Curriculum Description

Overview

'Games and Animations' - A project-oriented programming curriculum that introduces new programmers to the skills required to build fun animations and games on the HTML canvas using JavaScript. Students develop real and applicable programming proficiencies like: declaring and using variables, conditional statements, data types, objects, loops, arrays and functions through engaging lessons and projects.

The Games and Animations curriculum introduces these skills on a platform that includes an easy-to-use code editor, helpful error messages and robust documentation that lowers the learning curve for users. Students are quickly able to manipulate the canvas with graphics like shapes, images, and colors. Before long, students learn to build animations, develop logic statements and complex data structures within their programs. Students combine these skills to develop fun games in guided projects at the end of each instructional segment.

Organization

Units - The 'Games and Animations' curriculum is divided into two large units of study (soon to be three!). Unit 1 focuses on the basic skills required to build simple games. Unit 2 builds on the skills developed in Unit 1 by introducing more abstract concepts like for loops and complex conditional statements.

Stages - Within a unit, students progress through 5 thematic stages. Each stage introduces a handful of fundamental programming skills. Stages conclude with a ‘Challenge’. Students must demonstrate mastery of the skills introduced by passing the challenge to move on to the next stage.

Lessons – Each stage is composed of:

- 5 ‘Teaching lessons’ that introduce new programming skills.
- 5 ‘Practice lessons’ that provide extra practice.
- 1 ‘Review lesson’ that allows students to review material in a condensed format.

- 1 ‘Vocabulary quiz’ - test if students have retained important vocabulary.
- 1 ‘Challenge lesson’ that tests what students have learned in the stage.

Lessons vary in length but grow in complexity as the student progresses through a Stage. Every lesson (except Challenge lessons) include a ‘Show me’ feature that provides the correct line of code so students are never stuck on a task.

Guided Projects - The transition from curated lessons to a sandbox environment can be intimidating for students. Each stage three self-directed 'Guided projects' that ask students to use the skills they’ve learned to create a complex computer program in the Workshop.

Workshop – Blackbird School has a robust sandbox environment (the Workshop) where students can develop their own code using what they’ve learned in the lessons. Students can explore model programs in the Nest, share code with friends and classmates on the platform, and receive feedback on projects from instructors.

Educator Dashboard – The platform includes a full-featured learning management system (LMS) where teachers can create classes and invite students. Teachers get detailed information about student progress, like where a student is in the curriculum and how much time they’ve spent actively engaged with lessons and projects. Teachers can view student’s work in the Workshop and can comment on their code allowing for quick and responsive feedback.

Curriculum outline

Unit 1 – Simple games

- **Stage 1 – The canvas:** Students learn about the HTML canvas and how to place points, lines and text on it.
- **Stage 2 – Shapes and colors:** Students learn how to use functions to create colorful shapes and images on the canvas.
- **Stage 3 – Animation:** Students learn how to use the animate function to create engaging programs on the canvas.
- **Stage 4 – Variables and objects:** Students learn how about the data structures variables and objects.

- **Stage 5 – Simple games:** Students learn how to incorporate the skills they've learned to program simple games.

Unit 2 – More games

- **Stage 6 – For Loops:** Students learn how to construct repeating sections of code called for loops.
- **Stage 7 – Conditional statements:** Students learn more about different types of logic statements.
- **Stage 8 – Arrays:** Students learn to organize data in a list called an array.
- **Stage 9 – Loops and arrays:** Students learn how to build lists using loops.
- **Stage 10 – Advanced games:** Students learn how to incorporate the skills they've learned to program more complex games.

Computer Science Standards Addressed

The Games and Animations curriculum is suitable for students ages 11 and up. Below are the computer science standards addressed as identified in the CSTA K-12 Computer Science Standards for Level 2 with commentary about how the Games and Animations curriculum addresses each.

Level 2 Standards

- **Algorithms and Programming**
  - 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.
    - *Students are provided models of variable naming conventions throughout the Games and Animations curriculum.*
  - 2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
    - *Many lessons in Games and Animations demonstrate the iterative nature of programming by adding and refining portions of code written in previous lessons.*
  - 2-AP-13: Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
    - *Lessons in the Games and Animations curriculum feature 'Step tasks' that guide learners, line-by-line, through complex portions of code to*

demonstrate how the program works, and to help uncover problems in the code.

- 2-AP-14: Create procedures with parameters to organize code and make it easier to reuse.
  - *Students master the use of a limited set of functions, built-in objects and data structures so they are able to employ those skills in work of their own.*
- 2-AP-15: Seek and incorporate feedback from team members and users to refine a solution that meets user needs.
  - *All work completed in the Workshop on Blackbird School is viewable by instructors who have the ability to leave detailed code review. Projects are also viewable by friends and classmates on the platform to encourage collaboration and refinement of programs.*
- 2-AP-16: Incorporate existing code, media, and libraries into original programs, and give attribution.
  - *Each lesson in the Games and Animation curriculum guides students in the creation of a complete computer program. The structures modeled in these lessons allow students to apply them in their own work without the added complication of importing libraries or accessing other resources.*
- 2-AP-17: Systematically test and refine programs using a range of test cases.
  - *Allowing students to uncover problems in their code (debugging) is at the heart of the development of Blackbird School. The platform features a robust syntax checker, easy to interpret error messages and complete documentation. In addition the code editor allows students to step through their programs one line at a time. Lessons in the curriculum deliberately include bugs and highlight how a programmer can solve these issues in their code.*
- 2-AP-18: Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.
  - *The Workshop in Blackbird School allows students to easily access the work of their collaborators. Guided projects in the workshop provide opportunities for students to work on a common task with their classmates. Teachers are able to monitor this work and review code - all within the same application.*

- 2-AP-19: Document programs in order to make them easier to follow, test, and debug
  - *Lessons in the Games and Animation curriculum are all completely commented, provided ample models of proper documentation of code. With Code Review, teachers are able to monitor and reinforce proper documentation practices with their students.*

Logistics

- **Time requirements** - One semester or about 16-18 weeks.
- **Technology requirements** – Computers with internet (desktop or laptop), classroom with projector is a plus.