MT. DIABLO UNIFIED SCHOOL DISTRICT
COURSE OF STUDY

COURSE TITLE:	**Coding and Gaming**
COURSE NUMBER:	**009140**
CBEDS NUMBER:	**7219**
DEPARTMENT:	**Career Tech**
LENGTH OF COURSE:	**Year long**
CREDITS PER SEMESTER:	**5 credits**
GRADE LEVEL(S):	**10th**
REQUIRED OR ELECTIVE:	**Elective**

PREREQUISITES:
Required - **None**
Recommended - **Algebra 1A or Algebra I (recommended)**

BOARD OF EDUCATION ADOPTION: (Date of Action Meeting)

COURSE DESCRIPTION:

This course is an introductory programming and game development course for students with little or no experience. Using engineering, physics and mathematics concepts critical to game development, they will work individually and in collaborative teams to design digital games. Students will learn technical skills like programming, graphic design and animation, testing, debugging, and other skills that are needed for game development.

COURSE PURPOSE:

Coding and Gaming is a beginner level course for students wishing to explore computer programming and video game development. The purpose of the course is to prepare students to become pragmatic and analytical thinkers who are fluent in 21st century technological skills – as such the curriculum is tied very closely to real world applications and career options in computer programming and interactive design. Throughout the course students are asked to analyze problems in order to construct logical solutions based on concepts rooted in an understanding of fundamental computer science principles. Students will be asked to construct programs and applications in response to real and complex problems through project based learning. Students will also be expected to grow as collaborators in a technical context. Students will consistently participate in critiques, group projects and otherwise support one another. Upon completion of the Coding and Gaming course, students will demonstrate skill in using industry-standard development environments to design and program 2D games as well as an understanding of computer science concepts that are translatable to many different programming environments and languages. Thus students who take this course are beginning to explore interactive design career pathways such as video game and software application design, website architecture and webpage design, mobile application development, and technical project management.

COURSE OUTLINE:

Unit 1: Programming Languages

In this unit students will learn about different programming languages and the ways in which languages evolved from one another. They will learn the history of programming languages and understand similarities and differences between different styles of languages: scripting, procedural, and object-oriented programming, as

well as recognize the benefits and limitations of text-based and languages compared to drop-and-drop or WYSIWYG development environments. Students will be introduced to a text-based language such as C++ or a drag-and-drop programming environment (Scratch, Snap!, Blockly, etc.). By the end of the unit the students will know how to create a code project, add files to the project, write, compile, run, and debug a simple program.

Unit 2: Data Types and Operators

In this unit students will learn about data types and operators. Students will learn how to declare variables prior to using them in a program. They will learn about basic data types such as integers, doubles, Boolean, and strings, and they will learn about values that are appropriate for each type and recognize the consequences of confusing them. They will understand the difference between local and global variables. They will learn that certain characters have special meaning and the need for character escape sequences. The unit will introduce students to relational and logical operators, truth tables, and how the compiler interprets them in order to perform logic and mathematical operations.

Unit 3: Control Statements and the Order of a Program

In this unit students will learn how a program executes based off the order of statements and learn to generate and follow flow charts. They will also learn how to control the order of their program by using if, if else, if else if else statements. They will also learn how to use for loops, while loops, do-while loops, and switch case statements.

Unit 4: Arrays and Strings

In this unit students will learn that Arrays and Strings are a collection of variables of the same data type and learn to recognize scenarios in which the use of an array would be desirable. They will learn the necessary steps to create, populate, modify, and access a one dimensional array. Students will learn how to manipulate data types and convert values between them in order to combine data across types. Students will use concatenation and in-string searches, set variables from user keyboard input, and respond to user feedback to control program workflow.

Unit 5: Functions

In this unit students will learn about functions and subroutines, the building blocks of most programming languages. Students will learn that functions and subroutines are a collection of statements that can be called from anywhere in a program to execute a certain task, and will explore their value in organizing code structure and allowing for more efficient code reuse. Students will learn about the scope of local and global variables in a function. They will learn about function parameters/arguments, return types and function prototypes.

Unit 6: Classes and Object Oriented Programming

In this unit students will get an introduction to classes and object oriented programming. Students learn what an objects is, how a class defines the type, and how the object is the variable. They will learn how a class is similar to a data structure and that classes have access specifiers such as private, protected, and public members.

Unit 7: Gaming Technical Skills

In this unit students will learn how create games in a drag and drop development environment. They will program by using advanced coding methodologies (variables, functions, arrays, loops). They will learn how to follow technical and increasingly complex programming instructions in order and detail. Students will program

original game projects and become familiar and competent in using professional 2D game development engines such as GameMaker Pro, or Construct 2/3. They will learn how to work with key frame animations, sprites, layers, backgrounds, events and behaviors. They will learn how open files, save files, create and program original games, integrate separate files into a final game project, create and edit audio sound effects & music, and publish games to the desktop, web, or mobile devices.

Unit 8: Creative, Innovative, and Critical Thinking

In this unit students will engage in several larger scale projects to learn how to create an original game based off of the knowledge they gained in the previous units. Students will work both independently and in teams of three to four students. They will use the engineering design cycle within the development team to achieve directive of creating and original game. The students will use creativity and critical thinking, and learn how to communicate and collaborate with each other in a production team environment.

A final project is designed so the students can demonstrate they have mastered the learning objectives for a coding and gaming course. In this unit students will learn how to create an original game based off of the knowledge they gained in the previous units. They will use the engineering design cycle within the development team to achieve the directive of ideating and producing an original video game. A game proposal will be required along with a game submission, and intermittent status updates will be required over the course of the project. After successful completion of the project the students will present their games to the class, evaluate their classmates' work as well as perform team evaluations in a post-project reflection, and complete a course survey.

KEY ASSIGNMENTS:

Unit 1: Programming Languages
Sample Assignment 1: "Hello, World!"
In this assignment students will learn how to create their first programming assignment called "Hello World". This program displays the line of text "Hello World!" to the screen.

Sample Assignment 2: Using the Interface
In this assignment students use provided graphical elements to make a sprite character of an animal move back and forth on the project workspace in response to the keyboard arrow keys and import audio to make the dog "woof" when the spacebar is pressed.

Unit 2: Data Types and Operators
Sample Assignment 1: Math Operations
Create a program that will display the square and cube of the numbers 1 - 10.

1. There is no user input in this program.
2. You must create a variable that can be multiplied to find the square and cube.
    1. Create a variable called x and initialize it to 1
    2. Reinitialize the variable x to 2 and multiple again
    3. Repeat steps a and b until you initialize the variable x to 10.
3. Format the output like the image below.
    1. demonstrate character escape sequence for tabs
    2. demonstrate character escape sequence for newline
    3. You may not use **endl** in the program.
4. Submit a screenshot of your code and the output.

    A sample run of the program will look like the image below:

Number Square Cube

Number	Square	Cube
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216

Unit 3: Control Statements and the Order of a Program
Sample Assignment 1: Grade Calculator

1. Create a console application and save it **lastNameFirstInitial_ifStatement**
2. Create a program that stores three separate grades.
3. Ask the user to enter the three grades.
    1. Use a for loop, while loop or do-while loop to ask the user and store the three separate grades.
4. Calculate and store the average of the three grades.
5. Create an if statement that checks your grade and compares it to a certain percentage
    1. Based on the percentage, display the grade you received for the three assignments.
    2. Display a message to the user.
        1. A, Excellent Job!
        2. B, Good Job!
        3. C, Average, study more.
        4. D, Need to study.
        5. F, Failed, try studying next time.

Grading Scale:

- A >= 90
- B >= 80
- C >= 70
- D >= 60
- F < 60

Sample Assignment 2: Times Table Drill
Create an interface that will present the user with ten "times tables drill" questions, providing two random integers between 0 and 12. Depending on the user's response keyboard input, output whether their answer is correct or incorrect based on calculations performed in your program. Keep track of the number of correct and incorrect answers in global variables. After ten questions, end the game by telling the user their total correct and incorrect answers.

Unit 4: Arrays and Strings
Sample Assignment 1: Array Manipulation with Strings
In this program you will be reversing the words "Hello World!". There will be no user input in this program. First you will create an array of type char and initialize it with a string of characters "Hello World!".

There are many ways that you can do this:

1. char hello[] = "Hello World!"; //Compiler determines size and adds the null character.
2. char hello[] = {'H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd', '!', '\0'} //Compiler determines size, programmer explicitly initializes and adds null character to array. **This is very tedious to do and will cause errors if you forget to add the null character.**

char hello[13] = {'H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd', '!', '\0'} //programmer explicitly declares size of array, initializes elements, and adds null character to array. **This is even more tedious to do and will cause errors if you forget to add the null character or do not allocate a large enough array for the string of characters including the null character.**

Directions:

1. Create a character array that is initialized with the words "Hello World!"
2. display the original array using a for loop
3. create a second array that is declared to be the same size as the first array
4. Use a for loop to reverse the first array and initialize the second array. **You are not just reversing the output, you are reversing the index from the first array and the second array.**

Output:
Displaying the original array: Hello World!
Now let's reverse the words!
No Displaying the second array in reverse order: !dlroW olleH
Displaying the original array once more: Hello World!

Sample Assignment 2: Make a Magic 8-Ball
First, here's how a traditional Magic 8-ball works: http://en.wikipedia.org/wiki/Magic_8_ball

We will need ten or more answers of our own. You can use the traditional answers or make up new ones. Be creative! Instead of a Magic 8-Ball you could make a Fortune Cookie Machine, a Pokemon Name Generator, or something else. But it does need to have an indexed array to store the possible output values, a random number generator, use conditional logic and concatenation to generate the output, take advantage of variables to store values, and use operators to compare them.

Unit 5: Functions
Sample Assignment 1: Calculate an Area
In this assignment students will create a program that can calculate the area of a rectangle and the volume of a Cuboid.

Directions:

1. Create a console application and save as functions_areaVolume
2. User input: Ask the user to enter the length and width for a rectangle.
3. Create a function prototype for the area function.
4. Create a function to calculate the area of the rectangle based on the user input.
5. Return type: Function should be able to accept decimal values
   a. Function should return a decimal value back to the function call
   b. Store return value: Program should be able to store the return value from the area function in main.
6. Display the area of the rectangle.
7. Create another function that can calculate the volume of a Cuboid. Your function should accept three parameters.

Sample Assignment 2: Create an RPG Character
Student may work with a partner or work alone. Student will need to use **functions, logic, random numbers, variables**, and **concatenation.**

Characters in traditional RPGs have "stats" which define how good they are with certain skills- these are initially determined by rolling dice and then choosing a character class, or profession, that is a good match. In old-school games, these stats range from 3-18, based on rolling three 6-sided dice.

Write and call functions that will perform the following:

- For each of three traits-- STRENGTH, INTELLIGENCE, and DEXTERITY:
  - Roll three 6-sided dice
  - Add the three numbers up to make a stat value.
  - Store that value in a variable
- Select a character class based on the highest stat:
  - High strength makes a good **warrior**
  - High intelligence makes a good **wizard**
  - High dexterity makes a good **rogue**
- Depending on the main stat, determine how good a character is- something like:
  - 3-12 = poor
  - 13-14 = decent
  - 15-16 = good
  - 17-18 = great
  - For example, a character with a high stat of 14 strength would be a "pretty good warrior."
- Display the results by **concatenating** all this information into a sentence and present it to the user.

Unit 6: Classes and Object Oriented Programming
Sample Assignment 1: Class Creation
In this assignment the students will learn how to create a simple class of type student. They will learn how to create private data and how set and get the data.

Directions:

1. Create a console application and save as **lastNameLast5ID_studentClass**
2. Define a class called **Student**
3. create private data members to store:
   1. Student name
   2. Student I.D.
   3. Student Grade level
4. Create public member functions to:
   1. Set the student name
   2. Get the student name
   3. Set the student I.D.
   4. Get the student I.D.
   5. Set the student Grade Level
   6. Get the student Grade level
5. Create a class declaration. You are basically declaring a variable or object of the student type.
   1. Student **yourName**;
6. Set the student name to your name
7. Set the student I.D. to your student I.D.
8. Set the student grade level to your grade level.
9. Use the membership operator / dot operator to retrieve and display:
   1. the student name
   2. the student I.D.
   3. the student Grade level.

Unit 7: Gaming Technical Skills
This unit is divided into five project assignments. In the first three projects, the students will create three separate games that increase in difficulty and build upon the previous game. Students will be given a detailed pdf document that will guide them with detailed instructions that will walk them through the process of creating

a project, to adding, images and sound. The instructions will guide them through the drag and drop programming in which they will create events and actions that will determine the outcome of the game. The students will also receive all of the assets for the game via a compressed zip folder. In the last two projects, students will be given creative license to improve upon and enhance the basic products they developed in the first three sections, by building in additional challenges, logic, levels, audio and graphical sophistication, score keeping, splash screens, end game, etc., based on a requirements document listing expected mechanics. Students will present their final products to the class for a rough draft critique session, after which a final product will be presented and peer and self-assessments are performed.

Unit 8: Creative, Innovative, and Critical Thinking
This unit has four primary project assignments. In the first project, students work with a partner or small team to research, analyze, and attempt to reproduce a classic 2D arcade video game. In the second project, students work independently to create an "interactive storybook" through which they demonstrate skills in moving between multiple levels/layouts, managing text, and responding to user input outside of standard game design behaviors. In the third project, students will create an interactive animated sequence to demonstrate mastery of storyboarding, timing, event management, and manipulation of audio, visual, and text assets.

A final project is designed so the students can demonstrate they have mastered the learning objectives for a coding and gaming course. The students will develop a complete game in a small group. The project involves creating an original game based on the design engineering cycle. A game proposal will be required along with a game submission. After successful completion of the project the students will present their game to the class, evaluate their classmates work, and complete a course survey.

INSTRUCTIONS METHODS and/or STRATEGIES:

Guided Practice & Modeling:
Students follow steps illustrated by teacher on projector. The teacher and students will always have nearly identical views of the computer.

Cooperative Learning:
Students will always be grouped in groups of mixed abilities. Even during regular instruction time students will be consistently encouraged to help one another. When students ask questions, the teacher will encourage their assigned partner or group to help them come up with a solution.

Supervised Individual Application:
After periods of initial guided practice and modeling students will have time to practice the skills on their own terms. The teacher will circulate the room helping students achieve individual goals, troubleshooting technical issues, exploring possible logical strategies for solving problems, and/or coaching any students in need of more extensive lessons.

Self-Discovery:
When new tools or information are presented to the class, the teacher will strive to go beyond merely delivering the information. That is the teacher will always lead students to the correct methods by asking questions, probing and encouraging students to explore.

Structured Discussion:
During and after any period of instruction, students will always be encouraged to ask and answer questions. This is especially important during any analysis of a code or game sample. Students will always be placed into class and group discussion for information and analysis. For example, during analysis of video game genre, students will get into groups and have discussions about the discovery each student makes in their research to

generate a larger report to share with the class. Many discussions will have students act within a specific role (time keeper, recorder, skeptic, prober, etc.).

Student-Centered Critiques:
After key assignments and between drafts students will engage in a respectful but honest discussion on the quality of work turned in. Most of these critiques will be done verbally as a class. Each student's work will have a turn to be critiqued. The teacher will encourage a discussion of both the strengths and weaknesses of any given piece. Ideally, the teacher will facilitate the critiques but not necessarily be a principal participant. Critiques may also have a written component.

Project Based Learning:
A student and group-based approach that focuses on creative problem solving. The course consistently asks students to think of solutions to real world problems. As an instructional method, project based learning encourages students to take ownership of their learning in authentic situations. That is, when students are asked to directly engage with relevant problems in a meaningful way they are more likely to become independent 21st century thinkers.

ASSESSMENTS INCLUDING METHODS and/or TOOLS

Participation:
Each day students are expected to participate during instruction. Students are encouraged to follow along in guided practice and to ask/answer question. Keeping track of participation holds students accountable for learning on a day to day basis and makes contribution a part of the classroom environment.

Presentations:
Students present their research findings, inspiration, or code product to the rest of the class for peer feedback and teacher feedback. This occurs at least once per unit-- usually as the culmination of their learning. Students are graded with a rubric based on communication skills: public speaking, visual presentation and depth of understanding of the topic. This method holds students accountable for both individual and group work and forces students to take ownership of their work in front of their peers. Two more substantial end-of-semester projects provide higher stakes: these presentations are made to a mixture of family, community members, and professional judges.

Written Assignments:
Students are consistently asked to demonstrate their understanding in written format. These range from small daily assignments (warm ups, exit cards) to more involved assignments (research essays, written critiques, written self-evaluations). Students are graded with a rubric based on their ability to meet the goal of the assignment, which range from persuasion, objective criticism, informed analysis and reflection.

Individual Assignments:
Throughout the course students are given daily or weekly individual assignments to demonstrate their understanding of a particular programming or design skill. Many of the assignments happen in class but some require work outside of class. Students are graded with a rubric based on their ability to meet the goal of the assignment. These assessments enable students to explore tools and strategies on their own and sharpen their creative problem solving skills. In other words, these assignments allow students to discover learning and problem solving on their own.

Group Assignments:

These are assignments and projects that students work on in teams of two to four members. Each member will be expected to take on a specific role that complements their strengths. These projects enable students to demonstrate teamwork, communication, leadership, and many other CTE goals.

Project Based Learning:
A student and group-based approach that focuses on creative problem solving. The course consistently asks students to think of solutions to real world problems. As an assessment method, project based learning tests students in their capacity to take on real world roles in this CTE field. That is, these assessments measure not only student talent but also their capacity to take on collaborative career roles. Can students become effective leaders? Can they push themselves to work with others? Can they complete difficult work under competitive pressure?

INSTRUCTIONAL MATERIALS:

Text

Title	Author	Publisher	Edition	Website	Primary
C++ Programming From Problem Analysis to Program Design	D.S. Malik	Course Technology, Cengage Learning	4th Edition / 2009	http://www.cengage.com/search/productOverview.do?N=+16+4294922389+4294966920+110+4294921370&Ntk=P_Isbn13&Ntt=9781423902096&Ntx=mode%2Bmatchallpartial	Yes
A Theory of Fun for Game Design	Raph Koster	O'Reilly Media	2nd Edition/ 2013	NA	Yes

Websites and Software Resources:

Title	Author(s)/Editor(s)/Compiler(s)	Affiliated Institution or Organization	URL
Construct 2	Scirra Ltd	Scirra Ltd	https://www.scirra.com/construct2
Construct 3	Scirra Ltd	Scirra Ltd	https://editor.construct.net/
Scratch	Lifelong Kindergarten Group	MIT Media Lab	https://scratch.mit.edu/
Visual Studio Express	Microsoft Inc	Microsoft Inc	
C++ Reference	-	The C++ Resources Network	http://www.cplusplus.com

For CTE Pathway Distinction:

Sequence of Courses:
Interactive Design CTE Pathway
1) **Coding and Gaming**
2) **Webpage Design**
3) **Digital Innovation and Design**

Committee Members:

1. **Josie Kirkland, MDHS CTE teacher, Digital Safari Academy**
2. **Heather Fontanilla, Program Specialist for Career Pathways and Linked Learning**
3. **Katalina Gallo, MDHS VAPA teacher, Digital Safari Academy CTE teacher**
4. **Liane Cismowski, MDHS Principal**
5. **David Hevel, MDHS VAPA teacher**
6. **Erica Shaw, MDHS Social Studies teacher, Digital Safari Academy Lead Teacher**