This curriculum may be modified through varying techniques, strategies, and materials, as per an individual student’s Individualized Educational Plan (IEP).

Approved by the Berkeley Heights Board of Education
at the regular meeting held on 12/5/19.
VISION STATEMENT

STEM is the integrated approach to education in the areas of Science, Technology, Engineering, and Mathematics. Instruction is student centered and driven by an iterative design process, exploratory learning, problem-solving, and engagement in authentic contexts.

Through the process of engaging in authentic, hands-on, open-ended design challenges, students will become familiar with the steps and processes associated with successful problem solving in the context of the engineering design process. Students will gain proficiency in the application of relevant Math, Science, and Technology concepts while expanding their comprehension and understanding of the human-designed world, the nature of technology and engineered systems, and the skills, knowledge, and attitudes necessary to become well-rounded and successful twenty-first century problem solvers and innovators.
MISSION STATEMENT

Video Game Design is a one quarter cycle class designed for 7th and 8th grade students at Columbia Middle school that addresses 21st century skills, career ready practices, and technology standards. Students are introduced to a problem solving and design process and given the opportunity to utilize these as they design an original video game.

Advanced Video Game Design is a one quarter cycle class designed for 8th grade students at Columbia Middle school that addresses 21st century skills, career ready practices, and technology standards. Students are empowered to delve deeper and explore more advanced programming concepts/techniques as they create an original game. As part of this, students utilize literacy skills to develop robust plot lines and characters. At the conclusion of the course, students compile their work into a polished video game that they can publish and share.

Throughout these courses, students will be engaged in an authentic problem-based learning environment working as a team. Students will learn to utilize feedback from peers to revise their designs and develop better solutions. Throughout these courses students gain programming skills using code.org's CS Discoveries framework. They learn to represent computing processes both on and offline and consider the impact of technology in the world. Additionally, these courses lay the foundational knowledge needed to solve real world problems across different areas of study and outside the classroom.
Having a strategy for approaching problems can help you develop new insights and come up with new and better solutions. This process is generally useful for solving all kinds of problems.

The Problem Solving Process
Define
● What problem are you trying to solve?
● What are your constraints?
● What does success look like?
Prepare
● Brainstorm / research possible solutions
● Compare pros and cons
● Make a plan
Try
● Put your plan into action
Reflect
● How do your results compare to the goals you set while defining the problem?
● What can you learn from this or do better next time?
● What new problems have you discovered?

Similarly, the design process is a loop because although the steps are listed in sequential order, you will likely return to previous steps multiple times throughout a project. It is often necessary to revisit stages or steps in order to improve that aspect of a project.

In the design process,
● Ask (What are we trying to solve?, What are the constraints?, What are the requirements?, What questions do you have about the challenge?)
● Imagine (What are the possible solutions?, Brainstorm ideas, list materials needed, explain the ideas, create a sketch for you ideas)
● Plan (Choose a final solution, sketch a final solution, decide the steps you will take to create your solution, create a technical drawing to explain your design)
● Create (follow your plan and create your design, what changes did you make while creating your design? Why?, Self reflection)
● Improve ( What worked well? What could have gone better? What improvements could you make to allow your design to be more successful? Why is the redesign better than the original design?)
Career Ready Practices

CRP1. Act as a responsible and contributing citizen and employee.
CRP2. Apply appropriate academic and technical skills.
CRP4. Communicate clearly and effectively and with reason.
CRP5. Consider the environmental, social and economic impacts of decisions.
CRP6. Demonstrate creativity and innovation.
CRP7. Employ valid and reliable research strategies.
CRP8. Utilize critical thinking to make sense of problems and persevere in solving them.
CRP9. Model integrity, ethical leadership and effective management.
CRP11. Use technology to enhance productivity.
CRP12. Work productively in teams while using cultural global competence.
STUDENT PROFICIENCIES

Students will understand:

- Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
- Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.
- Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation and review stages of program development.
- Describe how internal and external parts of computing devices function to form a system.
- Model how computer hardware and software work together as a system to accomplish tasks.
- Use flowcharts and/or pseudocode to address complex problems as algorithms.
- Systematically test and refine programs using a range of test cases.
- Compare trade-offs associated with computing technologies that affect people’s everyday activities and career options.
- Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.
- Design projects that combine hardware and software components to collect and exchange data.
- Seek and incorporate feedback from team members and users to refine a solution that meets user needs.
- Discuss issues of bias and accessibility in the design of existing technologies.
- Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
- Document programs in order to make them easier to follow, test, and debug.
- Create clearly named variables that represent different data types and perform operations on their values.
- Incorporate existing code, media, and libraries into original programs, and give attribution.
- Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
METHODS OF EVALUATION

1. Teacher observation/questioning/monitoring
2. Project Notebooks/Journals
3. Team evaluation rubrics
4. Self and peer evaluation
5. Performance tasks/assessments
6. Reports and presentations
7. Student created designs and models
8. Final presentations
MODIFICATIONS & ACCOMMODATIONS

Modifications and Accommodations for Special Education students, students with 504s, English Language Learners and Gifted and Talented students may include but are not limited to the following:

**Special Education**
- Individualized Education Plans (IEPs)
- Exemplars of varied performance levels
- Multimedia presentations
- Sheltered instruction
- Consultation with ESL teachers
- Manipulatives
- Tiered/Scaffolded Lessons
- Mnemonic devices
- Visual aids
- Modeling
- Guided note-taking
- Study Guides
- Modified homework
- Differentiated pre-typed class notes and example problems
- Use of the special education teacher to re-instruct in flexible small groups for the struggling learner
- Manipulatives
- Flipped Instruction
- Word banks
- Reduced choice on assessments
- Preferential seating
- Choice activities
- Modified time requirements
- Modified notes
- Modified lesson, assessment and study guide format
- Provide an enriched curriculum and activities
- Independent projects
- Contracts/behavior support plans
- Open-ended responses
- Project-based learning
- Group activities
Guided Notes
Functional learning incorporated into each lesson
Exploration Activities
Assessment read aloud
Small group assessments
Organizational Support
Oral questioning assessments to supplement written response
Pre-writing Structural Supports for extended writing tasks
Ongoing teacher feedback as part of the writing process
Interactive Study Guides
Multi-sensory approach to instruction
Written and spoken step-by-step directions
Content-focused assessment (not grading for spelling/grammar)
Graphic organizers
Non-verbal cues to begin task/remain on task/refocus
Individual monitoring for understanding/reinforced instruction
Printed copies of class readings for application of Active Reading Strategies

Gifted & Talented
Provide one-to-one teacher support
Curriculum Compacting
Advanced problems to extend the critical thinking skills of the advanced learner
Supplemental reading material for independent study
Elevated questioning techniques using Webb’s Depth of Knowledge matrix
Curriculum Compacting
Flexible grouping
Tiered assignments
Topic selection by interest
Manipulatives
Tiered Lessons
Flipped Instruction
Multimedia Presentations
Open-ended responses
Project-based learning
Group activities
Guided Notes
Conclusions and analysis of exploratory activities
Career based learning incorporated into each lesson
Exploration Activities
• Student choice

**ELLs**

• Exemplars of varied performance levels
• Multimedia presentations
• Sheltered instruction
• Consultation with ESL teachers
• Manipulatives
• Tiered/Scaffolded Lessons
• Mnemonic devices
• Visual aids
• Modeling
• Guided note-taking
• Study Guides
• Modified homework
• Differentiated pre-typed class notes and example problems
• Individualized instruction plans
• Manipulatives
• Flipped Instruction
• Words banks
• Reduced choice on assessments
• Preferential seating
• Choice activities
• Modified time requirements
• Modified notes
• Modify lesson, assessment and study guide format
• Provide an enriched curriculum and activities
• Contracts/management plans
• Open-ended responses
• Project-based learning
• Group activities
• Guided Notes
• Exploration Activities
• Assessment read aloud
• Small group assessments
• Oral questioning assessments to supplement written response
• Pre-writing Structural Supports for extended writing tasks
• Ongoing teacher feedback as part of the writing process
• Interactive Study Guides
• Multi-sensory approach to instruction
• Written and spoken step-by-step directions
• Graphic organizers
• Non-verbal cues to begin task/remain on task/refocus
• Individual monitoring for understanding/reinforced instruction
• Printed copies of class readings for application of Active Reading Strategies

504s
• Exemplars of varied performance levels
• Multimedia presentations
• Sheltered instruction
• Tiered/Scaffolded Lessons
• Mnemonic devices
• Visual aids
• Modeling
• Guided note-taking
• Study Guides
• Differentiated pre-typed class notes and example problems
• Manipulatives
• Words banks
• Reduced choice on assessments
• Preferential seating
• Modified time requirements
• Modified notes
• Modify lesson, assessment and study guide format
• Modified homework
• Independent projects
• Contracts/management plans
• Open-ended responses
• Project-based learning
• Group activities
• Guided Notes
• Exploration Activities
• Assessment read aloud
• Small group assessments
• Organizational Support
• Oral questioning assessments to supplement written response
• Pre-writing Structural Supports for extended writing tasks
• Ongoing teacher feedback as part of the writing process
• Interactive Study Guides
• Multi-sensory approach to instruction
• Written and spoken step-by-step directions
• Content-focused assessment (not grading for spelling/grammar)
• Graphic organizers
- Non-verbal cues to begin task/remain on task/refocus
- Individual monitoring for understanding/reinforced instruction
- Printed copies of class readings for application of Active Reading Strategies

**Students at Risk of Failure**
- Exemplars of varied performance levels
- Multimedia presentations
- Tiered/Scaffolded Lessons
- Modeling
- Guided note-taking
- Study Guides
- Differentiated pre-typed class notes and example problems
- Individualized instruction plans
- Words banks
- Reduced choice on assessments
- Preferential seating
- Choice activities
- Modified time requirements
- Modified notes
- Modified lesson, assessment and study guide format
- Modified homework
- Provide an enriched curriculum and activities
- Contracts/management plans
- Open-ended responses
- Project-based learning
- Group activities
- Guided Notes
- Exploration Activities
- Assessment read aloud
- Small group assessments
- Oral questioning assessments to supplement written response
- Pre-writing Structural Supports for extended writing tasks
- Ongoing teacher feedback as part of the writing process
- Interactive Study Guides
- Multi-sensory approach to instruction
- Written and spoken step-by-step directions
- Graphic organizers
- Non-verbal cues to begin task/remain on task/refocus
- Individual monitoring for understanding/reinforced instruction
- Printed copies of class readings for application of Active Reading Strategies
Unit 1: Introduction to the Problem Solving Process

**Duration:** 4 days

**Overview:** This unit guides students to develop and adopt a more formal structured problem solving process by reflecting on problems they have encountered, both in the classroom and everyday life. By working through a diverse set of problems, such as logic puzzles, engineering challenges, and planning a trip, students learn to identify different classes of problems, decompose large problems, and develop their personal problem solving skills.

**Standards:** 8.2.8.C.1; 8.2.8.C.2; 8.2.8.C.4; 8.2.8.C.5

**21st Century:** CRP2; CRP4; CRP6; CRP8; 9.3.ST-ET.3; 9.3.ST-SM.2; 9.3.IT-PRG.5; 9.3.IT-PRG.6; 9.3.IT-PRG.7

**Cross-Curricular:** RST.6-8.3; RST.6-8.4; RST.6-8.9; RST.6-8.10; MP.1; MP.2; MP.4; MS-ETS1-2; MS-ETS1-3; MS-ETS1-4

**Essential Questions:**
- What strategies and processes can I use to become a more effective problem solver?

**Students Learning Objectives:**
*Students will know and be able to...*
- Communicate and collaborate with classmates in order to solve a problem
- Iteratively improve a solution to a problem
- Identify different strategies used to solve a problem
- Identify the four steps of the problem solving process
- Given a problem, identify individual actions that would fall within each step of the problem solving process
- Identify useful strategies within each step of the problem solving process
- Apply the problem solving process to approach a variety of problems
- Assess how well-defined a problem is and use strategies to define the problem more precisely

**Possible Activities:**
- The students work in groups to design aluminum foil boats that will support as many pennies as possible. They then reflect on their experiences with the activity and make connections to the types of problem solving they will be doing for the rest of the course.
• Examine the formal problem solving process that the class will use over the course of the year. (Define - Prepare - Try - Reflect). They then relates these steps to the aluminum boat problem from the previous lesson, then a problem they are good at solving, then a problem they want to improve at solving. The class collects a list of generally useful strategies for each step of the process to put on posters that will be used throughout the unit and year.

• The students apply the problem solving process to three different problems: a word search, a seating arrangement for a birthday party, and planning a trip. The problems grow increasingly complex and poorly defined to highlight how the problem solving process is particularly helpful when tackling these types of problems.
Unit 2: Computers and Problem Solving
Duration: 7 days

Overview: Students move on to thinking about computers as machines that solve information problems. Students begin by building a common definition for a computer that focuses on functionality instead of specific hardware. They then explore the ways that computers approach problems. For their final project, students propose an app that could be used to solve a problem of their choosing.

Standards: 8.2.8.E.2; 8.2.8.E.3; 8.2.8.C.3; 8.2.8.C.8

21st Century: CRP2; CRP4; CRP6; CRP8; 9.3.ST-ET.3; 9.3.ST-SM.2; 9.3.IT-PRG.5; 9.3.IT-PRG.6; 9.3.IT-PRG.7

Cross-Curricular: RST.6-8.3; RST.6-8.4; RST.6-8.9; RST.6-8.10; MP.1; MP.2; MP.4; MS-ETS1-2; MS-ETS1-3; MS-ETS1-4

Essential Questions:
- How do computers help people to solve problems?
- How do people and computers approach problems differently?
- What does a computer need from people in order to solve problems effectively?

Students Learning Objectives:
Students will know and be able to...
- Identify a computer as a machine that processes information
- Provide a high level description of the different parts of the Input - Output - Store - Process model of a computer
- Identify the inputs and outputs of common computing devices
- Select the inputs and outputs used to perform common computing tasks
- Define processing as the work done (possibly by a computer) to turn an input into an output
- Define an algorithm as the series of commands a computer uses to process information
- Develop and iteratively improve an algorithm for processing information based on given constraints
- Provide examples of common types of information that is stored on a computer
- Explain the need for storage as part of processing information with a computer
- Develop an algorithm that incorporates storage considerations
- Describe how information can be processed to solve a particular problem.
- Identify the information an app would need to be provided as input in order to produce a given output
- Identify and define a problem that could be solved using computing
- Design an app that inputs, outputs, stores, and processes information in order to solve a problem
- Provide and incorporate targeted peer feedback to improve a computing artifact
Possible Activities:

- Students develop a preliminary definition of a computer. They then work in groups to sort pictures into “is a computer” or “is not a computer” and explain their motivations for choosing some of the most difficult categorizations. The students examine the definition of a computer and begin revising their posters according to the new definition.

- Students consider a number of computing devices to determine what types of inputs and outputs they use. Groups are assigned to a computing device and list the inputs and outputs of their device. They then examine common activities they do on a computing device and select the inputs and outputs used for that activity.

- Pairs work together to put a deck of cards in order, a form of processing information. In the end, the students discuss what processing means within the context of solving information problems.

- The students focus on the storage component of the definition of a computer, within the content of processing information. The class spends the majority of the lesson developing and sharing algorithms to process information, with an emphasis on how much storage is needed for any particular algorithm. Finally, the students discuss the importance of storage while processing information.

- The students evaluate various web applications to analyze the specific problems that they were designed to solve, the inputs that they need to work, and the outputs they provide to users. The students make observations of these apps and discuss the impact of apps on society.

- The students work in pairs to propose app designs to solve real-world problems. This project culminates in a poster presentation highlighting the features of each app.
Unit 3: Images and Animations

Duration: 15 days

Overview: Students build up toward programming interactive animations in the Game Lab environment. They begin with simple shapes and sprite objects, then use loops to create flipbook style animations. Next, they learn to use booleans and conditionals to respond to user input. At the end of the chapter, students design and create an interactive animation that they can share with the world.

Standards: 8.1.8.A.3; 8.1.12.B.2; 8.1.8.D.3; 8.2.8.D.1

21st Century: CRP2; CRP4; CRP6; CRP8; 9.3.ST-ET.3; 9.3.ST-SM.2; 9.3.IT-PRG.5; 9.3.IT-PRG.6; 9.3.IT-PRG.7

Cross-Curricular: RST.6-8.3; RST.6-8.4; RST.6-8.9; RST.6-8.10; MP.1; MP.2; MP.4; MS-ETS1-2; MS-ETS1-3; MS-ETS1-4

Essential Questions:
- What is a computer program?
- What are the core features of most programming languages?
- How does programming enable creativity and individual expression?
- What practices and strategies will help me as I write programs?

Students Learning Objectives:
Students will know and be able to...
- Identify how Computer Science is used in a field of entertainment
- Reason about locations on the Game Lab coordinate grid
- Communicate how to draw an image accounting for shape position, color, and order
- Use the Game Lab IDE to plot different colored shapes on the screen.
- Sequence code correctly to overlay shapes.
- Debug code written by others.
- Use and reason about drawing commands with multiple parameters
- Generate and use random numbers in a program
- Identify a variable as a way to label and reference a value in a program
- Use variables in a program to store a piece of information that is used multiple times
- Reason about and fix common errors encountered when programming with variables
- Assign a sprite to a variable and use dot notation to update a sprite's properties
- Create a static scene combining sprites, shapes, and text
- Explain what an animation is and how it creates the illusion of smooth motion
- Explain how the draw loop allows for the creation of animations
- Use the draw loop in combination with the randomNumber() command, shapes, and sprites to make simple animations

Berkeley Heights Public Schools
• Describe the connection between updating a sprite's location properties and sprite movement on the screen.
• Read and follow the steps of a short program written in pseudocode that manipulates variable values.
• Use the counter pattern to increment or decrement sprite properties
• Identify which sprite properties need to be changed, and in what way, to achieve a specific movement
• Organize objects based on simple and compound boolean statements
• Describe the properties of an object using boolean statements and predict outputs
• Use conditionals to react to changes in variables, sprite properties and keyboard input
• Move sprites in response to keyboard input
• Use an else statement as the fallback case to an if statement
• Differentiate between conditions that are true once per interaction, and those that remain true through the duration of an interaction.
• Use conditionals to react to keyboard input or changes in variables / properties
• Sequence commands to draw in the proper order
• Apply an iterator pattern to variables or properties in a loop

Possible Activities:
• Students consider the "problems" of boredom and self expression, and reflect on how to approach those problems in their own lives. They then explore how computer science plays a role in either a specific form of entertainment or as a vehicle for self expression.
• Students explore the challenges of communicating how to draw with shapes and investigate how this problem is approached in Game Lab. Students work together to use Game Lab to interactively place shapes on Game Lab's 400 by 400 grid and then take turns instructing each other how to draw a hidden image using its tools
• The students explore Game Lab Programming Environment further and begin to use it to position shapes on the screen. Students investigate the concepts of sequencing and debugging, as well as a few simple commands. Finally the students create an online version of an image.
• Students extend their drawing skills to include width, height, and the concept of random number generation. They learn to draw with versions of the ellipse() and rect() function that include width and height parameters and to use the background() block to fill the screen with color.
• Introduce students to variables as a way to label a number in a program or save a randomly generated value. They begin with a very basic description of the purpose of a variable and practices using the new blocks. They then use variables to save a random number, allowing the programs to use the same random number multiple times.
• Students are introduced to the sprite object. Students examine how sprites can be assigned an image to show, and can keep track of multiple values about themselves. Students demonstrate their understanding by creating a scene using sprites.
• Students combines the draw loop with random numbers to manipulate some simple animations with dots and then with sprites. Afterwards, students use what they learned to update the sprite scene from earlier.
• Students use notecards and string to simulate variables within a program and execute a few short programs. They students then uses the same process with sprite properties, tracking a sprite's progress across the screen.
• The students combine the Draw Loop and the Counter Pattern to writes programs that move sprites across the screen, as well as animate other sprite properties.
• Students are introduced to boolean values, logic, and conditional statements. The students starts offline by playing a simple game in which the boolean (true/false) statements describe personal properties (hair or eye color, age, etc). The students then groups objects based on increasingly complex boolean statements and look at how conditionals can impact the flow of a program.
• The students use booleans to compare the current value of a sprite property with a target value (ex: determine when a sprite has reached a point on the screen, grown to a given size, etc.). They then add conditional if-statements to write code that responds to those boolean comparisons.
• Students are introduced to a new block called keyDown() which returns a boolean and can be used in conditional statements to move sprites around the screen. Students will then write programs that take keyboard input from the user to control sprites on the screen.
• The students explore more complex ways to use conditional statements to take user input as well as mouse inputs.
• The students culminate the unit by planning for and developing an interactive greeting card using all of the programming techniques they've learned to this point.
Unit 4: Programming

Duration: 15 days

Overview: In this unit students combine the constructs that they learned in the first unit to program more complex movement and collisions in their sprites. As they create more complex programs, they begin to use functions to organize their code. In the end, students use a design process to create an original game.

Standards: 8.2.8.A.2; 8.2.8.A.2; 8.2.8.C.1; 8.2.8.C.2; 8.2.8.C.3; 8.2.8.C.4; 8.2.8.C.5; 8.2.8.E.2; 8.2.8.E.3; 8.2.8.E.4

21st Century: CRP2; CRP4; CRP6; CRP8; 9.3.ST-ET.3; 9.3.ST-SM.2; 9.3.IT-PRG.5; 9.3.IT-PRG.6; 9.3.IT-PRG.7

Cross-Curricular: RST.6-8.3; RST.6-8.4; RST.6-8.9; RST.6-8.10; MP.1; MP.2; MP.4; MS-ETS1-2; MS-ETS1-3; MS-ETS1-4

Essential Questions:
● How do software developers manage complexity and scale?
● How can programs be organized so that common problems only need to be solved once?
● How can I build on previous solutions to create even more complex behavior?

Students Learning Objectives:
Students will know and be able to...
● Use the velocity and rotationSpeed blocks to create and change sprite movements
● Describe the advantages of simplifying code by using higher level blocks
● Use the isTouching block to determine when two sprites are touching
● Describe how abstractions help to manage the complexity of code
● Use sprite velocity with the counter pattern to create different types of sprite movement
● Explain how individual programming constructs can be combined to create more complex behavior
● Use the `displace`, `collide`, `bounce`, and `bounceOff` blocks to produce sprite interactions
● Describe how abstractions can be built upon to develop even further abstractions
● Create and use functions for blocks of code that perform a single high-level task within a program
● Create and use functions to remove repeated blocks of code from their programs and improve the readability of their programs
● Explain how abstractions allow programmers to reason about a program at a higher level
● Identify core programming constructs necessary to build different components of a game
- Create and use multi frame animations in a program
- Implement different features of a program by following a structured project guide
- Independently scope the features of a piece of software
- Create a plan for building a piece of software by describing its major components
- Implement a plan for creating a piece of software

**Possible Activities:**

- Introduce students to the properties that set velocity and rotation speed directly and have them use these new properties in different ways, thereby building up the skills they need to create a basic side scroller game.
- Pairs explore how a computer could use sprite location and size properties and math to detect whether two sprites are touching. They then use the `isTouching()` block to create different effects when sprites collide. Lastly, they use their new skills to improve the sidescroller game that they started previously.
- Combine the velocity properties of sprites with the counter pattern to create more complex sprite movement, such as simulating gravity, making a sprite jump, and allowing a sprite to float left or right. They then combine these movements to animate and control a single sprite and build a simple game in which a character flies around and collects coins.
- Brainstorm other ways that two sprites could interact and then use `isTouching` to make one sprite push another across the screen before practicing with the four collision commands (collide, displace, bounce, and `bounceOff`).
- Investigate functions as a way to organizing code. They discuss how abstract steps make it easier to understand and reason about steps and begins to create functions in Game Lab. Finally they use these skills to organize and add functionality to the final version of their side scroller game.
- Examine and utilize multi-frame animations in Game Lab. They can then make improvements to a game to make it their own.
- The students use the problem solving process to create a platform jumper game. The students design their game and use a structured process to build them. Finally, they reflect on how the games could be improved, and implements those changes.
- The class plans and builds original games using the project guide they created. In small groups the students plan, develop, and give feedback on the games they create.
Unit 5: ADVANCED VIDEO GAME DESIGN

Duration: Full Marking Period Cycle

Overview: Students will plan and build their own game using the framework from the introductory class to guide their project. Working individually or in pairs, students will first decide on the type of game they’d like to build, taking as inspiration a set of sample games. They will then complete a blank project guide where they will describe the game’s behavior and scope out the variables, sprites, and functions they’ll need to build. In Code Studio, a series of levels prompts them on a general sequence they can use to implement this plan. Partway through the process, students will share their projects for peer review and will incorporate feedback as they finish their game. At the end of the lesson, students will share their completed games with their classmates.

Standards: 8.2.8.A.2; 8.2.8.A.2; 8.2.8.C.1; 8.2.8.C.2; 8.2.8.C.3; 8.2.8.C.4; 8.2.8.C.5; 8.2.8.E.2; 8.2.8.E.3; 8.2.8.E.4; 8.2.8.D.2; 8.2.8.C.7

21st Century: CRP2; CRP4; CRP6; CRP8; 9.3.ST-ET.3; 9.3.ST-SM.2; 9.3.IT-PRG.5; 9.3.IT-PRG.6; 9.3.IT-PRG.7

Cross-Curricular: RST.6-8.3; RST.6-8.4; RST.6-8.9; RST.6-8.10; MP.1; MP.2; MP.4; MS-ETS1-2; MS-ETS1-3; MS-ETS1-4

Essential Questions:
● How do software developers manage complexity and scale?
● How can programs be organized so that common problems only need to be solved once?
● How can I build on previous solutions to create even more complex behavior?

Students Learning Objectives:
Students will know and be able to...
● Develop a backstory for the game and develop a game description
● Sketch out visuals and digitize these using different imaging approaches
● Define game conditionals and variables
● Create and call functions, conditionals, and user inputs
● Create a start screen with necessary game play instructions
● Develop and code all of the sprites and objects
● Create score, life count, and edges
● Make improvements based on peer feedback
● Research advanced programming approaches to address feedback and issues encountered during development
● Finalize a 10 level video game that includes sprite animation, visuals, logical story development and a variety of programming techniques
## Sample Course Sequence

<table>
<thead>
<tr>
<th>Day 1</th>
<th>Day 2</th>
<th>Day 3</th>
<th>Day 4</th>
<th>Day 5</th>
<th>Day 6</th>
<th>Day 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction and Overview</td>
<td>Develop Game Backstory, Sketch out visuals, visuals/game description</td>
<td>Define game conditionals, Variables, Functions</td>
<td>Present game plan idea to class</td>
<td>Review how to make a start screen, create and call functions, conditionals, and user inputs.</td>
<td>Create Welcome Screen with instructions in code studio</td>
<td>Complete Welcome Screen with instructions in code studio</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Day 8</th>
<th>Day 9</th>
<th>Day 10</th>
<th>Day 11</th>
<th>Day 12</th>
<th>Day 13</th>
<th>Day 14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sprites review</td>
<td>Develop and code all of the sprites and objects</td>
<td>Develop and code all of the sprites and objects</td>
<td>How to create Score, Life Count, Edges</td>
<td>Level 1 Begin creating level 1</td>
<td>Level 1 continue creating level 1</td>
<td>Level 1 finalize creating level 1</td>
</tr>
</tbody>
</table>

| Day 15 | Day 16 | Day 17 | Day 18 | Day 19 | Day 20 | Day 21 |
| Day 16 | Level 2-6 Create Levels 2-6 | Level 2-6 Create Levels 2-6 | Level 2-6 Create Levels 2-6 | Level 2-6 Create Levels 2-6 | Level 2-6 Create Levels 2-6 | Level 2-6 Create Levels 2-6 |

<table>
<thead>
<tr>
<th>Day 17</th>
<th>Day 18</th>
<th>Day 19</th>
<th>Day 20</th>
<th>Day 21</th>
<th>Day 22</th>
<th>Day 23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 2-6 Create Levels 2-6</td>
<td>Level 2-6 Create Levels 2-6</td>
<td>Level 2-6 Create Levels 2-6</td>
<td>Level 2-6 Create Levels 2-6</td>
<td>Level 2-6 Create Levels 2-6</td>
<td>Peer Review Of game</td>
<td>Make improvements based on peer feedback</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 7-10 Create Levels 7-10</td>
<td>Level 7-10 Create Levels 7-10</td>
<td>Level 7-10 Create Levels 7-10</td>
<td>Level 7-10 Create Levels 7-10</td>
<td>Level 7-10 Create Levels 7-10</td>
<td>Level 7-10 Completed 7-10</td>
<td>Peer Review Of game</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Day 31</th>
<th>Day 32</th>
<th>Day 33</th>
<th>Day 34</th>
<th>Day 35</th>
<th>Day 36</th>
<th>Day 37</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make improvements based on peer feedback</td>
<td>Self Reflection and Rubric Completion Game Due</td>
<td>Presentation of Game</td>
<td>Presentation of Game, cont.</td>
<td>Game Gallery</td>
<td>Berkeley Heights Public Schools</td>
<td>Day 38</td>
</tr>
</tbody>
</table>
SUGGESTED MATERIALS

Materials:

Aluminum foil
Plastic containers
Paper towels or rags
Pennies
Poster paper
Markers/colored pencils
Scissors
Glue and tape
Deck of cards
Packs of sticky notes
Design Journals
Desktop computer
iPad apps and peripherals

Websites:

https://curriculum.code.org/csd/resources/

https://www.code.org

https://www.codecademy.com/

https://www.youtube.com/watch?v=jQoNiU2ChoM

https://www.google.com/maps

https://www.youtube.com/watch?v=G41G_PEWFjE&feature=youtu.be

https://www.youtube.com/watch?v=y3rCKJN0wpA&feature=youtu.be

https://www.youtube.com/watch?v=jQoNiU2ChoM