvirtualacademy.com/training-courses/learn-to-code-with-codexist-bot-levels-up
    - *TouchDevelop – Programming on the Go* (Book, 271 pages): https://www.touchdevelop.com/docs/book
2. **Integrate TouchDevelop into your classroom.** The TouchDevelop website has many useful tips about integrating TouchDevelop into your classroom: https://www.touchdevelop.com/docs/teach.

If you are a teacher with no prior knowledge about coding, you will want to focus on two additional areas prior to teaching *Creative Coding through Games and Apps*:

1. **Have a beginner's mindset.** Learning alongside your students is not something that all teachers are comfortable with. However, doing so reflects the real world of software development. The world of technology is changing at an incredibly rapid pace. Today's experts are tomorrow's dinosaurs. The one true skill in this world is the ability to learn. By learning alongside your students, by *not* being an authority on the subject, and by encouraging students to arrive at answers on their own, you may provide a very valuable experience, one that reflects the nature of the real world and that will serve them well in any future endeavor.
2. **Learn coding fundamentals**. As valuable as having a beginner's mindset may be, we all want to be at least one step ahead of our students. You should find that TouchDevelop tutorials and resources in each lesson allow you to do just that. Additionally, as a head start to the course, the resources above will give you a great jump on learning coding fundamentals.

Teaching with TouchDevelop

To orient you and your students to TouchDevelop, the following online lessons are integrated into the course:

- http://aka.ms/WelcomeToTouchDevelop
- http://aka.ms/ExploringTouchDevelop
- http://aka.ms/CodingInTouchDevelop
- http://aka.ms/BeyondBasicsInTouchDevelop

How do I integrate TouchDevelop into my classroom?

The TouchDevelop website has many resources for integrating TouchDevelop into your classroom, including information about joining a teacher's group and managing your classroom: https://www.touchdevelop.com/docs/teach.

One integration point that will require some forethought has to do with reviewing students work. The simplest way to do this is to have students publish finished scripts and share the script URL with you. So that you can identify the student who created the script, require that students put their names in the script description. Another option is to create a TouchDevelop group for your class and have students share finished scripts with the group. Information about TouchDevelop groups can be found here: https://www.touchdevelop.com/docs/groups.

Of note when using TouchDevelop to teach and learn

As you learn TouchDevelop and become familiar with using it as a teaching tool, these are important things to keep in mind.

- **TouchDevelop changes.** The engineers at Microsoft Research are continually looks for ways to make TouchDevelop more intuitive and easier to use. This means that you may see subtle changes in the TouchDevelop interface from time to time. These changes generally do not affect functionality – you just need to be aware that changes do occur.
- **TouchDevelop is designed for touch devices.** Some gestures in TouchDevelop may seem awkward when using a keyboard and mouse. This is generally because TouchDevelop has been developed first and foremost for use with touch screens (i.e. tablets and smart phones).
- **Students can upload and share any image.** This means that students can upload and share copyrighted and/or offensive images to use in their games and apps. This places the burden of discretion on students. This is best addressed as a teaching opportunity: Make sure students think twice and understand the potential impact when uploading or using a shared image. To report abuse, follow the *report abuse* link in a published script. For more information about using art in TouchDevelop scripts, see https://www.touchdevelop.com/app/beta#list:topics:topic:art:overview.
- **Students can share and copy each other's work.** This can be extremely beneficial, allowing students to learn from each other and build upon each other's work. However, it also presents an opportunity for plagiarism. For major projects, it is important to monitor students work throughout the development process to insure that a student's work is his/her own and that he/she understands the code.
- **TouchDevelop has different skill levels.** TouchDevelop has 3 different skill levels: *beginner*, *coder*, and *expert*. Each level exposes increasingly complex functionality and

features. The TouchDevelop tutorials for this course are designed to be done in *coder* mode and will change a student's skill level accordingly. Students may change between skill levels by clicking "Skill level" from the TouchDevelop hub. However, the coder skill level provides the best set of functionality and features for middle and high school students.

- **Always sign in/sign out on shared computers.** TouchDevelop stores data locally on a per user basis. This can pose issues when students are using shared computers. To make sure students have access to their account information, students should always sign in and sign out when using a shared computer. Another way to make sure that students access only their own account data is to do the following:
  - Have students open a browser in private mode. In Internet Explorer, this is called "InPrivate browsing" mode. In Chrome, it is called "incognito" mode. In Firefox, it's simply called "private" mode.
  - Students then sign in to their TouchDevelop accounts.
  - When the browser is closed, the student is automatically signed out.
- **TouchDevelop has a thriving community.** Encouraging students to investigate and participate in this community can be a valuable part of this course.
  - **Forums**: Encourage students to ask questions in the TouchDevelop forums: https://www.touchdevelop.com/app#list:comments:forum.
  - **Community-created tutorials**: Encourage students to investigate the try any of the many excellent tutorials created by the TouchDevelop community: https://www.touchdevelop.com/app/#list:topics:topic:tutorials:overview
  - **Built-in, community driven documentation**: Using documentation is an important part of any software developer's job. Encourage students to learn from the built-in, community driven documentation: https://www.touchdevelop.com/docs/contents. When students don't find what they need in the documentation, have them create documentation after they have learned something new: https://www.touchdevelop.com/docs/addyourowndocs.
- **Students can skip instructional videos in tutorials.** TouchDevelop does not force students to watch instructional videos that are embedded in tutorials. Students can skip the video instruction and go directly to the programming portions of tutorials. However, by doing so, students will miss very important information – concepts, details, and context that are critical to their success in the course. For this reason, you should emphasize the importance of watching these instructional videos in tutorials. In fact, it's worth encouraging students to re-play videos to make sure they grasp concepts. Finally, it is worth noting that these videos can be played in full-screen mode.
- **You can print code for TouchDevelop scripts.** While online lessons and working in an online environment have their benefits, it is sometimes nice to have something tangible to share and work with. TouchDevelop makes it possible to do this. You can print the code for a script by navigating to a script and clicking the printer icon. A new browser tab will open with nicely formatted, ready-to-print code.

Flexible Teaching Options

Creative Coding through Games and Apps enables flexibility in teaching approach by providing a combination of online and in-class instructional resources and learning activities. The combination of online and in-class activities allows educators to choose the right balance for their students of in-class instruction and out-of-class study. Because TouchDevelop is an online programming environment accessible from any browser, tutorials and videos can be part of outside-of-the-classroom learning experiences, when appropriate. In-class time can be devoted to providing more guided instruction and extensive collaborative projects. Or the online tutorials may be presented during class time to provide a common foundation of direct instruction shared by all students. In addition, the sequence of short-duration activities provided within any given 50-minute lesson is easy to adapt to longer or shorter class periods and block schedules.

A Typical Cycle of Learning in Creative Coding with Games and Apps

The twelve course units average five or six 50-minute lessons per unit. Each unit provides a blend of online, immersive learning with unplugged, group, and independent activities. The units typically progress along a trajectory in which students learn fundamental programming and computational thinking concepts through a variety of teaching strategies and learning experiences, threading direct instruction with hands-on practice and open-ended creation as appropriate.

Individual weeks of instruction typically follow this learning flow:

Day 1: Introductory activity that places the concepts within the students' life experiences

Day 2: Online guided tutorials

Day 3: Additional activities that reinforce the concepts through inquiry-based activities

Days 4 & 5: Collaborative projects applying the concepts to solving a problem in TouchDevelop

Assigning Independent Study

Independent study of some course learning objectives may be assigned for suitably capable and motivated students. At teacher discretion, independent study may be enabled for select portions of the course through assignment of the online lessons and associated Student Guide resources included in each unit, along with the project assignments for Units 6 and 12.

Glossary

- **algorithm** – A set of (often repeated) steps used to solve a problem. The set of steps for doing long division of number is an algorithm.
- **app** – In TouchDevelop, an a script created from the *blank app* template and consisting of one or more pages. An app is different than a game in that it is created from a different template, is made up of one or more pages that can be navigated, and includes libraries that make user interaction with text and buttons easy. In general, an app is a computer program often designed for a particular purpose and often designed for mobile devices.
- **array** – See **collection**.
- **assign** – To set the value of a **variable**. The symbol for assign in TouchDevelop is :=. The following code assigns a value of 5 to the variable x: `x := 5`.
- **Boolean** – A **variable type** that can be either true or false. A Boolean condition is a condition that evaluates to either true or false.
- **the cloud** – The group of computers that are accessible via the Internet and are used for remote storage and computation.
- **cloud computing** – Computing done in **the cloud**.
- **cloud variable** – a variable whose value is stored in **the cloud**. Also see **replicated variable**.
- **collection** – Also known as an **array** in other programming languages. A group of objects, all of the same **variable type**, that can be referenced as a single variable. Objects in a collection are referred to as **elements** or items. An individual element can be referenced by its **index**.
- **computer program** – A set of instructions that a computer can follow. Apps and games are examples of computer programs. In TouchDevelop, a computer program is called a **script**.
- **concatenate** – To put two strings end-to-end to form a single string. For example, the concatenation of “Hello” and “world” is “Helloworld”. The symbol for concatenate in TouchDevelop is two vertical lines.
- **conditional statement** – Also known as an *if/then* or *if/then/else* statement. A programming construct used for making decisions within a program. A conditional statement allows for execution of code based on the evaluation of a condition.
- **custom function** – A function created by the programmer (as opposed to functions associated with objects). A custom function usually has a specific purpose and can be/is used in multiple places within a program.
- **debug** – The process of correcting errors within a program. i.e. The process of removing “bugs” from a program.
- **declare** – To create a **variable**. Also see **initialize**.
- **decrypt** – The process of making **encrypted** data readable.
- **element** – An individual item in a **collection**.
- **encrypt** – The process of making data unusable to anyone who doesn’t understand how to decrypt the data. Encrypting data usually involves using an algorithm, known as

the *encryption key*, to map characters to other characters in such a way that the data is only usable if the encryption key is known and used to decrypt the data.

- **event** – Something that happens outside a program (like a screen tap or mouse click) that the program can respond to.
- **for loop** – A programming construct that allows for a block of code to be executed a specified number of times.
- **for each loop** – A programming construct that allows for a block of code to be executed one time for each object in a collection, table, or other data structure with multiple elements. For example, a **for each** loop allows for a block of code to be executed for each element in a collection or for each row in a table.
- **Frayer Model** – A graphical tool for building definitions. For more information, see http://www.readingeducator.com/strategies/frayer.htm.
- **function** – A self-contained set of instructions for performing a specific task within a **computer program**. Most objects have multiple functions associated with them. For example, the *turtle* object has a *forward* function: `turtle->forward(100)`. Also see **custom function**.
- **game** – In TouchDevelop, a script created using the *blank game* template, with a player earning points and losing lives based on the logic of the script. A game is different than an **app** in that it is created from a different template and includes libraries for making movement and scoring easy.
- **genre** – A category of game types.
- **index** – A numerical value that corresponds to an element in a collection or table. Index values start at zero, so the first element in an array has an index value of 0.
- **initialize** – To set the value of a **variable** for the first time. In TouchDevelop, variable **declaration** and initialization usually happen at the same time. The following code declares a variable named x and initializes its value to 5: `var x := 5`. When a variable is initialized, its **variable type** is also determined. In the example shown, the variable type for x is *number*. i.e. The variable x can only be **assigned** values that are of variable type *number*.
- **loop** – See **for loop** or **for each loop**. In general, a programming construct that allows for a block of code to be repeated multiple times.
- **loop counter** – The variable used in a **for loop** to determine the number of times the loop will execute.
- **object** –A fundamental building block for any **computer program**, designed to hole data and allow for manipulation of that data through **functions** and **properties**.
- **parameter** – Data passed to a **function**. In the following line of code, 100 is a parameter: `turtle->forward(100)`.
- **program** – See **computer program**.
- **replicated variable** – A variable whose value is replicated (i.e. copied or stored) in **the cloud**.
- **script** – See **computer program**.

- **sprite** – An **object** that is usually a character in a game. Sprites have **properties** and **functions** that are used to control their look and behavior.
- **sprite sheet** – A single file with multiple images that, when viewed in rapid succession, create an animation.
- **storyboard** - A visual, frame-by-frame plan for a program or project. A storyboard often contains sketches and descriptions that tell the story of a game or project.
- **string** – A sequence of alphanumeric characters and/or symbols. In a **computer program**, a string is indicated by quotation marks (“”) at the beginning and end of the string. For example, “Hello Bot!” is a string made up of 10 characters (5 letters, a space, and an exclamation mark). String can also be used to refer the the **variable type** of a **variable**. If a variable is initialized to a string (e.g. `var x := “Hello Bot!”`), its variable type is string.
- **table** – In TouchDevelop, a table is a data structure that contains rows and columns. All data in each column is of the same **variable type** and each row can be referenced by its **index**.
- **variable** – A container for data. Every variable has a name that is used to reference the data that it contains. Every variable also has a **variable type**.
- **variable type** – The type of data that a **variable** can contain. Examples of variable types are *number*, *string*, and *sprite*. In TouchDevelop, the variable type for a variable is determined when the variable is **initialized**. Once the variable type for a variable has been determined, the variable can only contain data of the same type.
- **variable scope** – The part of a program where a variable can be read. For example, a variable declared in one function is said to be a *local* variable, and cannot be read from other functions. However, variables can be declared with a *global* scope, making them readable in all functions of a program.